Wfuzz Documentation

Release 2.1.4

Xavi Mendez

Nov 06, 2020

Contents

Wfuzz provides a framework to automate web applications security assessments and could help you to secure your web applications by finding and exploiting web application vulnerabilities.

CHAPTER 1

See Wfuzz in action

- Wfuzz cli:

```
$ wfuzz -w wordlist/general/common.txt --hc 404 http://testphp.vulnweb.com/FUZZ
********************************************************
* Wfuzz 2.2 - The Web Bruteforcer                      *
********************************************************

Target: http://testphp.vulnweb.com/FUZZ
Total requests: 950

==================================================================
ID      Response   Lines      Word         Chars          Request
==================================================================

00022:  C=301      7 L        12 W          184 Ch        "admin"
00130:  C=403     10 L        29 W          263 Ch        "cgi-bin"
00378:  C=301      7 L        12 W          184 Ch        "images"
00690:  C=301      7 L        12 W          184 Ch        "secured"
00938:  C=301      7 L        12 W          184 Ch        "CVS"

Total time: 5.519253
Processed Requests: 950
Filtered Requests: 945
Requests/sec.: 172.1247
```

- Wfuzz library:

```
>>> import wfuzz
>>> for r in wfuzz.get_payload(range(100)).fuzz(hl=[97], url="http://testphp.
↪vulnweb.com/listproducts.php?cat=FUZZ"):
...     print r
...
00125:  C=200    102 L       434 W         7011 Ch        "1"
00126:  C=200     99 L       302 W         4442 Ch        "2"
```

other tools included in the wfuzz framework.

- Wfuzz payload generator:

```
$ wfpayload -z range,0-10
0
1
2
3
4
5
6
7
8
9
10
```

- Wfuzz encoder/decoder:

```
$ wfencode -e md5 test
098f6bcd4621d373cade4e832627b4f6
```

- You can also run wfuzz from the official docker image:

```
$ docker run -v $(pwd)/wordlist:/wordlist/ -it ghcr.io/xmendez/wfuzz wfuzz
********************************************************
* Wfuzz 3.0.3 - The Web Fuzzer                         *
*                                                      *
* Version up to 1.4c coded by:                         *
* Christian Martorella (cmartorella@edge-security.com) *
* Carlos del ojo (deepbit@gmail.com)                   *
*                                                      *
* Version 1.4d to 3.0.3 coded by:                      *
* Xavier Mendez (xmendez@edge-security.com)            *
********************************************************

Usage:  wfuzz [options] -z payload,params <url>

        FUZZ, ..., FUZnZ  wherever you put these keywords wfuzz will replace them␣
↪with the values of the specified payload.
        FUZZ{baseline_value} FUZZ will be replaced by baseline_value. It will be␣
↪the first request performed and could be used as a base for filtering.


Examples:
        wfuzz -c -z file,users.txt -z file,pass.txt --sc 200 http://www.site.com/
↪log.asp?user=FUZZ&pass=FUZ2Z
        wfuzz -c -z range,1-10 --hc=BBB http://www.site.com/FUZZ{something not␣
↪there}
        wfuzz --script=robots -z list,robots.txt http://www.webscantest.com/FUZZ

Type wfuzz -h for further information or --help for advanced usage.
```

CHAPTER 2

How it works

Wfuzz it is based on a simple concept: it replaces any reference to the FUZZ keyword by the value of a given payload.

A payload in Wfuzz is a source of data.

This simple concept allows any input to be injected in any field of an HTTP request, allowing to perform complex web security attacks in different web application components such as: parameters, authentication, forms, directories/files, headers, etc.

Wfuzz is more than a web brute forcer:

- Wfuzz's web application vulnerability scanner is supported by plugins.
- Wfuzz is a completely modular framework and makes it easy for even the newest of Python developers to contribute. Building plugins is simple and takes little more than a few minutes.
- Wfuzz exposes a simple language interface to the previous HTTP requests/responses performed using Wfuzz or other tools, such as Burp. This allows you to perform manual and semi-automatic tests with full context and understanding of your actions, without relying on a web application scanner underlying implementation.

CHAPTER 3

Installation Guide

3.1 Installation

3.1.1 Pip install Wfuzz

To install WFuzz using pip

```
$ pip install wfuzz
```

3.1.2 Use the wfuzz docker image

You can pull wfuzz docker image from github registry by executing:

```
$ docker pull ghcr.io/xmendez/wfuzz
```

3.1.3 Get the Source Code

Wfuzz is actively developed on GitHub.

You can either clone the public repository:

```
$ git clone git://github.com/xmendez/wfuzz.git
```

Or download last release.

