dev 3 Branches 19 Tags Go to file Code

dependabot[bot] Merge pull request #683 from projectdiscovery/dependabot/go_mo...
dd1f2f9 · 5 days ago

.github	Use asnmap latest api (#581)	4 months ago
cmd	Merge pull request #410 from jh...	last year
integration_tests	Updating GH workflows/Actions...	2 years ago
internal	Remove default noerror filter (#...	2 months ago
libs/dnsx	introduce query type selection (...	3 months ago
static	more ups	4 years ago
.gitignore	Update utils helpers library #262...	2 years ago
.goreleaser.yml	workflow updates (#456)	9 months ago
Dockerfile	docker build + go mod fix + no c...	3 months ago
LICENSE.md	Update LICENSE.md	3 years ago
Makefile	Disable static compilation for os...	2 years ago
README.md	Update README.md	3 months ago
SECURITY.md	Create SECURITY.md	3 years ago
go.mod	chore(deps): bump github.com/...	5 days ago
go.sum	chore(deps): bump github.com/...	5 days ago

README Code of conduct More

dnsx

A fast and multi-purpose DNS toolkit designed for running DNS queries

go report A+ contributions welcome release v1.2.1 Follow @pdiscoveryio chat 834 on

Features • Installation • Usage • Running `dnsx` • Wildcard • Notes • Join Disc

`dnsx` is a fast and multi-purpose DNS toolkit designed for running various probes through the retryabledns library. It supports multiple DNS queries, user supplied resolvers, DNS wildcard filtering like shuffledns etc.

Features

About

dnsx is a fast and multi-purpose DNS toolkit allow to run multiple DNS queries of your choice with a list of user-supplied resolvers.

docs.projectdiscovery.io/tools/dnsx

#cli #dns-client #dns-resolution #dns-bruteforcer #dns-records #wildcard-filtering

Readme
MIT license
Code of conduct
Security policy
Activity
Custom properties
2k stars
37 watching
231 forks
Report repository

Releases 19

v1.2.1 Latest
on Mar 4

+ 18 releases

Packages

Used by 242

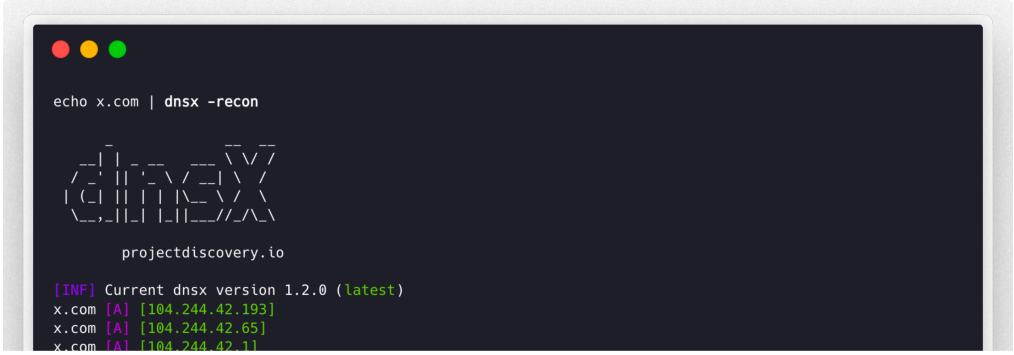
+ 234

Contributors 24

+ 10 contributors

Languages

Go 97.2% Shell 1.7% Other 1.1%



- Simple and Handy utility to query DNS records.
- **A, AAAA, CNAME, PTR, NS, MX, TXT, SRV, SOA** query support
- DNS **Resolution** / **Brute-force** support
- Custom **resolver** input support
- Multiple resolver format **(TCP/UDP/DOH/DOT)** support
- **stdin** and **stdout** support
- Automatic **wildcard** handling support

Installation Instructions

`dnsx` requires **go1.21** to install successfully. Run the following command to install the latest version:

```
go install -v github.com/projectdiscovery/dnsx/cmd/dnsx@latest
```

