projectdiscovery / **httpx** Public

Code   Issues 25

5 Branches   52 Tags   Go to file   Go to file   Code

ehsandeep Merge branch 'dev' of https://github.com/projectdiscovery/httpx into dev ✓

842e59e · 4 days ago

.github	adding tests	2 weeks ago
cmd	change `-tech-detect` flag var ty…	last month
common	feat: added enhancements to fa…	4 days ago
examples	adding tests	2 weeks ago
integration_tests	Updating GH workflows + Sonar …	2 years ago
internal/testutils	Update integration.go (#1319)	10 months ago
runner	version update	4 days ago
scripts	Create asn2cidr	4 years ago
static	added url format to host result (…	7 months ago
.gitignore	Extract `body_domains` and `body…`	last week
.goreleaser.yml	Workflow update (#1290)	last year
Dockerfile	Merge pull request #1497 from …	6 months ago
LICENSE.md	Update LICENSE.md	3 years ago
Makefile	Disable static compilation for os…	2 years ago
README.md	Added the Functionality of stori…	2 weeks ago
go.mod	chore(deps): bump github.com/…	4 days ago
go.sum	chore(deps): bump github.com/…	4 days ago
snapcraft.yaml	snapcraft config update	2 years ago

About

httpx is a fast and multi-purpose HTTP toolkit that allows running multiple probes using the retryablehttp library.

docs.projectdiscovery.io/tools/httpx

#cli #http #osint #pipeline #lib #cybersecurity #ssl-certificate #bugbounty #hacktoberfest #pentest-tool

Readme
MIT license
Code of conduct
Security policy
Activity
Custom properties
7.1k stars
80 watching
783 forks
Report repository

Releases 52

v1.6.5 Latest
4 days ago

+ 51 releases

Packages

No packages published

Used by 143

+ 135

Contributors 74

+ 60 contributors

Languages

● **Go** 51.8%   ● **HTML** 47.8%
● **Other** 0.4%

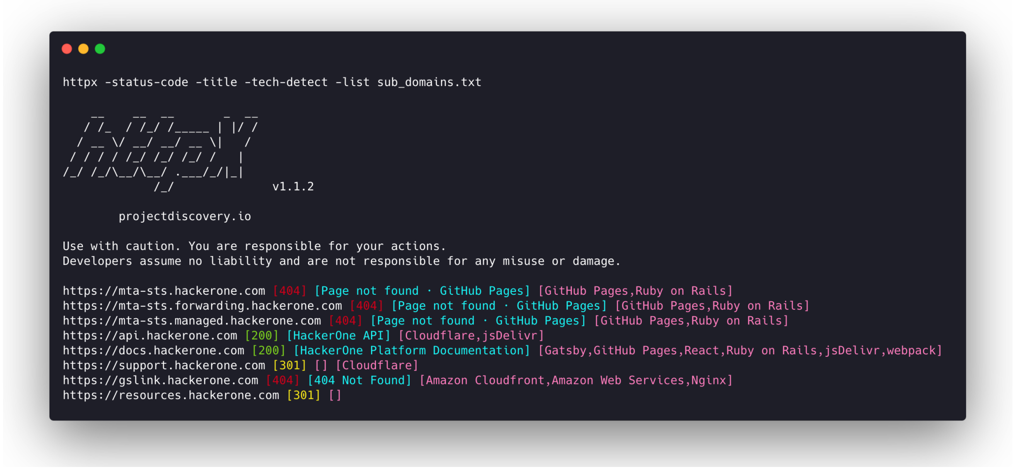
httpX

license MIT   go report A+   release v1.6.5   docker pulls 30k   Follow @pdiscoveryio   chat 8

Features • Installation • Usage • Documentation • Notes • Join Discord

`httpx` is a fast and multi-purpose HTTP toolkit that allows running multiple probes using the retryablehttp library. It is designed to maintain result reliability with an increased number of threads.

