

fuzzing with



\$ whoami

- Head of Security Testing @ Visma
- Open source: ffuf, certbot, acme-dns, confused, certainly ...
- <3 Board games and craft beer <3
- Disobey puzzle team lead, TurkuSec & various global infosec communities



@joohoi on the intertubes

fuzzing with ffuf

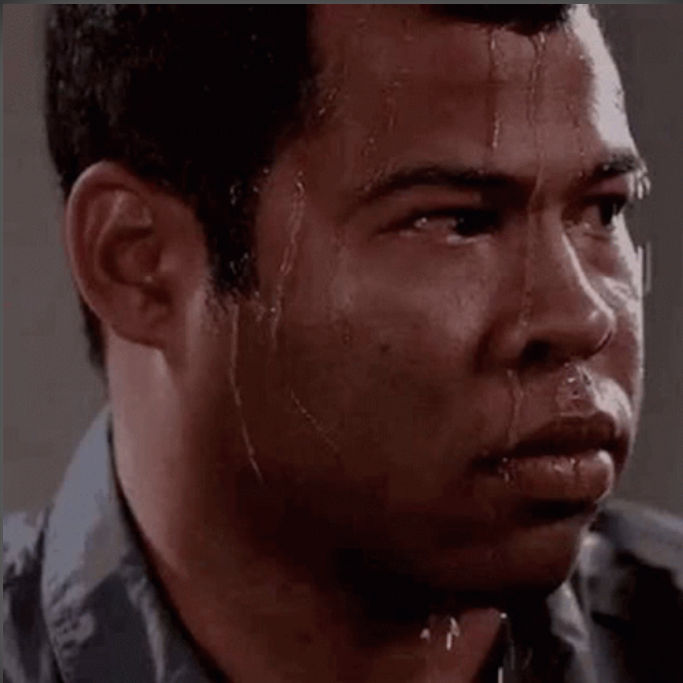
What is this ffuf thing anyway?

- **Swiss army-knife HTTP bruteforce tool**
- **Sending in a lot of requests and detecting anomalies**
- **Target any part of HTTP request**
- **Anomaly detection for various different aspects:**
 - **Status codes**
 - **Response size (bytes / words / lines)**
 - **Request-response roundtrip time**
 - **Regex matching**
- **Very customizable**
- **Focusing around flexibility**