Usage

```
dnsx -h
```

This will display help for the tool. Here are all the switches it supports.

```
INPUT:
   -l, -list string      list of sub(domains)/hosts to resolve (file or stdin)
   -d, -domain string    list of domain to bruteforce (file or comma separated or stdin)
   -w, -wordlist string  list of words to bruteforce (file or comma separated or stdin)

QUERY:
   -a                    query A record (default)
   -aaaa                 query AAAA record
   -cname                query CNAME record
   -ns                   query NS record
   -txt                  query TXT record
   -srv                  query SRV record
   -ptr                  query PTR record
   -mx                   query MX record
   -soa                  query SOA record
   -any                  query ANY record
   -axfr                 query AXFR
   -caa                  query CAA record
   -recon                query all the dns records (a,aaaa,cname,ns,txt,srv,ptr,mx,soa,axfr,caa)
```

```
   -e, -exclude-type value  dns query type to exclude (a,aaaa,cname,ns,txt,srv,ptr,mx,soa,axfr,caa) (default none)

FILTER:
   -re, -resp          display dns response
   -ro, -resp-only     display dns response only
   -rc, -rcode string  filter result by dns status code (eg. -rcode noerror,servfail,refused)

PROBE:
   -cdn  display cdn name
   -asn  display host asn information

RATE-LIMIT:
   -t, -threads int      number of concurrent threads to use (default 100)
   -rl, -rate-limit int  number of dns request/second to make (disabled as default) (default -1)

UPDATE:
   -up, -update                 update dnsx to latest version
   -duc, -disable-update-check  disable automatic dnsx update check

OUTPUT:
   -o, -output string  file to write output
   -j, -json           write output in JSONL(ines) format
   -omit-raw, -or      omit raw dns response from jsonl output

DEBUG:
   -hc, -health-check  run diagnostic check up
   -silent             display only results in the output
   -v, -verbose        display verbose output
   -raw, -debug        display raw dns response
   -stats              display stats of the running scan
   -version            display version of dnsx
   -nc, -no-color      disable color in output

OPTIMIZATION:
   -retry int                number of dns attempts to make (must be at least 1) (default 2)
   -hf, -hostsfile           use system host file
   -trace                    perform dns tracing
   -trace-max-recursion int  Max recursion for dns trace (default 32767)
   -resume                   resume existing scan
   -stream                   stream mode (wordlist, wildcard, stats and stop/resume will be disabled)

CONFIGURATIONS:
   -auth                         configure projectdiscovery cloud (pdcp) api key (default true)
   -r, -resolver string          list of resolvers to use (file or comma separated)
   -wt, -wildcard-threshold int  wildcard filter threshold (default 5)
   -wd, -wildcard-domain string  domain name for wildcard filtering (other flags will be ignored - only json output i
```

Running dnsx

DNS Resolving

Filter active hostnames from the list of passive subdomains, obtained from various sources:

```
subfinder -silent -d hackerone.com | dnsx -silent

a.ns.hackerone.com
www.hackerone.com
api.hackerone.com
docs.hackerone.com
mta-sts.managed.hackerone.com
mta-sts.hackerone.com
resources.hackerone.com
b.ns.hackerone.com
mta-sts.forwarding.hackerone.com
events.hackerone.com
support.hackerone.com
```

Print **A** records for the given list of subdomains:

```
subfinder -silent -d hackerone.com | dnsx -silent -a -resp

www.hackerone.com [104.16.100.52]
www.hackerone.com [104.16.99.52]
hackerone.com [104.16.99.52]
```

```
hackerone.com [104.16.100.52]
api.hackerone.com [104.16.99.52]
api.hackerone.com [104.16.100.52]
mta-sts.forwarding.hackerone.com [185.199.108.153]
mta-sts.forwarding.hackerone.com [185.199.109.153]
mta-sts.forwarding.hackerone.com [185.199.110.153]
mta-sts.forwarding.hackerone.com [185.199.111.153]
a.ns.hackerone.com [162.159.0.31]
resources.hackerone.com [52.60.160.16]
resources.hackerone.com [3.98.63.202]
resources.hackerone.com [52.60.165.183]
resources.hackerone.com [read.uberflip.com]
mta-sts.hackerone.com [185.199.110.153]
mta-sts.hackerone.com [185.199.111.153]
mta-sts.hackerone.com [185.199.109.153]
mta-sts.hackerone.com [185.199.108.153]
gslink.hackerone.com [13.35.210.17]
gslink.hackerone.com [13.35.210.38]
gslink.hackerone.com [13.35.210.83]
gslink.hackerone.com [13.35.210.19]
b.ns.hackerone.com [162.159.1.31]
docs.hackerone.com [185.199.109.153]
docs.hackerone.com [185.199.110.153]
docs.hackerone.com [185.199.111.153]
docs.hackerone.com [185.199.108.153]
support.hackerone.com [104.16.51.111]
support.hackerone.com [104.16.53.111]
mta-sts.managed.hackerone.com [185.199.108.153]
mta-sts.managed.hackerone.com [185.199.109.153]
mta-sts.managed.hackerone.com [185.199.110.153]
mta-sts.managed.hackerone.com [185.199.111.153]
```

