A Practical Guide to `httpx`

Eyad Islam El-Taher

October 19, 2025

Introduction

`httpx` is a fast and versatile HTTP probing and reconnaissance toolkit developed in Go by the team behind ProjectDiscovery. It enables users (particularly in web-security, bug-bounty and infrastructure analysis contexts) to efficiently scan, probe and gather information about web services (hosts, ports, paths, responses) via HTTP(S).

Installation

According to the detailed guide:

- Ensure Go (golang) version 1.20 (or compatible) is installed.
- Then install via:
  ```
  go install -v github.com/projectdiscovery/httpx/cmd/httpx@latest
  ```
- After installation go to your Golang Path, /go/bin/ (in my case)
- copy httpx to directory /usr/bin/ ⇒ sudo cp httpx /usr/bin/
- Confirm installation by running:
  ```
  httpx --help
  ```

Core Features and Use-Cases

Scanning / Probing Hosts

`httpx` can probe individual hosts or many hosts via file/input. Example:

```
httpx -u example.com -probe
```

This checks the availability of the host via HTTP. You can also supply a file of hosts:

```
httpx -l hosts.txt
```

Which reads hosts from `hosts.txt` each on its own line. And you can pipe input from other tools (for example from a sub-domain enumeration tool) into httpx:

```
cat hosts.txt | grep example.com | httpx
```

For multiple ports:

```
httpx -u example.com -ports 80,443,8009,8080,8081,8090,8180,9443
```

This is useful because web services may run on non-standard HTTP ports.

Probes – Gathering Response Details

You can ask httpx to collect various attributes about each HTTP response: status code, content type/length, title, server software, technology detection, hash of response body, response time, etc. Example:

```
httpx -status-code -content-type -content-length -location \
    -title -web-server -tech-detect -ip -cname -word-count \
    -line-count -response-time -cdn -hash sha256 \
    -include-response -silent -stats \
    -follow-host-redirects -max-redirects 2
```

Such detailed probing is useful for infrastructure mapping, fingerprinting, and reconnaissance.

Filtering, Matching, Extraction

`httpx` supports matchers (to include based on criteria) and filters (to exclude undesired results). Examples:

- Match specific HTTP codes:

  ```
  cat hosts.txt | httpx -mc 200,302
  ```

- Match responses containing a specific string:

  ```
  cat hosts.txt | httpx -ms admin
  ```

- Filter out unwanted status codes:

  ```
  httpx -l urls.txt -fc 404,403,401,400,500
  ```

- Extract parts of a response via regex:

  ```
  echo "http://example.com" | httpx -er 'admin*'
  ```

Performance / Rate-Limiting / Threading

When scanning many targets, you'll want to tune performance: number of threads, rate limits, timeouts, retries. Example:

```
httpx -u example.com -t 10 -rate-limit 50
```

This sets 10 threads and limits to 50 requests per second. Other options: `-rl` (requests per second), `-rlm` (per minute), `-timeout`, `-retries`, etc.

Output Options

You can save results in different formats for later processing:

- Save to a plain file:

  ```
  httpx -l urls.txt -o httpx.log
  ```

- Output as JSON Lines:

  ```
  httpx -l urls.txt -j
  ```

- Store full HTTP responses in a directory:

  ```
  httpx -l urls.txt -sr http-responses/
  ```

Advanced / Additional Features

Other notable capabilities:

- Screenshots: headless browser captures of the target web pages. Example:

  ```
  echo https://example.com | httpx -ss -st
  ```

- Configuration via YAML file: Instead of specifying many flags, you can maintain a config file, e.g., `httpx-config.yaml`.
- HTTP methods probe (which methods are allowed):

  ```
  echo "http://example.com" | httpx -x all -probe
  ```

- Proxy support (HTTP, SOCKS):

  ```
  echo "http://example.com" | httpx -http-proxy http://127.0.0.1:8080
  ```

Usage Examples

- Simple host probe:

  ```
  httpx -u example.com -probe
  ```

- Batch scan with title and status code:

  ```
  httpx -l subdomains.txt -title -status-code -tech-detect
  ```

- Scan for a specific path (e.g., robots.txt) across many hosts:

  ```
  httpx -l hosts.txt -path "/robots.txt" -sc
  ```

- Match only HTTP 200/301/302 responses:

  ```
  cat subdomains.txt | httpx -mc 200,301,302 -sc
  ```

- Save output to JSON and process with jq:

  ```
  httpx -l urls.txt -j -o httpx.json
  cat httpx.json | jq 'select(.status_code == 200)'
  ```