OpenDNS DNS Database Client Library Documentation

Release 0.1

Frank Denis

October 21, 2016

CHAPTER 1

Installation

```
$ gem install opendns-dnsdb
```

CHAPTER 2

Example

```
# Setup
db = OpenDNS::DNSDB.new(sslcert: 'client.p12', sslcertpasswd: 'opendns')

# A short list of known spam domains using a fast-flux infrastructure
spam_names = ['com-garciniac.net', 'bbc-global.co.uk', 'com-october.net']

# Retrieve all the IP addresses these morons have been using
ips = db.distinct_ips_by_name(spam_names)

# Discover new domains mapping to the IP addresses we just found
all_spam_names = db.distinct_names_by_ip(ips)

# Find all the name servers used by these new domains
all_spam_names_ns = db.distinct_nameservers_ips_by_name(all_spam_names)

# Find all the domains served by these name servers
maybe_more_spam = db.distinct_names_by_nameserver_ip(all_spam_names_ns)

# Return the subset of names not flagged as malware by OpenDNS yet
not_blocked_yet = db.not_suspicious_names(maybe_more_spam)

# Does this list of domains include domains used by malware?
is_malware = db.include_suspicious?(['wh4u6igxiglekn.su', 'excue.ru'])

# Specifically, is excue.ru suspicious?
is_suspicious = db.is_suspicious?('excue.ru')

# Find all .ru names frequently observed with wh4u6igxiglekn.su and excue.ru:
rel_ru = db.distinct_related_names(['wh4u6igxiglekn.su', 'excue.ru'],
                                    max_names: 500,
                                    max_depth: 4) { |n| n.end_with? '.ru.' }

# Get the number of daily requests for the past 10 days, for
# github.com and github.io:
traffic = db.daily_traffic_by_name(['www.github.com', 'www.github.io'],
                                   days_back: 10)

# Cut the noise from this traffic - Days with less than 10 queries
traffic = db.high_pass_filter(traffic, cutoff: 10)

# Check if the traffic for github.io is suspiciously spiky:
traffic_is_suspicious =
  db.relative_standard_deviation(traffic['www.github.io']) > 90
```

CHAPTER 3

Parallel requests

This client library transparently supports parallel requests.

Most operations can be given either a single name or single IP, as well as a list of names or IPs. The library will transparently paralellize operations in order for bulk queries to complete as fast as possible.

Bulk operations can be performed on arbitrary large sets of names or IP addresses.

Setup

```
require 'opendns-dnsdb'
db = OpenDNS::DNSDB.new(sslcert: 'client.pem', sslcertpasswd: 'opendns')
```

Supported options:

- `timeout`: timeout for each query, in seconds (default: 15 seconds)
- `sslcert`: path to the SSL certificate
- `sslcerttype`: SSL certificate type, defaults to `pem`.
- `sslcertpasswd`: SSL certificate password
- `max_concurrency`: max number of parallel operations (default: 10)

CHAPTER 5

Note on certificates format

The curl library currently has some major issues dealing with certificates stored in the PKCS12 format.

If the certificate you have been given is in PKCS12 format (*.p12* file extension), just convert it to a *.pem* certificate file:

```
openssl pkcs12 -in client.p12 -out client.pem -clcerts
```

And supply the path to the `.pem` file to the library.

CHAPTER 6

Operations

6.1 Getting information out of a name

6.1.1 Getting the nameserver IP addresses for a name

```
db.nameservers_ips_by_name('github.com')
```

This returns a `Response::Distinct` of IP addresses seen for this name for the past 3 months, or an empty list if none have been seen.

```
[
    [0] "204.13.250.16",
    [1] "204.13.251.16",
    [2] "208.78.70.16",
    [3] "208.78.71.16"
]
```

6.1.2 Getting the nameserver IPs for a set of names

```
db.nameservers_ips_by_name(['github.com', 'github.io'])
```

This returns a `Response::HashByName`:

```
{
    "github.com" => [
        [0] "204.13.250.16",
        [1] "204.13.251.16",
        [2] "208.78.70.16",
        [3] "208.78.71.16"
    ],
     "github.io" => [
        [0] "204.13.250.16",
        [1] "204.13.251.16",
        [2] "208.78.70.16",
        [3] "208.78.71.16"
    ]
}
```

