Package ‘gdns’

May 15, 2020

Title Tools to Work with Google's 'DNS'-over-'HTTPS' ('DoH') API

Version 0.5.0

Maintainer Bob Rudis <bob@rud.is>

Description To address the problem of insecurity of 'UDP'-based 'DNS' requests, Google Public 'DNS' offers 'DNS' resolution over an encrypted 'HTTPS' connection. 'DNS'-over-'HTTPS' greatly enhances privacy and security between a client and a recursive resolver, and complements 'DNSSEC' to provide end-to-end authenticated DNS lookups. Functions that enable querying individual requests that bulk requests that return detailed responses and bulk requests are both provided. Support for reverse lookups is also provided. See <https://developers.google.com/speed/public-dns/docs/dns-over-https> for more information.

License MIT + file LICENSE

LazyData true

Encoding UTF-8

Depends R (>= 3.5.0)

Imports httr, stats, jsonlite, stringi, magrittr, tinytest

RoxygenNote 7.1.0

NeedsCompilation no

Author Bob Rudis [aut, cre] (<https://orcid.org/0000-0001-5670-2640>)

Repository CRAN

Date/Publication 2020-05-15 14:00:03 UTC

R topics documented:

as.data.frame.gdns_response *Coerce a gdns query response answer to a data frame*

Description

Helper function to get to the ‘Answer‘ quickly

Usage

```
## S3 method for class 'gdns_response'
as.data.frame(x, ...)
```

Arguments

x	a ‘gdns_response‘ object
...	unused

bulk_query *Vectorized query, returning only answers in a data frame*

Description

Vectorized query, returning only answers in a data frame

Usage

```
bulk_query(
  entities,
  type = 1,
  cd = FALSE,
  do = FALSE,
  edns_client_subnet = "0.0.0.0/0"
)
```

Arguments

`entities` character vector of entities to query

`type` RR type can be represented as a number in [1, 65535] or canonical string (A, aaaa, etc). More information on RR types can be found here.

`cd` (Checking Disabled) flag. Use ‘TRUE‘ to disable DNSSEC validation; Default: ‘FALSE‘.

`do` (DNSSEC OK) flag. Use ‘TRUE‘ include DNSSEC records (RRSIG, NSEC, NSEC3); Default: ‘FALSE‘.

`edns_client_subnet`
The edns0-client-subnet option. Format is an IP address with a subnet mask. Examples: `1.2.3.4/24`, `2001:700:300::/48`.
If you are using DNS-over-HTTPS because of privacy concerns, and do not want any part of your IP address to be sent to authoritative nameservers for geographic location accuracy, use `edns_client_subnet=0.0.0.0/0`. Google Public DNS normally sends approximate network information (usually replacing the last part of your IPv4 address with zeroes). `0.0.0.0/0` is the default.

Value

`data.frame` of only answers (use `query()` for detailed responses)

Note

this is a fairly naive function. It expects `Answer` to be one of the return value list slots. The intent for it was to make it easier to do bulk forward queries. It will get smarter in future versions.

References

https://developers.google.com/speed/public-dns/docs/dns-over-https

Examples

```
if (tinytest::at_home()) {
  hosts <- c("rud.is", "r-project.org", "rstudio.com", "apple.com")
  gdns::bulk_query(hosts)
}
```

`dns_classes` *DNS CLASSes (dataset)*

Description

DNS CLASSes

Usage

```
data('dns_classes')
```

Format

data frame with columns: `decimal`, `hexadecimal`, `name`, `reference`

Note

As noted in , Multicast DNS can only carry DNS records with classes in the range 0-32767. Classes in the range 32768 to 65535 are incompatible with Multicast DNS.