Once you have a copy of the source, you can embed it in your own Python package, or install it into your site-packages easily:

```
$ python setup.py install
```

3.1.4 Dependencies

Wfuzz uses:

- pycurl library to perform HTTP requests.
- pyparsing library to create filter's grammars.
- JSON.miniy (C) Gerald Storer to read json recipes.
- chardet to detect dictionaries encoding.
- coloroma to support ANSI escape characters in Windows.

3.2 Installation issues

3.2.1 Pycurl on MacOS

Wfuzz uses pycurl as HTTP library. You might get errors like the listed below when running Wfuzz:

```
pycurl: libcurl link-time ssl backend (openssl) is different from compile-time ssl␣
↪backend (none/other)
```

Or:

```
pycurl: libcurl link-time ssl backend (none/other) is different from compile-time ssl␣
↪backend (openssl)
```

This is due to the fact that, MacOS might need some tweaks before pycurl is installed correctly:

1. First you need to install OpenSSL via Homebrew:

   ```
   $ brew install openssl
   ```

2. Curl is normally already installed in MacOs, but to be sure it uses OpenSSL, we need to install it using brew:

   ```
   $ brew install curl-openssl
   ```

3. Curl is installed keg-only by brew. This means that is installed but not linked. Therefore, we need to instruct pip to use the recently installed curl before installing pycurl. We can do this permanently by changing our bash_profile:

   ```
   $ echo 'export PATH="/usr/local/opt/curl-openssl/bin:$PATH"' >> ~/.bash_profile
   ```

4. Or temporary in the current shell:

   ```
   $ export PATH="/usr/local/opt/curl-openssl/bin:$PATH"
   ```

5. Then, we need to install pycurl as follows:

   ```
   $ PYCURL_SSL_LIBRARY=openssl LDFLAGS="-L/usr/local/opt/openssl/lib" CPPFLAGS="-I/
   ↪usr/local/opt/openssl/include" pip install --no-cache-dir pycurl
   ```

6. Finally, if we re-install or execute wfuzz again it should work correctly.

If you get errors such as:

```
Fatal exception: dlopen(xxx/lib/python3.7/site-packages/pycurl.cpython-37m-darwin.so,␣
↪2): Library not loaded:      /usr/local/opt/openssl/lib/libssl.1.0.0.dylib
Referenced from: /usr/local/opt/curl-openssl/lib/libcurl.4.dylib
Reason: image not found. Wfuzz needs pycurl to run. Pycurl could be installed using␣
↪the following command:
```

Run brew update && brew upgrade

If you get an error such as:

```
ImportError: pycurl: libcurl link-time ssl backends (secure-transport, openssl) do␣
↪not include compile-time ssl backend (none/other)
```

That might indicate that pycurl was reinstalled and not linked to the SSL correctly. Uninstall pycurl as follows:

```
$ pip uninstall pycurl
```

and re-install pycurl starting from step 4 above.

3.2.2 Pycurl on Windows

Install pycurl matching your python version from https://pypi.org/project/pycurl/#files

3.2.3 PyCurl SSL bug

If you experience errors when using Wfuzz against SSL sites, it could be because an old know issue:

http://bugs.debian.org/cgi-bin/bugreport.cgi?bug=515200

Briefly, pycurl is built against libcurl3-gnutls, which does not work with a number of web sites. Pycurl fails with the following error message:

```
pycurl.error: (35, 'gnutls_handshake() failed: A TLS packet with unexpected length␣
↪was received.')
```

3.2.3.1 Verifying the problem

- Pycurl linked against gnutls:

  ```
  $ python
  >>> import pycurl
  >>> pycurl.version
  libcurl/7.21.3 GnuTLS/2.8.6 zlib/1.2.3.4 libidn/1.18'
  ```

- Pycurl linked against openssl:

  ```
  $ python
  >>> import pycurl
  >>> pycurl.version
  'libcurl/7.21.3 OpenSSL/0.9.8o zlib/1.2.3.4 libidn/1.18'
  ```

3.2.3.2 Installing pycurl openssl flavour

In newer Ubuntu versions, you can install libcurl openssl flavour:

```
$ sudo apt install libcurl4-openssl-dev
$ sudo pip3 install --upgrade wfuzz
```

3.2.3.3 Installing pycurl against openssl

Alternatively, it can be done manually:

1. sudo apt-get install build-essential fakeroot dpkg-dev
2. mkdir ~/python-pycurl-openssl
3. cd ~/python-pycurl-openssl
4. sudo apt-get source python-pycurl
5. sudo apt-get build-dep python-pycurl -y