6.1.3 Getting a list of distinct name servers for a set of names

A very common need is to retrieve the list of IP unique addresses seen
for a set of domain names over the past 3 months.
This can be achieved as follows:

```
db.distinct_nameservers_ips_by_name(['github.com', 'github.io'])
```

Returns a `Response::Distinct`:

```
[
    [0] "204.13.250.16",
    [1] "204.13.251.16",
    [2] "208.78.70.16",
    [3] "208.78.71.16"
]
```

The output is always a `Response::Distinct` of distinct IP addresses.
This method also works with a single domain name, and is an alias for
`nameservers_ips_by_name` in that case.

6.1.4 Getting the list of IP addresses for a name

This returns the list of IP addresses seen over the past 3 months for
a name:

```
db.ips_by_name('github.com')
```

Returns a `Response::Distinct`“

```
[
    [0] "192.30.252.129",
    [1] "192.30.252.130",
    [2] "192.30.252.131",
    [3] "192.30.252.128",
    [4] "204.232.175.90",
    [5] "207.97.227.239"
]
```

6.1.5 Getting the list of IP addresses for a set of names

Bulk lookups can be achieved by providing a list instead of a string:

```
db.ips_by_name(['github.com', 'github.io'])
```

Returns a `Response::HashByName`:

```
{
    "github.com" => [
        [0] "192.30.252.129",
        [1] "192.30.252.130",
        [2] "192.30.252.131",
        [3] "192.30.252.128",
        [4] "204.232.175.90",
        [5] "207.97.227.239"
    ],
     "github.io" => [
        [0] "204.232.175.78"
    ]
}
```

6.1.6 Getting the list of unique IP addresses for a set of names

```
db.distinct_ips_by_name(['github.com', 'github.io'])
```

Returns a `Response::Distinct`:

```
[
    [0] "192.30.252.129",
    [1] "192.30.252.130",
    [2] "192.30.252.131",
    [3] "192.30.252.128",
    [4] "204.232.175.90",
    [5] "207.97.227.239",
    [6] "204.232.175.78"
]
```

6.1.7 Getting the list of mail exchangers for a name

```
db.mxs_by_name('github.com')
```

Returns a `Response::Distinct`:

```
[
    [0] "alt1.aspmx.l.google.com.",
    [1] "alt2.aspmx.l.google.com.",
    [2] "aspmx.l.google.com.",
    [3] "aspmx2.googlemail.com.",
    [4] "aspmx3.googlemail.com."
]
```

6.1.8 Getting the list of mail exchangers for a set of names

```
db.mxs_by_name(['github.com', 'github.io'])
```

Returns a `Response::HashByName`:

```
{
    "github.com" => [
        [0] "alt1.aspmx.l.google.com.",
        [1] "alt2.aspmx.l.google.com.",
        [2] "aspmx.l.google.com.",
        [3] "aspmx2.googlemail.com.",
        [4] "aspmx3.googlemail.com."
    ],
     "github.io" => []
}
```

6.1.9 Getting the list of unique mail exchangers for a set of names

```
db.distinct_mxs_by_name(['github.com', 'github.io'])
```

Returns a `Response::Distinct` of unique mail exchangers:

```
[
    [0] "alt1.aspmx.l.google.com.",
    [1] "alt2.aspmx.l.google.com.",
    [2] "aspmx.l.google.com.",
    [3] "aspmx2.googlemail.com.",
    [4] "aspmx3.googlemail.com."
]
```

6.1.10 Getting the list of CNAMEs for a name

```
db.cnames_by_name('www.skyrock.com')
```

Returns a `Response::Distinct` of CNAME records seen over the past 3 months for this name:

```
[
    [0] "skyrockv4.gslb.skyrock.net."
]
```

6.1.11 Getting the list of CNAMEs for a set of names

```
db.cnames_by_name(['www.skyrock.com', 'www.apple.com'])
```

Returns a `Response::HashByName`:

```
{
    "www.skyrock.com" => [
        [0] "skyrockv4.gslb.skyrock.net."
    ],
      "www.apple.com" => [
        [0] "www.isg-apple.com.akadns.net."
    ]
}
```