Extract **A** records for the given list of subdomains:

```
subfinder -silent -d hackerone.com | dnsx -silent -a -resp-only

104.16.99.52
104.16.100.52
162.159.1.31
104.16.99.52
104.16.100.52
185.199.110.153
185.199.111.153
185.199.108.153
185.199.109.153
104.16.99.52
104.16.100.52
104.16.51.111
104.16.53.111
185.199.108.153
185.199.111.153
185.199.110.153
185.199.111.153
```

Extract **CNAME** records for the given list of subdomains:

```
subfinder -silent -d hackerone.com | dnsx -silent -cname -resp

support.hackerone.com [hackerone.zendesk.com]
resources.hackerone.com [read.uberflip.com]
mta-sts.hackerone.com [hacker0x01.github.io]
mta-sts.forwarding.hackerone.com [hacker0x01.github.io]
events.hackerone.com [whitelabel.bigmarker.com]
```

Extract **ASN** records for the given list of subdomains:

```
subfinder -silent -d hackerone.com | dnsx -silent  -asn

b.ns.hackerone.com [AS13335, CLOUDFLARENET, US]
a.ns.hackerone.com [AS13335, CLOUDFLARENET, US]
hackerone.com [AS13335, CLOUDFLARENET, US]
www.hackerone.com [AS13335, CLOUDFLARENET, US]
```

```
api.hackerone.com [AS13335, CLOUDFLARENET, US]
support.hackerone.com [AS13335, CLOUDFLARENET, US]
```

Probe using dns status code on given list of (sub)domains:

```
subfinder -silent -d hackerone.com | dnsx -silent -rcode noerror,servfail,refused

ns.hackerone.com [NOERROR]
a.ns.hackerone.com [NOERROR]
b.ns.hackerone.com [NOERROR]
support.hackerone.com [NOERROR]
resources.hackerone.com [NOERROR]
mta-sts.hackerone.com [NOERROR]
www.hackerone.com [NOERROR]
mta-sts.forwarding.hackerone.com [NOERROR]
docs.hackerone.com [NOERROR]
```

Extract subdomains from given network range using `PTR` query:

```
echo 173.0.84.0/24 | dnsx -silent -resp-only -ptr

cors.api.paypal.com
trinityadminauth.paypal.com
cld-edge-origin-api.paypal.com
appmanagement.paypal.com
svcs.paypal.com
trinitypie-serv.paypal.com
ppn.paypal.com
pointofsale-new.paypal.com
pointofsale.paypal.com
slc-a-origin-pointofsale.paypal.com
fpdbs.paypal.com
```

Extract subdomains from given ASN using `PTR` query:

```
echo AS17012 | dnsx -silent -resp-only -ptr

apiagw-a.paypal.com
notify.paypal.com
adnormserv-slc-a.paypal.com
a.sandbox.paypal.com
apps2.paypal-labs.com
pilot-payflowpro.paypal.com
www.paypallabs.com
paypal-portal.com
micropayments.paypal-labs.com
minicart.paypal-labs.com
```

DNS Bruteforce

Bruteforce subdomains for given domain or list of domains using `d` and `w` flag:

```
dnsx -silent -d facebook.com -w dns_worldlist.txt

blog.facebook.com
booking.facebook.com
api.facebook.com
analytics.facebook.com
beta.facebook.com
apollo.facebook.com
ads.facebook.com
box.facebook.com
alpha.facebook.com
apps.facebook.com
connect.facebook.com
c.facebook.com
careers.facebook.com
code.facebook.com
```