Features



```
httpx -status-code -title -tech-detect -list sub_domains.txt


    __    __  __       _  __
   / /_  / /_/ /_____ | |/ /
  / __ \/ __/ __/ __ \|   /
 / / / / /_/ /_/ /_/ /   |
/_/ /_/\__/\__/ .___/_/|_|
             /_/              v1.1.2

        projectdiscovery.io

Use with caution. You are responsible for your actions.
Developers assume no liability and are not responsible for any misuse or damage.

https://mta-sts.hackerone.com [404] [Page not found · GitHub Pages] [GitHub Pages,Ruby on Rails]
https://mta-sts.forwarding.hackerone.com [404] [Page not found · GitHub Pages] [GitHub Pages,Ruby on Rails]
https://mta-sts.managed.hackerone.com [404] [Page not found · GitHub Pages] [GitHub Pages,Ruby on Rails]
https://api.hackerone.com [200] [HackerOne API] [Cloudflare,jsDelivr]
https://docs.hackerone.com [200] [HackerOne Platform Documentation] [Gatsby,GitHub Pages,React,Ruby on Rails,jsDelivr,webpack]
https://support.hackerone.com [301] [] [Cloudflare]
https://gslink.hackerone.com [404] [404 Not Found] [Amazon Cloudfront,Amazon Web Services,Nginx]
https://resources.hackerone.com [301] []
```

- Simple and modular code base making it easy to contribute.
- Fast And fully configurable flags to probe multiple elements.
- Supports multiple HTTP based probings.
- Smart auto fallback from https to http as default.
- Supports hosts, URLs and CIDR as input.
- Handles edge cases doing retries, backoffs etc for handling WAFs.

Supported probes

Probes	Default check	Probes	Default check
URL	true	IP	true
Title	true	CNAME	true
Status Code	true	Raw HTTP	false
Content Length	true	HTTP2	false
TLS Certificate	true	HTTP Pipeline	false
CSP Header	true	Virtual host	false
Line Count	true	Word Count	true
Location Header	true	CDN	false
Web Server	true	Paths	false
Web Socket	true	Ports	false
Response Time	true	Request Method	true
Favicon Hash	false	Probe Status	false
Body Hash	true	Header Hash	true
Redirect chain	false	URL Scheme	true
JARM Hash	false	ASN	false

Installation Instructions

`httpx` requires **go1.21** to install successfully. Run the following command to get the repo:

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

To learn more about installing httpx, see https://docs.projectdiscovery.io/tools/httpx/install.

❗ Disclaimer
This project is in active development. Expect breaking changes with releases. Review the changelog before updating.

❗ **Disclaimer**

This project was primarily built to be used as a standalone CLI tool. **Running it as a service may pose security risks.** It's recommended to use with caution and additional security measures.