6.1.12 Getting the list of unique CNAMEs seen for a list of names

```
db.distinct_cnames_by_name(['www.skyrock.com', 'www.apple.com'])
```

Returns a `Response::Distinct`:

```
[
    [0] "skyrockv4.gslb.skyrock.net.",
    [1] "www.isg-apple.com.akadns.net."
]
```

6.2 Getting information out of an IP address

6.2.1 Getting the list of names served by a name server

This returns the list of names that have been served by an authoritative name server:

```
db.names_by_nameserver_ip('199.185.137.3')
```

Returns a `Response::Distinct`:

```
[
    [ 0] "openbsd.com.",
    [ 1] "openssh.com.",
    [ 2] "yycix.ca.",
    [ 3] "caisnet.com.",
    [ 4] "cdnpowerpac.com.",
    [ 5] "miarch.com.",
    [ 6] "openbsd.org.",
    [ 7] "theos.com.",
    [ 8] "enhanced-business.com.",
    [ 9] "onpa.ca.",
    [10] "openbsdfoundation.org.",
    [11] "eton-west.com.",
    [12] "barr-ryder.com.",
    [13] "chemco-elec.com.",
    [14] "rakeng.com.",
    [15] "yycix.com.",
    [16] "elementsustainable.com.",
    [17] "hartwigarchitecture.com.",
    [18] "pentagonstructures.com.",
    [19] "freezemaxwell.com.",
    [20] "workungarrick.com.",
    [21] "alpineheating.com.",
    [22] "caisnet.ca.",
    [23] "watertech.ca.",
    [24] "desco.cc.",
    [25] "openbsd.net.",
    [26] "krawford.com.",
    [27] "protostatix.com.",
    [28] "rms-group.ca.",
    [29] "cmroofing.ca.",
    [30] "hoeng.com.",
    [31] "openssh.net.",
    [32] "cuthbertsmith.com.",
    [33] "alta-tech.ca.",
    [34] "bockroofing.com."
]
```

6.2.2 Getting the list of names that a set of name servers have been serving

This returns the list of names that have been served by a set of name
servers:

```
db.names_by_nameserver_ip(['199.185.137.3', '65.19.167.109'])
```

Returns a `Response::HashByIP`:

```
{
    "199.185.137.3" => [
        [ 0] "openbsd.com.",
        [ 1] "openssh.com.",
        [ 2] "yycix.ca.",
        [ 3] "caisnet.com.",
        [ 4] "cdnpowerpac.com.",
        [ 5] "miarch.com.",
        [ 6] "openbsd.org.",
        [ 7] "theos.com.",
        [ 8] "enhanced-business.com.",
        [ 9] "onpa.ca.",
        [10] "openbsdfoundation.org.",
        [11] "eton-west.com.",
        [12] "barr-ryder.com.",
        [13] "chemco-elec.com.",
        [14] "rakeng.com.",
        [15] "yycix.com.",
        [16] "elementsustainable.com.",
        [17] "hartwigarchitecture.com.",
        [18] "pentagonstructures.com.",
        [19] "freezemaxwell.com.",
        [20] "workungarrick.com.",
        [21] "alpineheating.com.",
        [22] "caisnet.ca.",
        [23] "watertech.ca.",
        [24] "desco.cc.",
        [25] "openbsd.net.",
        [26] "krawford.com.",
        [27] "protostatix.com.",
        [28] "rms-group.ca.",
        [29] "cmroofing.ca.",
        [30] "hoeng.com.",
        [31] "openssh.net.",
        [32] "cuthbertsmith.com.",
        [33] "alta-tech.ca.",
        [34] "bockroofing.com."
    ],
    "65.19.167.109" => [
        [0] "backplane.com.",
        [1] "dragonflybsd.org."
    ]
}
```

6.2.3 Getting the list of unique names served by a set of name servers

This returns a `Response::Distinct` of unique names served by a set of name servers:

```
db.distinct_names_by_nameserver_ip(['199.185.137.3', '65.19.167.109'])
```