6. sudo apt-get install libcurl4-openssl-dev -y * **CAUTION: BE CAREFUL WITH THIS OR DELETE THE DIRECTORY MANUALLY TO BE SAFE** * 7. sudo rm -r *./ / ; dpkg-source -x pycurl_7*.dsc # * **CAUTION: BE CAREFUL WITH THIS OR DELETE THE DIRECTORY MANUALLY TO BE SAFE** * 8. cd pycurl*/ 9. edit debian/control file and replace all instances of "libcurl4-gnutls-dev" with "libcurl4-openssl-dev": sed -i 's/libcurl4-gnutls-dev/libcurl4-openssl-dev/g' debian/control sed -i 's/rm -f/rm -rf/g' debian/rules # fix debian/rules 'rm -r' typo preventing existing directory delete 10. sudo PYCURL_SSL_LIBRARY=openssl; dpkg-buildpackage -rfakeroot -b -uc -us 11. sudo dpkg -i ../python-pycurl_7*.deb

If there is still the error:

```
ImportError?: No module named bottle
```

Check this http://stackoverflow.com/questions/9122200/importerror-no-module-named-bottle

3.3 Breaking changes

Following https://semver.org/ versioning since Wfuzz 3.0.0.

- **Wfuzz 3.0.0:**
  - In wfuzz library prefilter is a list of filters not a string.
  - When using –recipe, stored options that are a list are appended. Previously, the last one took precedence.

CHAPTER 4

User Guide

4.1 Getting Started

A typical Wfuzz command line execution, specifying a dictionary payload and a URL, looks like this:

```
$ wfuzz -w wordlist/general/common.txt http://testphp.vulnweb.com/FUZZ
```

The obtained output is shown below:

```
********************************************************
* Wfuzz 2.2 - The Web Fuzzer                           *
********************************************************

Target: http://testphp.vulnweb.com/FUZZ
Total requests: 950

==================================================================
ID      Response   Lines      Word         Chars          Request
==================================================================

00006:  C=301      7 L        12 W          184 Ch        "admin"
00135:  C=403     10 L        29 W          263 Ch        "cgi-bin"
00379:  C=301      7 L        12 W          184 Ch        "images"
00686:  C=301      7 L        12 W          184 Ch        "secured"
...
00935:  C=301      7 L        12 W          184 Ch        "CVS"

Total time: 4.214460
Processed Requests: 950
Filtered Requests: 0
Requests/sec.: 225.4143
```

Wfuzz output allows to analyse the web server responses and filter the desired results based on the HTTP response message obtained, for example, response codes, response length, etc.

Each line provides the following information:

- ID: The request number in the order that it was performed.
- Response: Shows the HTTP response code.
- Lines: Shows the number of lines in the HTTP response.
- Word: Shows the number of words in the HTTP response.
- Chars: Shows the number of characters in the HTTP response.
- Payload: Shows the payload used.

4.1.1 Getting help

Use the –h and –help switch to get basic and advanced help usage respectively.

Wfuzz is a completely modular framework, you can check the available modules by using the -e <<category>> switch:

```
$ wfuzz -e iterators

Available iterators:

Name    | Summary
------------------------------------------------------------------------------------
↪--------
product | Returns an iterator cartesian product of input iterables.
zip     | Returns an iterator that aggregates elements from each of the iterables.
chain   | Returns an iterator returns elements from the first iterable until it is␣
↪exhaust
        | ed, then proceeds to the next iterable, until all of the iterables are␣
↪exhausted
        | .
```

Valid categories are: payloads, encoders, iterators, printers or scripts.

4.1.2 Payloads

Wfuzz is based on a simple concept: it replaces any reference to the keyword FUZZ by the value of a given payload. A payload in Wfuzz is a source of input data.

The available payloads can be listed by executing:

```
$ wfuzz -e payloads
```

Detailed information about payloads could be obtained by executing:

```
$ wfuzz -z help
```

The latter can be filtered using the –slice parameter:

```
$ wfuzz -z help --slice "dirwalk"

Name: dirwalk 0.1
Categories: default
Summary: Returns filename's recursively from a local directory.
Description:
```

(continues on next page)

(continued from previous page)

```
   Returns all the file paths found in the specified directory.
   Handy if you want to check a directory structure against a webserver,
   for example, because you have previously downloaded a specific version
   of what is supposed to be on-line.
Parameters:
   + dir: Directory path to walk and generate payload from.
```

4.1.2.1 Specifying a payload:

Each FUZZ keyword must have its corresponding payload. There are several equivalent ways of specifying a payload:

- The long way explicitly defining the payload's parameter name through the command line:

```
$ wfuzz -z file --zP fn=wordlist/general/common.txt http://testphp.vulnweb.com/
↪FUZZ
```

- The not so long way explicitly defining the payload's default parameter through the –zD command line option:

```
$ wfuzz -z file --zD wordlist/general/common.txt http://testphp.vulnweb.com/FUZZ
```

- The not so long way defining only the value of the payload's default parameter:

```
$ wfuzz -z file,wordlist/general/common.txt http://testphp.vulnweb.com/FUZZ
```

- The short way when using the file payload alias:

```
$ wfuzz -w wordlist/general/common.txt http://testphp.vulnweb.com/FUZZ
```

The stdin payload could be used when using a external wordlist generator:

```
$ crunch 2 2 ab | wfuzz -z stdin http://testphp.vulnweb.com/FUZZ
Crunch will now generate the following amount of data: 12 bytes
0 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 4
********************************************************
* Wfuzz 2.2 - The Web Fuzzer                           *
********************************************************

Target: http://testphp.vulnweb.com/FUZZ
Total requests: <<unknown>>

==================================================================
ID      Response   Lines      Word         Chars          Request
==================================================================

00002:  C=404      7 L        12 W          168 Ch        "ab"
00001:  C=404      7 L        12 W          168 Ch        "aa"
00003:  C=404      7 L        12 W          168 Ch        "ba"
00004:  C=404      7 L        12 W          168 Ch        "bb"

Total time: 3.643738
Processed Requests: 4
```

(continues on next page)

(continued from previous page)

```
Filtered Requests: 0
Requests/sec.: 1.097773
```

4.1.2.2 Multiple payloads

Several payloads can be used by specifying several -z or -w parameters and the corresponding FUZZ, ... , FUZnZ keyword where n is the payload number. The following example, brute forces files, extension files and directories at the same time:

```
$ wfuzz -w wordlist/general/common.txt -w wordlist/general/common.txt -w wordlist/
↪general/extensions_common.txt --hc 404 http://testphp.vulnweb.com/FUZZ/FUZ2ZFUZ3Z
```

4.1.3 Filters

Filtering results in Wfuzz is paramount:

- Big dictionaries could generate a great amount of output and can easily drown out legitimate valid results.
- Triaging HTTP responses is key to perform some attacks, for example, in order to check for the presence of a SQL injection vulnerability we need to distinguish a legitimate response from the one that generates an error or different data.

Wfuzz allows to filter based on the HTTP responses code and the length of the received information (in the form of words, characters or lines). Regular expressions can also be used. Two approaches can be taken: showing or hiding results matching a given filter.

4.1.3.1 Hiding responses

The following command line parameters can be used to hide certain HTTP responses "–hc, –hl, –hw, –hh". For example, the following command filters the web resources unknown by the web server (http://en.wikipedia.org/wiki/HTTP_404):

```
wfuzz -w wordlist/general/common.txt --hc 404 http://testphp.vulnweb.com/FUZZ
```

Multiple values can be specified, for example, the following wfuzz execution adds the forbidden resources to the filter:

```
wfuzz -w wordlist/general/common.txt --hc 404,403 http://testphp.vulnweb.com/FUZZ
```

Lines, words or chars are handy when we are looking for resources with the same HTTP status code. For example, it is a common behaviour (sometimes due to misconfiguration) that web servers return a custom error page with a 200 response code, this is known as soft 404.

Below is shown an example:

```
$ wfuzz -w wordlist/general/common.txt --hc 404 http://datalayer.io/FUZZ
********************************************************
* Wfuzz 2.2 - The Web Fuzzer                           *
********************************************************

Target: http://datalayer.io/FUZZ
Total requests: 950

==================================================================
```

(continues on next page)

(continued from previous page)

```
ID      Response   Lines      Word         Chars          Request
==================================================================

00000:  C=200    279 L       635 W         8972 Ch        "W3SVC3"
00001:  C=200    279 L       635 W         8972 Ch        "Log"
00002:  C=200    279 L       635 W         8972 Ch        "10"
00003:  C=200    279 L       635 W         8972 Ch        "02"
00004:  C=200    279 L       635 W         8972 Ch        "2005"
...
00024:  C=200    301 L       776 W         9042 Ch        "about"
...
```

Looking carefully at the above results, is easy to ascertain that all the "not found" resources have a common patter of 279 lines, 635 words and 8972 chars. Thus, we can improve our "–hc 404" filter by using this information (various filters can be combined):

```
$ wfuzz -w wordlist/general/common.txt --hc 404 --hh 8972  http://datalayer.io/FUZZ

00022:  C=200    301 L       776 W         9042 Ch        "about"
00084:  C=302      0 L         0 W            0 Ch        "blog"
00192:  C=302      0 L         0 W            0 Ch        "css"
...
00696:  C=200    456 L      1295 W        15119 Ch        "service"
00751:  C=200    238 L       512 W         6191 Ch        "store"
00788:  C=302      0 L         0 W            0 Ch        "text"
00913:  C=302      0 L         0 W            0 Ch        "template"
```

4.1.3.2 Showing responses

Showing results works the same way but using the command line parameters preceded by an "s": "–sc, –sl, –sw, –sh".