Usage

```
httpx -h
```

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

```
Usage:
  ./httpx [flags]

Flags:
INPUT:
   -l, -list string      input file containing list of hosts to process
   -rr, -request string  file containing raw request
   -u, -target string[]  input target host(s) to probe

PROBES:
   -sc, -status-code      display response status-code
   -cl, -content-length   display response content-length
   -ct, -content-type     display response content-type
   -location              display response redirect location
   -favicon               display mmh3 hash for '/favicon.ico' file
   -hash string           display response body hash (supported: md5,mmh3,simhash,sha1,sha256,sha512)
   -jarm                  display jarm fingerprint hash
   -rt, -response-time    display response time
   -lc, -line-count       display response body line count
   -wc, -word-count       display response body word count
   -title                 display page title
   -bp, -body-preview     display first N characters of response body (default 100)
   -server, -web-server   display server name
   -td, -tech-detect      display technology in use based on wappalyzer dataset
   -method                display http request method
   -websocket             display server using websocket
   -ip                    display host ip
   -cname                 display host cname
   -extract-fqdn, -efqdn  get domain and subdomains from response body and header in jsonl/csv output
   -asn                   display host asn information
   -cdn                   display cdn/waf in use (default true)
   -probe                 display probe status

HEADLESS:
   -ss, -screenshot                 enable saving screenshot of the page using headless browser
   -system-chrome                   enable using local installed chrome for screenshot
   -ho, -headless-options string[]  start headless chrome with additional options
   -esb, -exclude-screenshot-bytes  enable excluding screenshot bytes from json output
   -ehb, -exclude-headless-body     enable excluding headless header from json output
   -st, -screenshot-timeout int     set timeout for screenshot in seconds (default 10)

MATCHERS:
   -mc, -match-code string            match response with specified status code (-mc 200,302)
   -ml, -match-length string          match response with specified content length (-ml 100,102)
   -mlc, -match-line-count string     match response body with specified line count (-mlc 423,532)
   -mwc, -match-word-count string     match response body with specified word count (-mwc 43,55)
   -mfc, -match-favicon string[]      match response with specified favicon hash (-mfc 1494302000)
   -ms, -match-string string[]        match response with specified string (-ms admin)
   -mr, -match-regex string[]         match response with specified regex (-mr admin)
   -mcdn, -match-cdn string[]         match host with specified cdn provider (leaseweb, stackpath, cloudfront, fastly
   -mrt, -match-response-time string  match response with specified response time in seconds (-mrt '< 1')
   -mdc, -match-condition string      match response with dsl expression condition

EXTRACTOR:
   -er, -extract-regex string[]   display response content with matched regex
   -ep, -extract-preset string[]  display response content matched by a pre-defined regex (url,ipv4,mail)

FILTERS:
   -fc, -filter-code string            filter response with specified status code (-fc 403,401)
   -fep, -filter-error-page            filter response with ML based error page detection
   -fl, -filter-length string          filter response with specified content length (-fl 23,33)
   -flc, -filter-line-count string     filter response body with specified line count (-flc 423,532)
   -fwc, -filter-word-count string     filter response body with specified word count (-fwc 423,532)
   -ffc, -filter-favicon string[]      filter response with specified favicon hash (-ffc 1494302000)
   -fs, -filter-string string[]        filter response with specified string (-fs admin)
   -fe, -filter-regex string[]         filter response with specified regex (-fe admin)
   -fcdn, -filter-cdn string[]         filter host with specified cdn provider (leaseweb, stackpath, cloudfront, fast
   -frt, -filter-response-time string  filter response with specified response time in seconds (-frt '> 1')
   -fdc, -filter-condition string      filter response with dsl expression condition
   -strip                              strips all tags in response. supported formats: html,xml (default html)

RATE-LIMIT:
   -t, -threads int              number of threads to use (default 50)
   -rl, -rate-limit int          maximum requests to send per second (default 150)
   -rlm, -rate-limit-minute int  maximum number of requests to send per minute

MISCELLANEOUS:
   -pa, -probe-all-ips        probe all the ips associated with same host
   -p, -ports string[]        ports to probe (nmap syntax: eg http:1,2-10,11,https:80)
   -path string               path or list of paths to probe (comma-separated, file)
   -tls-probe                 send http probes on the extracted TLS domains (dns_name)
   -csp-probe                 send http probes on the extracted CSP domains
   -tls-grab                  perform TLS(SSL) data grabbing
   -pipeline                  probe and display server supporting HTTP1.1 pipeline
   -http2                     probe and display server supporting HTTP2
   -vhost                     probe and display server supporting VHOST
   -ldv, -list-dsl-variables  list json output field keys name that support dsl matcher/filter

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

OUTPUT:
   -o, -output string                  file to write output results
   -oa, -output-all                    filename to write output results in all formats
   -sr, -store-response                store http response to output directory