Returns a `Response::Distinct`:

```
[
    [ 0] "openbsd.com.",
    [ 1] "openssh.com.",
    [ 2] "yycix.ca.",
    [ 3] "caisnet.com.",
    [ 4] "cdnpowerpac.com.",
    [ 5] "miarch.com.",
    [ 6] "openbsd.org.",
    [ 7] "theos.com.",
    [ 8] "enhanced-business.com.",
    [ 9] "onpa.ca.",
    [10] "openbsdfoundation.org.",
    [11] "eton-west.com.",
    [12] "barr-ryder.com.",
    [13] "chemco-elec.com.",
    [14] "rakeng.com.",
    [15] "yycix.com.",
    [16] "elementsustainable.com.",
    [17] "hartwigarchitecture.com.",
    [18] "pentagonstructures.com.",
    [19] "freezemaxwell.com.",
    [20] "workungarrick.com.",
    [21] "alpineheating.com.",
    [22] "caisnet.ca.",
    [23] "watertech.ca.",
    [24] "desco.cc.",
    [25] "openbsd.net.",
    [26] "krawford.com.",
    [27] "protostatix.com.",
    [28] "rms-group.ca.",
    [29] "cmroofing.ca.",
    [30] "hoeng.com.",
    [31] "openssh.net.",
    [32] "cuthbertsmith.com.",
    [33] "alta-tech.ca.",
    [34] "bockroofing.com.",
    [35] "backplane.com.",
    [36] "dragonflybsd.org."
]
```

6.2.4 Getting the list of all names that resolved to an IP

This returns all the names that have been seen for an IP over the past
3 months:

```
db.names_by_ip('192.30.252.131')
```

Returns a `Response::Distinct`:

```
[
    [0] "github.com.",
    [1] "ip1d-lb3-prd.iad.github.com."
]
```

6.2.5 Getting the list of all names that resolved to a set of IPs

A bulk operation to retrieve the list of names having mapped to a set of IPs:

```
db.names_by_ip(['192.30.252.131', '199.233.90.68'])
```

Returns a `Response::HashByIP`:

```
{
    "192.30.252.131" => [
        [0] "github.com.",
        [1] "ip1d-lb3-prd.iad.github.com."
    ],
     "199.233.90.68" => [
        [0] "leaf.dragonflybsd.org."
    ]
}
```

6.2.6 Getting the list of unique names for a set of IPs

This method returns a list of distinct names seen for a set of IP addresses:

```
db.distinct_names_by_ip(['192.30.252.131', '199.233.90.68'])
```

Returns a `Response::Distinct`:

```
[
    [0] "github.com.",
    [1] "ip1d-lb3-prd.iad.github.com.",
    [2] "leaf.dragonflybsd.org."
]
```

6.3 Getting labels

6.3.1 Getting the label for a name

Domain names can be either benign (part of a whitelist), suspicious (flagged by the OpenDNS security team) or uncategorized.

This method returns the label for a given domain, which can be either `:suspicious`, `:benign` or `:unknown`.

```
db.label_by_name('github.com')
```

Returns a `Symbol`:

```
:benign
```

6.3.2 Getting the labels for a set of names

Domain names can be either benign (part of a whitelist), suspicious
(flagged by the OpenDNS security team) or uncategorized.

This method returns the labels for a set of names, which can be either
`:suspicious`, `:benign` or `:unknown`.

```
db.labels_by_name(['github.com', 'skyrock.com'])
```

The labels for up to 42,000 names can be queried at once.

Returns a `Response::HashByName`:

```
{
     "github.com" => :benign
    "skyrock.com" => :benign
}
```

6.3.3 Testing whether a set of names contains suspicious names

```
db.include_suspicious?(['github.com', 'skyrock.com'])
```

Returns `true` or `false`:

```
false
```

6.3.4 Testing whether a set of names contains benign names

```
db.include_benign?(['github.com', 'skyrock.com'])
```

Returns `true` or `false`:

```
true
```

6.3.5 Testing whether a set of names contains unknown names

```
db.include_unknown?(['github.com', 'skyrock.com'])
```

Returns `true` or `false`:

```
false
```

6.3.6 Testing whether a domain is suspicious

```
db.is_suspicious?('github.com')
```

Returns `true` or `false`:

```
false
```

6.3.7 Testing whether a domain is benign

```
db.is_benign?('github.com')
```

Returns `true` or `false`:

```
true
```

6.3.8 Testing whether a domain is unknown

```
db.is_unknown?('github.com')
```

Returns `true` or `false`:

```
false
```

6.3.9 Extracting the subset of suspicious names

Given a set of names, return a subset of names flagged as suspicious:

```
db.suspicious_names(['github.com', 'excue.ru'])
```