fuzzing with ffuf

\$ ffuf -h



Fuzz Faster U Fool - v2.1.0-dev

```
HTTP OPTIONS:
-H Header "Name: Value", separated by colon. Multiple -H flags are accepted.
-X HTTP method to use.
-s Cookie data "NAME=VALUE1; NAME2=VALUE2" for copy as curl functionality.
-cc Client cert for authentication. Client key needs to be defined as well for this to work.
-cK Client key for authentication. Client certificate needs to be defined as well for this to work.
-d POST data.
-http2 Use HTTP2 protocol (default: false)
-ignore-body Do not fetch the response content. (default: false)
-r Follow redirects (default: false)
-raw Do not encode URI (default: false)
-Scan Scan recursively. Only FUZZ keyword is supported, and URL (-u) has to end in it. (default: false)
-recursion Maximum recursion depth. (default: 0)
-recursion-depth Recursion strategy: "default" for a redirect based, and "greedy" to recurse on all matches (default: default)
-replay-proxy Replay matched requests using this proxy.
-sni Target TLS SNI, does not support FUZZ keyword
-timeout HTTP request timeout in seconds. (default: 10)
-u Target URL
-x Proxy URL (SOCKS5 or HTTP). For example: http://127.0.0.1:8080 or socks5://127.0.0.1:8080

GENERAL OPTIONS:
-V Show version information. (default: false)
-ac Automatically calibrate filtering options (default: false)
-acc Custom auto-calibration string. Can be used multiple times. Implies -ac
-ach Per host autocalibration (default: false)
-ack Autocalibration keyword (default: FUZZ)
-acs Custom auto-calibration strategies. Can be used multiple times. Implies -ac
-c Colorize output. (default: true)
-config Load configuration from a file
-json JSON output, printing newline-delimited JSON records (default: false)
-maxtime Maximum running time in seconds for entire process. (default: 0)
-maxtime-job Maximum running time in seconds per job. (default: 0)
-noninteractive Disable the interactive console functionality (default: false)
-p Seconds of delay between requests, or a range of random delay. For example "0.1" or "0.1-2.0"
-rate Rate of requests per second (default: 8)
-s Do not print additional information (silent mode) (default: false)
-sa Stop on all error cases. Implies -sf and -se. (default: false)
-scraperfile Custom scraper file path
-scrapers Active scraper groups (default: all)
-se Stop on spurious errors (default: false)
-search Search for a HTTPASH payload from first history
-sf Stop when > 95% of responses return 403 Forbidden (default: false)
-t Number of concurrent threads. (default: 40)
-v Verbose output, printing full URL and redirect location (if any) with the results. (default: false)

MATCHER OPTIONS:
-mc Match HTTP status codes, or "all" for everything. (default: 200-299,301,302,307,401,403,405,500)
-ml Match amount of lines in response
-mde Match set operator. Either or: and, or (default: or)
-mr Match regexp
-ms Match HTTP response size
-mt Match how many milliseconds to the first response byte, either greater or less than. EG: >100 or <100
-mw Match amount of words in response

FILTER OPTIONS:
-fc Filter HTTP status codes from response. Comma separated list of codes and ranges
-fl Filter by amount of lines in response. Comma separated list of line counts and ranges
-fnde Filter set operator. Either or: and, or (default: or)
-fr Filter regexp
-fs Filter HTTP response size. Comma separated list of sizes and ranges
-ft Filter by number of milliseconds to the first response byte, either greater or less than. EG: >100 or <100
-fw Filter by amount of words in response. Comma separated list of word counts and ranges

INPUT OPTIONS:
-d DirSearch wordlist compatibility mode. Used in conjunction with -e flag. (default: false)
-e Comma separated list of extensions. Extends FUZZ keywords.
-enc Encoders for keywords, eg. "FUZZ:urlencode;b64encode"
-ic Ignore wordlist comments (default: false)
-input-cmd Command producing the input. --input-num is required when using this input method. Overrides -w.
-input-num Number of inputs to test. Used in conjunction with --input-cmd. (default: 100)
-input-shell Shell to be used for running command
-mode Multisearch operation mode. Available modes: clusterbomb, pitchfork, sniper (default: clusterbomb)
-request File containing the raw http request
-request-proto Protocol to use along with raw request (default: https)
-w Wordlist file path and (optional) keyword separated by colon. eg. '/path/to/wordlist:KEYWORD'

OUTPUT OPTIONS:
-debug-log Write all of the internal logging to the specified file.
-o Write output to file
-od Directory path to store matched results to.
-of Output file format. Available formats: json, ejson, html, md, csv, csvs (or, 'all' for all formats) (default: json)
-or Don't create the output file if we don't have results (default: false)

EXAMPLE USAGE:
Fuzz file paths from wordlist.txt, match all responses but filter out those with content-size 42.
Colored, verbose output.
ffuf -w wordlist.txt -u https://example.org/FUZZ -mc all -fs 42 -c -v

Fuzz Host-header, match HTTP 200 responses.
ffuf -w hosts.txt -u https://example.org/ -H "Host: FUZZ" -mc 200

Fuzz POST JSON data. Match all responses not containing text "error".
ffuf -w entries.txt -u https://example.org/ -X POST -H "Content-Type: application/json" \
-d '{"name": "FUZZ", "anotherkey": "anothervalue"}' -fr "error"

Fuzz multiple locations. Match only responses reflecting the value of "PARAM" keyword. Colored.
ffuf -w params.txt:PARAM -w values.txt:VAL -u https://example.org/IPARAM=VAL -ar 'VAL' -c

More information and examples: https://github.com/ffuf/ffuf
```



ffuf -h



github.com/ffuf/ffuf/wiki/CLI-flags

fuzzing with ffuf

Wenger

Wenger 16999 Swiss Army Knife Giant



673 customer reviews

| 146 answered questions

Note: This item is only available from third-party sellers
(see all offers).



Surprising Results

By a fan on December 31, 2013

I tried to file my nails, but in the process I accidentally fixed a small engine that was near by. Which was nice



fuzzing with ffuf

Design philosophy

- **Enable - do not limit**
- **Can be done outside of ffuf? Won't implement it**
- **Avoid magic - stay idiomatic and understandable**
- **Maintain UX integrity across the board**
- **If there's a established standard - do not reinvent**



fuzzing with ffuf

Basics - wordlists

```
→ ~ ffuf -w content.txt:CONTENT -w paths.txt:PATH -u https://target.tld/PATH/CONTENT
```