Last updated 2019-06-27 11:16:48

References

`https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml#dns-parameters-2`

`https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml`

rfc6895

`dns_glob_names` *Underscored and Globally Scoped DNS Node Names (dataset)*

Description

Underscored and Globally Scoped DNS Node Names

Usage

```
data('dns_glob_names')
```

Format

data frame with columns: `rr_type`, `node_name`, `reference`

Note

Last updated 2019-06-27 11:16:48

References

`https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml#underscored-globally-scoped-dr`

`https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml`

rfc8552

`dns_opcodes` *DNS OpCodes (dataset)*

Description

DNS OpCodes

Usage

```
data('dns_opcodes')
```

Format

data frame with columns: `op_code`, `name`, `reference`

Note

Last updated 2019-06-27 11:16:48

References

https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml#dns-parameters-5

https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml

rfc6895, rfc1035

`dns_rcodes` *DNS RCODEs (dataset)*

Description

DNS RCODEs

Usage

```
data('dns_rcodes')
```

Format

data frame with columns: `rcode`, `name`, `description`, `reference`

Note

Last updated 2019-06-27 11:16:48

References

https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml#dns-parameters-6

https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml

rfc6895, rfc1035

`edns0_option_codes` *DNS EDNS0 Option Codes (OPT) (dataset)*

Description

DNS EDNS0 Option Codes (OPT)

Usage

```
data('edns0_option_codes')
```

Format

data frame with columns: `value`, `name`, `status`, `reference`

Note

Registrations made by standards-track documents are listed as "Standard," and by non-standards-track documents as "Optional." Registrations for which there are no final specifications are listed as "On-Hold."

Last updated 2019-06-27 11:16:48

References

https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml#dns-parameters-11

https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml

rfc6891, 3604

gdns *Tools to Work with Google DNS Over HTTPS API*

Description

Traditional DNS queries and responses are sent over UDP or TCP without encryption. This is vulnerable to eavesdropping and spoofing (including DNS-based Internet filtering). Responses from recursive resolvers to clients are the most vulnerable to undesired or malicious changes, while communications between recursive resolvers and authoritative nameservers often incorporate additional protection.

To address this problem, Google Public DNS offers DNS resolution over an encrypted HTTPS connection. DNS-over-HTTPS greatly enhances privacy and security between a client and a recursive resolver, and complements DNSSEC to provide end-to-end authenticated DNS lookups.

Support for reverse lookups is also provided.

See `https://developers.google.com/speed/public-dns/docs/dns-over-https` for more information.

Author(s)

Bob Rudis (bob@rud.is)

has_spf *Test for whether a DNS TXT record is an SPF record*

Description

Test for whether a DNS TXT record is an SPF record

Usage

```
has_spf(spf_rec)
```

Arguments

spf_rec	a character vector of DNS TXT records

Value

character vector

Examples

```
has_spf("v=spf1 include:_spf.apple.com include:_spf-txn.apple.com ~all")
```

is_soft_fail *SPF "all" type test*

Description

SPF "all" type test

Usage

```
is_soft_fail(spf_rec)

is_hard_fail(spf_rec)

passes_all(spf_rec)
```

Arguments

spf_rec	a character vector of DNS TXT records

Value

logical

Examples

```
is_soft_fail("v=spf1 include:_spf.apple.com include:_spf-txn.apple.com ~all")
is_hard_fail("v=spf1 include:_spf.apple.com include:_spf-txn.apple.com ~all")
passes_all("v=spf1 include:_spf.apple.com include:_spf-txn.apple.com ~all")
```

query *Perform DNS over HTTPS queries using Google*

Description

Traditional DNS queries and responses are sent over UDP or TCP without encryption. This is vulnerable to eavesdropping and spoofing (including DNS-based Internet filtering). Responses from recursive resolvers to clients are the most vulnerable to undesired or malicious changes, while communications between recursive resolvers and authoritative nameservers often incorporate additional protection.

To address this problem, Google Public DNS offers DNS resolution over an encrypted HTTPS connection. DNS-over-HTTPS greatly enhances privacy and security between a client and a recursive resolver, and complements DNSSEC to provide end-to-end authenticated DNS lookups.

Usage

```
query(
  name,
  type = "1",
  cd = FALSE,
  ct = "application/x-javascript",
  do = FALSE,
  edns_client_subnet = "0.0.0.0/0",
  random_padding = NULL
)

dig(
  name,
  type = "1",
  cd = FALSE,
  ct = "application/x-javascript",
  do = FALSE,
  edns_client_subnet = "0.0.0.0/0",
  random_padding = NULL
)
```

Arguments

`name` item to lookup. Valid characters are numbers, letters, hyphen, and dot. Length must be between 1 and 255. Names with escaped or non-ASCII characters are not supported. Internationalized domain names must use the punycode format (e.g. "`xn--qxam`").