Returns a `Response::Distinct`:

```
['excue.ru']
```

6.3.10 Extracting the subset of names not flagged as suspicious

Given a set of names, return a subset of names not flagged as suspicious:

```
db.not_suspicious_names(['github.com', 'excue.ru'])
```

Returns a `Response::Distinct`:

```
['github.com']
```

6.3.11 Extracting the subset of benign names

Given a set of names, return a subset of names flagged as benign:

```
db.benign_names(['github.com', 'excue.ru'])
```

Returns a `Response::Distinct`:

```
['github.com']
```

6.3.12 Extracting the subset of names not flagged as benign

Given a set of names, return a subset of names not flagged as benign:

```
db.not_benign_names(['github.com', 'excue.ru'])
```

Returns a `Response::Distinct`:

```
['excue.ru']
```

6.3.13 Extracting the subset of unknown names

Given a set of names, return a subset of names flagged as unknown:

```
db.unknown_names(['github.com', 'exue.ru'])
```

Returns a `Response::Distinct`:

```
['exue.ru']
```

6.3.14 Extracting the subset of names flagged as benign or suspicious

Given a set of names, return a subset of names flagged as benign or suspicious:

```
db.not_unknown_names(['github.com', 'excue.ru'])
```

Returns a `Response::Distinct`:

```
['github.com', 'excue.ru']
```

6.3.15 Getting comments (attribution) about a set of names

Given a set of names, return a string summarizing all the comments (attribution) describing why each name was given a specific label:

```
db.comments_for_names(['trustsreaders.in', 'paybal.com'])
```

Distinct comment strings can be retrieved with:

```
db.distinct_comments_for_names(['trustsreaders.in', 'paybal.com'])
```

6.4 Related names

Related names are names that have been frequently observed shortly before or after a reference name.

This has proven to be very useful to discover command and control domains used by malware when only a few of them were previously known. This is also useful to investigate an infection chain.

Internally, multiple complementary matching algorithms are used, but this client library takes care of aggregating and normalizing the results.

6.4.1 Getting the list of related names

Related names for a single name can be looked up, as well as for a vector of names:

```
db.related_names('www.github.com')
db.relates_names(['www.github.com', 'www.mozilla.org')
```

These functions return a `Response::Distinct` object, if a single name was used as a starting point, or a `Response::HashByName` if a vector was provided.

The maximum number of results can be specified:

```
db.related_names('www.skyrock.com', max_names: 50)
```

An optional block can also be given.

This block is a filter: it will be given each (name, score) as an argument, and only names for which the return value of this block is not `false`/`nil` will be kept.

For example, this only retrieves names matching a given regular expression:

```
db.related_names('www.skyrock.com') { |name| name.match /^miss-/ }
```

And this only retrieves names whose score is more than 0.1:

```
db.related_names('www.skyrock.com') { |name, score| score > 0.1 }
```

6.4.2 Getting the list of related names, with scores

In addition to a list of names, a "score" can be returned for each name found. This score is in the [0.0, 1.0] range, 1.0 meaning that a name is likely to be closely related to the reference name, 0.0 meaning that these have not been observed together very frequently.

Related names for a single name can be looked up, as well as for a vector of names:

```
db.related_names_with_score('www.github.com')
db.relates_names_with_score(['www.github.com', 'www.mozilla.org')
```

These functions return a `Response::HashByName`.

An optional filter can be provided:

```
db.related_names_with_score('www.skyrock.com') do |name|
  name.match /^miss-/
end
```

6.4.3 Getting a set of distinct related names for a list of names

Given a list of names, this returns a set of names related to these.

```
db.distinct_related_names(['www.github.com', 'www.github.io'])
```

This returns a `Result::Distinct` object.