Bruteforce targeted subdomain using single or multiple keyword input, as `d` or `w` flag supports file or comma separated keyword inputs:

```
dnsx -silent -d domains.txt -w jira,grafana,jenkins

grafana.1688.com
grafana.8x8.vc
grafana.airmap.com
grafana.aerius.nl
jenkins.1688.com
jenkins.airbnb.app
jenkins.airmap.com
jenkins.ahn.nl
jenkins.achmea.nl
jira.amocrm.com
jira.amexgbt.com
jira.amitree.com
jira.arrival.com
jira.atlassian.net
jira.atlassian.com
```

Values are accepted from **stdin** for all the input types (`-list` , `-domain` , `-wordlist`). The `-list` flag defaults to `stdin` , but the same can be achieved for other input types by adding a `-` (dash) as parameter:

```
cat domains.txt | dnsx -silent -w jira,grafana,jenkins -d -

grafana.1688.com
grafana.8x8.vc
grafana.airmap.com
grafana.aerius.nl
jenkins.1688.com
jenkins.airbnb.app
jenkins.airmap.com
jenkins.ahn.nl
jenkins.achmea.nl
jira.amocrm.com
jira.amexgbt.com
jira.amitree.com
jira.arrival.com
jira.atlassian.net
jira.atlassian.com
```

DNS Bruteforce with Placeholder based wordlist

```
$ cat tld.txt

com
by
de
be
al
bi
cg
dj
bs
```

```
dnsx -d google.FUZZ -w tld.txt -resp

     _                __  __
   __| | _ __   ___ \ \/ /
  / _' || '_ \ / __| \  /
 | (_| || | | |\__ \ /  \
  \__,_||_| |_||___//_/\_\ v1.1.2

     projectdiscovery.io

google.de [142.250.194.99]
google.com [142.250.76.206]
google.be [172.217.27.163]
google.bs [142.251.42.35]
google.bi [216.58.196.67]
google.al [216.58.196.68]
google.by [142.250.195.4]
google.cg [142.250.183.131]
google.dj [142.250.192.3]
```

Wildcard filtering

A special feature of `dnsx` is its ability to handle **multi-level DNS based wildcards**, and do it so with a very reduced number of DNS requests. Sometimes all the subdomains will resolve, which leads to lots of garbage in the output. The way `dnsx` handles this is by keeping track of how many subdomains point to an IP and if the count of the subdomains increase beyond a certain threshold, it will check for wildcards on all the levels of the hosts for that IP iteratively.

```
dnsx -l subdomain_list.txt -wd airbnb.com -o output.txt
```

Dnsx as a library

It's possible to use the library directly in your golang programs. The following code snippets is an example of use in golang programs. Please refer to here for detailed package configuration and usage.

```
package main

import (
        "fmt"

        "github.com/projectdiscovery/dnsx/libs/dnsx"
)

func main() {
        // Create DNS Resolver with default options
        dnsClient, err := dnsx.New(dnsx.DefaultOptions)
        if err != nil {
                fmt.Printf("err: %v\n", err)
                return
        }

        // DNS A question and returns corresponding IPs
        result, err := dnsClient.Lookup("hackerone.com")
        if err != nil {
                fmt.Printf("err: %v\n", err)
                return
        }
        for idx, msg := range result {
                fmt.Printf("%d: %s\n", idx+1, msg)
        }

        // Query
        rawResp, err := dnsClient.QueryOne("hackerone.com")
        if err != nil {
                fmt.Printf("err: %v\n", err)
                return
        }
        fmt.Printf("rawResp: %v\n", rawResp)

        jsonStr, err := rawResp.JSON()
        if err != nil {
                fmt.Printf("err: %v\n", err)
                return
        }
        fmt.Println(jsonStr)

        return
}
```

📋 Notes

- As default, `dnsx` checks for **A** record.
- As default `dnsx` uses Google, Cloudflare, Quad9 resolver.
- Custom resolver list can be loaded using the `r` flag.