If an IPv4 string is input, it will be transformed into a proper format for reverse lookups.

`type` RR type can be represented as a number in [1, 65535] or canonical string (A, aaaa, etc). More information on RR types can be found here. You can use `255` for an `ANY` query.

`cd` (Checking Disabled) flag. Use ‘TRUE‘ to disable DNSSEC validation; Default: ‘FALSE‘.

`ct` (Content Type) Desired content type option. Use ‘application/dns-message‘ to receive a binary DNS message in the response HTTP body instead of JSON text. Use ‘application/x-javascript‘ (the default) to explicitly request JSON text. Other content type values are ignored and default JSON content is returned.

`do` (DNSSEC OK) flag. Use ‘TRUE‘ include DNSSEC records (RRSIG, NSEC, NSEC3); Default: ‘FALSE‘.

`edns_client_subnet`

The edns0-client-subnet option. Format is an IP address with a subnet mask. Examples: `1.2.3.4/24`, `2001:700:300::/48`.

If you are using DNS-over-HTTPS because of privacy concerns, and do not want any part of your IP address to be sent to authoritative nameservers for geographic location accuracy, use `edns_client_subnet=0.0.0.0/0`. Google Public DNS normally sends approximate network information (usually replacing the last part of your IPv4 address with zeroes). `0.0.0.0/0` is the default.

`random_padding` clients concerned about possible side-channel privacy attacks using the packet sizes of HTTPS GET requests can use this to make all requests exactly the same size by padding requests with random data. To prevent misinterpretation of the URL, restrict the padding characters to the unreserved URL characters: upper- and lower-case letters, digits, hyphen, period, underscore and tilde.

Details

To perform vectorized queries with only answers (and no metadata) use `bulk_query()`).

Value

a `list` with the query result or `NULL` if an error occurred

References

<https://developers.google.com/speed/public-dns/docs/doh/json>

Examples

```
if (tinytest::at_home()) {
  query("rud.is")
  dig("example.com", "255") # ANY query
  query("microsoft.com", "MX")
  dig("google-public-dns-a.google.com", "TXT")
  query("apple.com")
  dig("17.142.160.59", "PTR")
}
```

`resource_record_tbl` *An overview of resource records (RRs) permissible in zone files of the Domain Name System (DNS)*

Description

A dataset containing the DNS resource record types, names, description and purpose

Usage

```
resource_record_tbl
```

Format

A data frame with 39 rows and 4 variables:

type numeric type of the resource record

name short name of the resource record

description short description of the resource record

purpose long-form description of the resource record purpose/function/usage

Source

https://en.wikipedia.org/wiki/List_of_DNS_record_types

`rrtypes` *Resource Record (RR) TYPEs (dataset)*

Description

Resource Record (RR) TYPEs

Usage

```
data('rrtypes')
```

Format

data frame with columns: `type`, `value`, `meaning`, `reference`, `template`, `registration_date`

Note

Last updated 2019-06-27 11:16:48

References

https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml#dns-parameters-4

https://www.iana.org/assignments/dns-parameters/dns-parameters.xhtml

rfc6895, rfc1035

spf_ipv4s *SPF field extraction functions*

Description

Various helper functions to extract SPF record components.

Usage

```
spf_ipv4s(spf_rec)

spf_ipv6s(spf_rec)

spf_includes(spf_rec)

spf_ptrs(spf_rec)

spf_exists(spf_rec)
```

Arguments

`spf_rec`	a character vector of DNS TXT records

Value

list; each element is a character vector of the specified component spf_ipv4s("v=spf1 +mx ip4:214.3.140.16/32 ip4:214.3.140.255/32 ip4:214.3.115.12/32")

split_spf *Split out all SPF records in a domain's TXT record*

Description

Given a vector of TXT records, this function will return a list of vectors of all the SPF records for each. If the given TXT record is not an SPF record, `NA` is returned (which makes it easy to skip with `purrr` functions).

Usage

```
split_spf(spf_rec)
```

Arguments

`spf_rec`	a character vector of DNS TXT records

Value

list; each element is chr vector of spf components

Examples

```
split_spf("v=spf1 include:_spf.apple.com include:_spf-txn.apple.com ~all")
```

Index