   -srd, -store-response-dir string    store http response to custom directory
   -ob, -omit-body                     omit response body in output
   -csv                                store output in csv format
   -csvo, -csv-output-encoding string  define output encoding
   -j, -json                           store output in JSONL(ines) format
   -irh, -include-response-header      include http response (headers) in JSON output (-json only)
   -irr, -include-response             include http request/response (headers + body) in JSON output (-json only)
   -irrb, -include-response-base64     include base64 encoded http request/response in JSON output (-json only)
   -include-chain                      include redirect http chain in JSON output (-json only)
   -store-chain                        include http redirect chain in responses (-sr only)
   -svrc, -store-vision-recon-cluster  include visual recon clusters (-ss and -sr only)
   -pr, -protocol string               protocol to use (unknown, http11)

CONFIGURATIONS:
   -config string                  path to the httpx configuration file (default $HOME/.config/httpx/config.yaml)
   -auth                           configure projectdiscovery cloud (pdcp) api key (default true)
   -r, -resolvers string[]         list of custom resolver (file or comma separated)
   -allow string[]                 allowed list of IP/CIDR's to process (file or comma separated)
   -deny string[]                  denied list of IP/CIDR's to process (file or comma separated)
   -sni, -sni-name string          custom TLS SNI name
   -random-agent                   enable Random User-Agent to use (default true)
   -H, -header string[]            custom http headers to send with request
   -http-proxy, -proxy string      http proxy to use (eg http://127.0.0.1:8080)
   -unsafe                         send raw requests skipping golang normalization
   -resume                         resume scan using resume.cfg
   -fr, -follow-redirects          follow http redirects
   -maxr, -max-redirects int       max number of redirects to follow per host (default 10)
   -fhr, -follow-host-redirects    follow redirects on the same host
   -rhsts, -respect-hsts           respect HSTS response headers for redirect requests
   -vhost-input                    get a list of vhosts as input
   -x string                       request methods to probe, use 'all' to probe all HTTP methods
   -body string                    post body to include in http request
   -s, -stream                     stream mode - start elaborating input targets without sorting
   -sd, -skip-dedupe               disable dedupe input items (only used with stream mode)
   -ldp, -leave-default-ports      leave default http/https ports in host header (eg. http://host:80 - https://host:
   -ztls                           use ztls library with autofallback to standard one for tls13
   -no-decode                      avoid decoding body
   -tlsi, -tls-impersonate         enable experimental client hello (ja3) tls randomization
   -no-stdin                       Disable Stdin processing
   -hae, -http-api-endpoint string  experimental http api endpoint

DEBUG:
   -health-check, -hc        run diagnostic check up
   -debug                    display request/response content in cli
   -debug-req                display request content in cli
   -debug-resp               display response content in cli
   -version                   display httpx version
   -stats                     display scan statistic
   -profile-mem string        optional httpx memory profile dump file
   -silent                    silent mode
   -v, -verbose               verbose mode
   -si, -stats-interval int   number of seconds to wait between showing a statistics update (default: 5)
   -nc, -no-color             disable colors in cli output

OPTIMIZATIONS:
   -nf, -no-fallback                  display both probed protocol (HTTPS and HTTP)
   -nfs, -no-fallback-scheme          probe with protocol scheme specified in input
   -maxhr, -max-host-error int        max error count per host before skipping remaining path/s (default 30)
   -e, -exclude string[]              exclude host matching specified filter ('cdn', 'private-ips', cidr, ip, regex)
   -retries int                       number of retries
   -timeout int                       timeout in seconds (default 10)
   -delay value                       duration between each http request (eg: 200ms, 1s) (default -1ns)
   -rsts, -response-size-to-save int  max response size to save in bytes (default 2147483647)
   -rstr, -response-size-to-read int  max response size to read in bytes (default 2147483647)
```

Running httpx

For details about running httpx, see https://docs.projectdiscovery.io/tools/httpx/running.

Using `httpx` as a library

`httpx` can be used as a library by creating an instance of the `Option` struct and populating it with the same options that would be specified via CLI. Once validated, the struct should be passed to a runner instance (to be closed at the end of the program) and the `RunEnumeration` method should be called. A minimal example of how to do it is in the examples folder

Notes

- As default, `httpx` probe with **HTTPS** scheme and fall-back to **HTTP** only if **HTTPS** is not reachable.
- The `-no-fallback` flag can be used to probe and display both **HTTP** and **HTTPS** result.
- Custom scheme for ports can be defined, for example `-ports http:443,http:80,https:8443`
- Custom resolver supports multiple protocol (**doh**|**tcp**|**udp**) in form of `protocol:resolver:port` (e.g. `udp:127.0.0.1:53`)
- The following flags should be used for specific use cases instead of running them as default with other probes:
  - `-ports`
  - `-path`
  - `-vhost`
  - `-screenshot`
  - `-csp-probe`
  - `-tls-probe`
  - `-favicon`
  - `-http2`
  - `-pipeline`
  - `-tls-impersonate`

Acknowledgement