The maximum number of results can be specified:

```
db.distinct_related_names(['www.github.com', 'www.github.io'],
                          max_results: 250)
```

By default, only direct neighbors of the given names are returned, but deep traversal is also fully supported.

This will return a list of names related to those provided in the vector, but also names related to these newly found names, names related to these related names:

```
db.distinct_related_names(['www.github.com', 'www.github.io'],
                          max_results: 250,
                          max_depth: 3)
```

Since a deep traversal can return a lot of results, some not being of interest, a filter can be provided. This filter will be automatically applied after each iteration:

```
db.distinct_related_names(['www.github.com', 'www.github.io'],
                          max_results: 250,
                          max_depth: 3) do |name, score|
  name.match(/^com-/) && score > 0.1
end
```

A single name can also be given instead of a vector. This is equivalent to `related_names` when a deep traversal is not performed.

This function returns a `Response::Distinct` object.

6.5 DNS traffic

The number of DNS queries observed for a name over a time period can be retrieved.

This is especially useful to see if a domain is popular, and to spot anomalies in its traffic.

6.5.1 Getting the number of queries observed for a name

The `daily_traffic_by_name` method returns a vector with the number of queries observed for each day, within a time period.

By default, the time period starts 7 days before the current day, and ends at the current day, a day starting at 00:00 UTC.

```
db.daily_traffic_by_name('www.github.com')
```

The output is a `Result::TimeSeries` object:

```
[
    [0] 6152525,
    [1] 4756714,
    [2] 4670300,
    [3] 5954983,
    [4] 6140915,
    [5] 6040669,
    [6] 5529869
]
```

This method accepts several options:

- `start`: a `Date` object representing the lower bound of the time interval
- `end`: a `Date` object representing the higher bound of the time interval
- `days_back`: if `start` is not provided, this represents the number of days to go back in time.

Here are some examples featuring these options:

```
db.daily_traffic_by_name('www.github.com', end: Date.today - 2, days_back: 10)

db.daily_traffic_by_name('www.github.com', start: Date.today - 10)
```

The traffic for multiple domains can be looked up, provided that a vector is given instead of a single name. In that case, the output is a `Result::HashByName` object.

```
db.daily_traffic_by_name(['www.github.com', 'www.github.io'])
```

For example, the following snippet compares the median number of queries for a set of domains:

```
ts = db.daily_traffic_by_name(['www.github.com', 'www.github.io'])
ts.merge(ts) { |name, ts| ts.median.to_i }
```

```
{
    "www.github.com" => 5954983,
     "www.github.io" => 528002
}
```

6.5.2 Anomaly detection in traffic

A benign web site tends to have a comparable traffic every day. Sudden spikes or drop of traffic usually indicate a major event (incident, unusual volume of sent email), or some suspicious activity.

Domain names used as C&C typically receive very little traffic, and suddenly get a spike of traffic for a short period of time. The same can be observed with compromised hosts acting as intermediaries.

After having retrieved the traffic for a name, computing the relative standard deviation is a simple and efficient way to detect anomalies.

To do so, the library includes the `descriptive_statistics` module and implements a `relative_standard_deviation` method. This method can work on the time series of a single domain, as well as on a set of multiple time series.

```
ts = d.daily_traffic_by_name(['skyrock.com', 'github.com', 'ooctmxmgwigqt.info'])
ap d.relative_standard_deviation(ts)
```

This outputs either a `Response::TimeSeries` or a `Response::HashByName` object:

```
{
           "skyrock.com" => 2.4300100908269657,
            "github.com" => 10.628632305278618,
    "ooctmxmgwigqt.info" => 244.18566965045403
}
```

In this example, we can clearly spot a domain name whose traffic doesn't follow what we usually observe for a benign domain.

6.5.3 High-pass filter

Domains receiving little traffic are frequently receiving more noise (bots, internal traffic) than queries sent by actual users.

A simple high pass filter sets to 0 all entries of a time series below a cutoff value. This is provided by the `high_pass_filter` method:

```
ts = d.high_pass_filter(ts, cutoff: 5.0)
```

This method works on the time series of a single domain, as well as on a set of multiple time series. The result is either a *Response::TimeSeries* or a *Response::HashByName* object.