- **Important - do not skip**
- **Can be a file, or piped through STDIN (-w -)**
- **Best: technology specific, but aware of smoke and mirrors tactics**
- **Resources available: [SecLists](#), [Assetnote](#), [PayloadsAllTheThings](#)**
- **Often beneficial to create a custom one!**
 - **Tooling available: [CEWL](#), [Katana](#)**
 - **Clankers**
- **Ffuf supports multi-wordlists to extend your capabilities**

fuzzing with ffuf

Basics - matching and filtering

- **Best filter is always the most specific one**
- **HTTP Status code -mc / -fc**
 - For example match 200 - OK responses
- **Number of lines -ml / -fl**
 - Good to use when content is dynamic
- **Number of words -mw / -fw**
 - If output includes the input value
- **Response content length -ms / -fs**
 - If responses are static
- **Roundtrip time -mt / -ft**
 - Slow endpoints, SQL injections
- **Automatic calibration available: -ac**



fuzzing with ffuf

Basics - keywords

- Each wordlist defined in the command line has its own keyword
 - For example `-w wordlist.txt:KEYWORD`
 - Defaults to `FUZZ` if not explicitly defined
- Points out the place(s) in request to substitute with wordlist entries
- Can be used in various places:
 - URL (`-u https://target.tld/FUZZ`)
 - Headers (`-H "Host: FUZZ.target.tld"`)
 - Data (`-d "{ 'key': 'FUZZ' }"`)

fuzzing with ffuf

Basics - complex requests

- **Best practice is to save the raw request to a text file and edit it accordingly**
 - **Adding keywords where needed**
 - **Removing unnecessary data**
- **Start ffuf with -request raw_request.txt**
 - **Note that the schema is not defined in raw request**
 - **Default: HTTPS, if target is HTTP use -request-proto http**
- **Configure rest of the options on command line as usual**

fuzzing with ffuf

Basics - rate limiting

- **Play nice! Don't bring down the target**
- **What is reasonable?**
 - ...depends
- **ffuf has different ways of rate limiting, but just use -rate if unsure.**
- **Too high fuzzing rate can result into resource exhaustion in the target in various ways**



fuzzing with ffuf

Intermediate - interactive

- **Hit ENTER during the run**
- **(Re)Configure or fine-tune your filters & matchers**
- **Apply filtering on existing results and show the output (note: irreversible)**
- **Save current results to a file**
- **Skip a recursion job**
- **Restart a run with newly applied configuration options**

```
:: Progress: [3/4658] :: Job [1/1] :: 22 req/sec :: Duration: [0:00:01] :: Errors: 0 ::  
[INFO] ----- PAUSING -----  
  
entering interactive mode  
type "help" for a list of commands, or ENTER to resume.  
> help  
  
available commands:  
afc [value]      - append to status code filter  
fc [value]       - (re)configure status code filter  
afl [value]      - append to line count filter  
fl [value]       - (re)configure line count filter  
afw [value]      - append to word count filter  
fw [value]       - (re)configure word count filter  
afs [value]      - append to size filter  
fs [value]       - (re)configure size filter  
aft [value]      - append to time filter  
ft [value]       - (re)configure time filter  
rate [value]     - adjust rate of requests per second (active: 2)  
queueshow        - show job queue  
queuedel [number] - delete a job in the queue  
queueskip        - advance to the next queued job  
restart          - restart and resume the current ffuf job  
resume           - resume current ffuf job (or: ENTER)  
show             - show results for the current job  
savejson [filename] - save current matches to a file  
help            - you are looking at it  
> |
```

fuzzing with ffuf

Intermediate - payload encoder

```
→ ~ ffuf -w wordlist.txt -u https://example.org/FUZZ -enc "FUZZ:urlencode b64encode"
```

Usage

pencode - complex payload encoder v0.3

Usage: ./pencode FUNC1 FUNC2 FUNC3...

./pencode reads input from stdin by default, which is typically piped from another process.

OPTIONS:

-input reads input from a file, line by line.

ENCODERS

b64encode	- Base64 encoder
hexencode	- Hex string encoder
jsonescape	- JSON escape
unicodeencodeall	- Unicode escape string encode (all characters)
urlencode	- URL encode reserved characters
urlencodeall	- URL encode all characters
utf16	- UTF-16 encoder (Little Endian)
utf16be	- UTF-16 encoder (Big Endian)
xmlescape	- XML escape



