Package 'gdns'

July 3, 2017

Title Tools to Work with Google DNS Over HTTPS API

Version 0.2.1

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 AGPL + file LICENSE

LazyData true

Encoding UTF-8

Depends R (>= 3.0.0)

Suggests testthat

Imports httr, jsonlite, purrr, stringi, tibble, dplyr

RoxygenNote 6.0.1

NeedsCompilation no

Author Bob Rudis [aut, cre]

Repository CRAN

Date/Publication 2017-07-02 22:07:06 UTC

R topics documented:

`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, 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.
`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

```
hosts <- c("rud.is", "dds.ec", "r-project.org", "rstudio.com", "apple.com")
gdns::bulk_query(hosts)
```

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

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

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", edns_client_subnet = "0.0.0.0/0")
```

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.

`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.

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/dns-over-https

Examples

```
query("rud.is")
query("example.com", "255") # ANY query
query("microsoft.com", "MX")
query("google-public-dns-a.google.com", "TXT")
query("apple.com")
query("17.142.160.59", "PTR")
```

---

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

`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

Index