
Async DNS Brute

A Python 3.5+ tool that uses asyncio to brute force domain names asynchronously.

```
dnstun  black ~ $ echo 1.1.1.1 | aiodnsbrute -r - google.ca
[*] Brute forcing google.ca with a maximum of 512 concurrent tasks...
[*] Using local resolver to verify google.ca exists.
[*] Using recursive DNS with the following servers: ['1.1.1.1']
[*] No wildcard response was detected for this domain.
[*] Wordlist loaded, proceeding with 1000 DNS requests
[+] www.google.ca 172.217.10.67
[+] m.google.ca 172.217.10.107
[+] store.google.ca 172.217.12.174
[+] news.google.ca 172.217.12.163
[+] mobile.google.ca 172.217.9.228
[+] apps.google.ca 172.217.6.238
[+] images.google.ca 172.217.10.142
[+] video.google.ca 172.217.6.206
[+] photos.google.ca 172.217.11.46
[+] image.google.ca 172.217.10.142
[+] archive.google.ca 172.217.12.206
[+] edu.google.ca 172.217.6.238
[+] photo.google.ca 172.217.11.46
[+] sms.google.ca 172.217.9.228
[+] book.google.ca 172.217.10.142
[+] w.google.ca 172.217.10.99
[+] ww.google.ca 172.217.11.3
[+] id.google.ca 172.217.6.227
[+] local.google.ca 172.217.11.14
[+] directory.google.ca 172.217.6.238
[+] work.google.ca 172.217.6.238
[+] map.google.ca 172.217.11.14
[+] desktop.google.ca 172.217.10.132
[+] maps.google.ca 172.217.11.14
[+] accounts.google.ca 172.217.11.35
[+] finance.google.ca 172.217.6.238
100% | 1000/1000 [00:06<00:00, 1202.84records/s]
[*] Completed, 26 subdomains found
dnstun  black ~ $
```

Speed

It's fast. Benchmarks on small VPS hosts put around 100k DNS resolutions at 1.5-2mins. An amazon M3 box was used to make 1 mil requests in just over 3 minutes. Your mileage may vary. It's probably best to avoid using Google's resolvers if you're purely interested in speed.

DISCLAIMER - Your ISP's and home router's DNS servers probably *suck*. Stick to a VPS with fast resolvers (or set up your own) if you're after speed. - **WARNING** This tool is capable of sending LARGE amounts of DNS traffic. I am not responsible if you DoS someone's DNS servers.

Installation

```
1 $ pip install aiodnsbrute
```

Note: using a virtualenv is highly recommended.

Alternate install

Alternately you can install the usual way:

```
1 $ git clone https://github.com/blark/aiodnsbrute.git
2 $ cd aiodnsbrute
3 $ python setup.py install .
```

Usage

Get help:

```
1 $ aiodnsbrute --help
2
3 Usage: cli.py [OPTIONS] DOMAIN
4
5 aiodnsbrute is a command line tool for brute forcing domain names
6 utilizing Python's asyncio module.
7
8 credit: blark (@markbaseggio)
9
10 Options:
11   -w, --wordlist TEXT           Wordlist to use for brute force.
12   -t, --max-tasks INTEGER       Maximum number of tasks to run
13   -r, --resolver-file FILENAME A text file containing a list of DNS
14                                resolvers
15                                to use, one per line, comments start
16                                with #.
17                                Default: use system resolvers
18   -v, --verbosity               Increase output verbosity
19   -o, --output [csv|json|off]   Output results to DOMAIN.csv/json (
20                                extension
21                                automatically appended when not using -
22                                f).
23   -f, --outfile FILENAME        Output filename. Use '-f -' to send
24                                file
25                                output to stdout overriding normal
26                                output.
27   --query / --gethostbyname     DNS lookup type to use query (default)
28                                should
29                                be faster, but won't return CNAME
30                                information.
31   --wildcard / --no-wildcard    Wildcard detection, enabled by default
32   --verify / --no-verify       Verify domain name is sane before
33                                beginning,
34                                enabled by default
35   --version                     Show the version and exit.
36   --help                       Show this message and exit.
```

Examples

Run a brute force with some custom options:

```
1 $ aiodnsbrute -w wordlist.txt -vv -t 1024 domain.com
```

Run a brute force, suppress normal output and send only JSON to stdout:

```
1 $ aiodnsbrute -f - -o json domain.com
```

...for an advanced pattern, use custom resolvers and pipe output into the awesome jq:

```
1 $ aiodnsbrute -r resolvers.txt -f - -o json google.com | jq '.[] |
  select(.ip[] | startswith("172."))'
```

Wildcard detection enabled by default (`-no-wildcard` turns it off):

```
1 $ aiodnsbrute foo.com
2
3 [*] Brute forcing foo.com with a maximum of 512 concurrent tasks...
4 [*] Using recursive DNS with the following servers: ['50.116.53.5', '
50.116.58.5', '50.116.61.5']
5 [!] Wildcard response detected, ignoring answers containing ['
23.23.86.44']
6 [*] Wordlist loaded, proceeding with 1000 DNS requests
7 [+] www.foo.com 52.73.176.251, 52.4.225.20
8 100%| 1000/1000 [00:05<00:00, 140.18records/s]
9 [*] Completed, 1 subdomains found
```

NEW use `gethostbyname` (detects CNAMEs which can be handy for potential subdomain takeover detection)

```
1 $ aiodnsbrute --gethostbyname domain.com
```

Supply a list of resolvers from file (ignoring blank lines and starting with #), specify `-r -` to read list from stdin.

```
1 $ aiodnsbrute -r resolvers.txt domain.com
```

Thanks

- Wordlists are from bitquark's `dnspop` repo (except the 10 mil entry one which I created using his tool).

-
- Creds to Sublist3r for pointing me there.
 - Click for making CLI apps so easy.
 - tqdm powers the pretty progress bar!
 - aiodns for providing the Python async interface to pycares which makes this all possible!

Notes

- You might want to do a `ulimit -n` to see how many open files are allowed. You can also increase that number using the same command, i.e. `ulimit -n <2048>`