Standalone: <https://github.com/ffuf/pencode>

fuzzing with ffuf

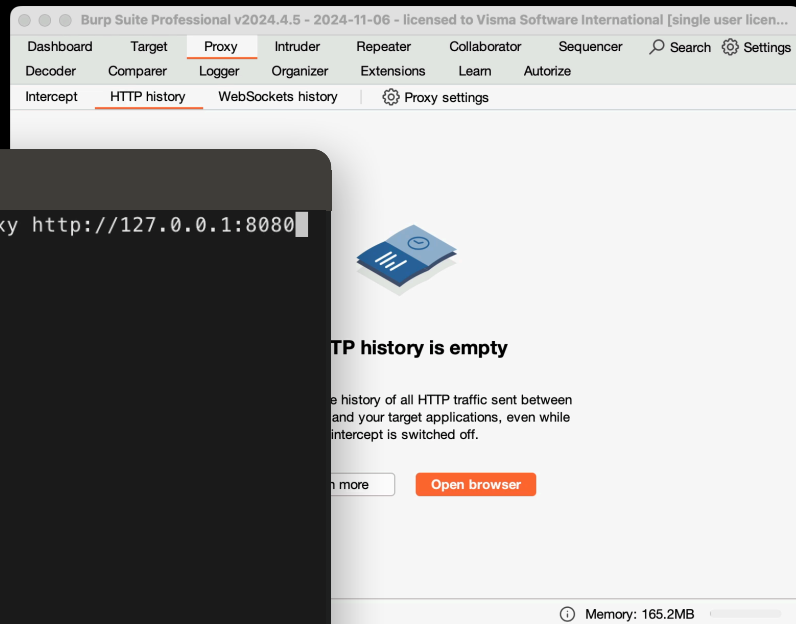
Intermediate - observe & capture

- **Save matched raw request & response data: -od output_dir_name**
- **Send matches to a proxy: -replay-proxy http://127.0.0.1:8080**
 - **Advanced: ssh reverse tunnel (-R) to local burp from ffuf running in VPS**
- **Show transport layer issues:**
-debug-log logname.log



- - replay-proxy

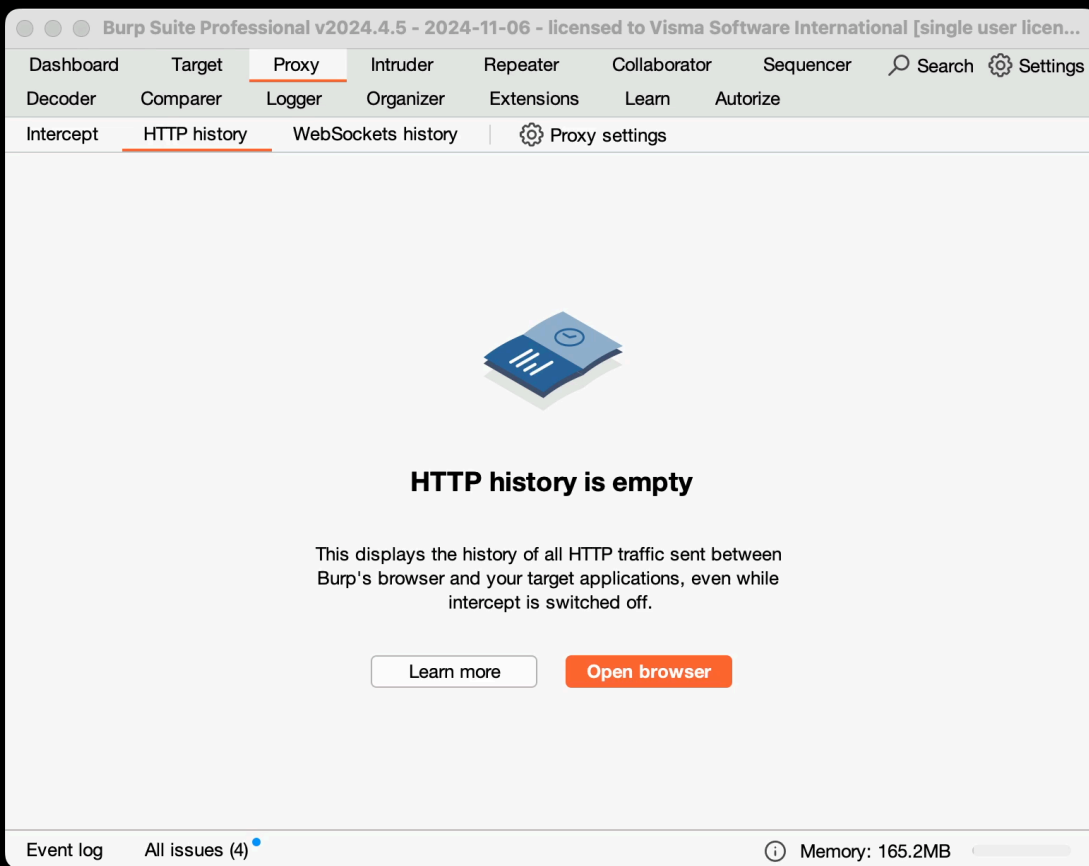
```
> tools ffuf -w common.txt -u https://ffuf.io.fi/FUZZ -rate 200 --replay-proxy http://127.0.0.1:8080
```



:: Progress: [39/39] :: Job [1/1] :: 25 req/sec :: Duration: [0:00:02] :: Errors: 0 ::

- - replay-proxy

```
tools ffuf -w common.txt -u https://ffuf.io.fi/FUZZ -rate 200 --replay-proxy http://127.0.0.1:8080
```



:: Progress: [39/39] :: Job [1/1] :: 25 req/sec :: Duration: [0:00:02] :: Errors: 0 ::

fuzzing with ffuf

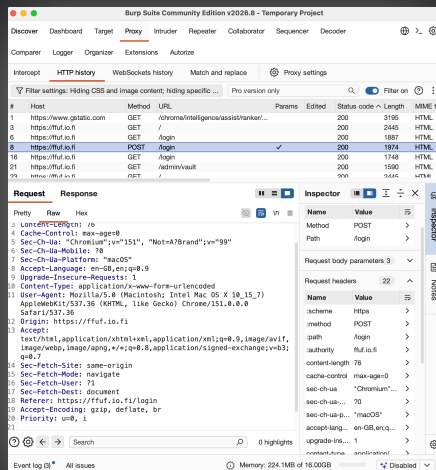
Advanced - the request you cannot replay



- Login forms hand out a single-use CSRF token
- Point ffuf at the form directly: every request is 403
- Session is per-request, wordlists are not
- Old fix: write a Python script, stop fuzzing
- Other old fix: a proxy in front of ffuf
- Same problem for nonces, one-time IDs, short-lived bearer tokens

fuzzing with ffuf

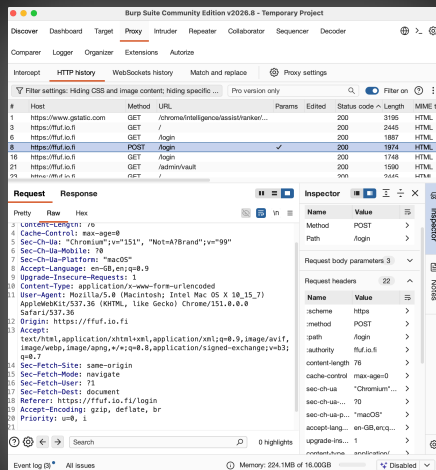
Advanced - preflight and postflight



- **-preflight <file>: raw request sent before each fuzzing request**
- **-postflight <file>: raw request sent after each fuzzing request**
- **Files are Burp-style raw HTTP, the same format as -request**

fuzzing with ffuf

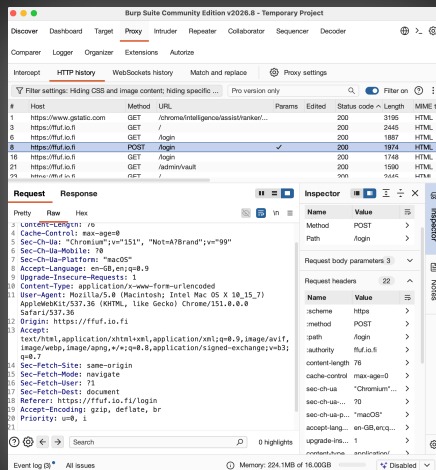
Advanced - preflight and postflight



- **-preflight-var "NAME:regex"**: capture group from the preflight response body
- **NAME** substitutes into URL, headers and body, exactly like FUZZ
- **A -preflight-var binds to the -preflight before it**
- **Chain them: multiple -preflight run in order, later ones see earlier vars**
- **Flight requests inherit main config headers, -H auth and -b cookies included**

fuzzing with ffuf

Advanced - preflight gotchas



- Extraction reads the response BODY only
- -preflight-mode per-request is the safe default, one extra request every time
- -preflight-mode per-thread runs once per thread and reuses, cheap but only for reusable values
- Single-use token plus per-thread equals silent garbage results
- Captured values cannot change the scheme, host or port of a flight request
- Traffic counts against the shared rate (-rate and -p)

Thanks!

Slides: https://io.fi/fuzzing_with_ffuf.pdf

<https://github.com/ffuf/ffuf>

<https://github.com/ffuf/ffuf/wiki>

<https://github.com/ffuf/pencode>

<https://ffuf.io.fi/play>