4.1.3.3 Using the baseline

Filters can be built against a reference HTTP response, called the "baseline". For example, the previous command for filtering "not found" resources using the –hh switch could have be done with the following command:

```
$ wfuzz -w wordlist/general/common.txt --hh BBB  http://datalayer.io/FUZZ{notthere}
...
00000:  C=200    279 L       635 W         8972 Ch        "notthere"
00001:  C=200    301 L       776 W         9042 Ch        "about"
00004:  C=200    456 L      1295 W        15119 Ch        "service"
...
```

Here the {} defines the value of the FUZZ word for this first HTTP request, and then the response can be used specifying "BBB" as a filter value.

4.1.3.4 Regex filters

The command line parameters "–ss" and "–hs" allow to filter the responses using a regular expression against the returned content. For example, the following allows to find web servers vulnerable to "shellshock" (see http://edge-security.blogspot.co.uk/2014/10/scan-for-shellshock-with-wfuzz.html for more information):

```
$ wfuzz -H "User-Agent: () { :;}; echo; echo vulnerable" --ss vulnerable -w cgis.txt␣
↪http://localhost:8000/FUZZ
```

A valid python regex should be used within these switches or an error will be prompted:

```
$ wfuzz -w wordlist/general/common.txt --hs "error)"  http://testphp.vulnweb.com/FUZZ

Fatal exception: Invalid regex expression: unbalanced parenthesis
```

4.2 Basic Usage

4.2.1 Fuzzing Paths and Files

Wfuzz can be used to look for hidden content, such as files and directories, within a web server, allowing to find further attack vectors. It is worth noting that, the success of this task depends highly on the dictionaries used.

However, due to the limited number of platforms, default installations, known resources such as logfiles, administrative directories, a considerable number of resources are located in predictable locations. Therefore, brute forcing these contents becomes a more feasible task.

Wfuzz contains some dictionaries, other larger and up to date open source word lists are:

- fuzzdb
- seclists

Below is shown an example of wfuzz looking for common directories:

```
$ wfuzz -w wordlist/general/common.txt http://testphp.vulnweb.com/FUZZ
```

Below is shown an example of wfuzz looking for common files:

```
$ wfuzz -w wordlist/general/common.txt http://testphp.vulnweb.com/FUZZ.php
```

4.2.2 Fuzzing Parameters In URLs

You often want to fuzz some sort of data in the URL's query string, this can be achieved by specifying the FUZZ keyword in the URL after a question mark:

```
$ wfuzz -z range,0-10 --hl 97 http://testphp.vulnweb.com/listproducts.php?cat=FUZZ
```

4.2.3 Fuzzing POST Requests

If you want to fuzz some form-encoded data like an HTML form will do, simply pass a -d command line argument:

```
$ wfuzz -z file,wordlist/others/common_pass.txt -d "uname=FUZZ&pass=FUZZ"  --hc 302␣
↪http://testphp.vulnweb.com/userinfo.php
********************************************************
* Wfuzz 2.2 - The Web Fuzzer                           *
********************************************************

Target: http://testphp.vulnweb.com/userinfo.php
```

(continues on next page)

(continued from previous page)

```
Total requests: 52

==================================================================
ID      Response   Lines      Word         Chars          Request
==================================================================

00044:  C=200    114 L       356 W         5111 Ch        "test"

Total time: 2.140146
Processed Requests: 52
Filtered Requests: 51
Requests/sec.: 24.29739
```

4.2.4 Fuzzing Cookies

To send your own cookies to the server, for example, to associate a request to HTTP sessions, you can use the -b parameter (repeat for various cookies):

```
$ wfuzz -z file,wordlist/general/common.txt -b cookie=value1 -b cookie2=value2 http://
↪testphp.vulnweb.com/FUZZ
```

The command above will generate HTTP requests such as the one below:

```
GET /attach HTTP/1.1
Host: testphp.vulnweb.com
Accept: */*
Content-Type:  application/x-www-form-urlencoded
Cookie:  cookie=value1; cookie2=value2
User-Agent:  Wfuzz/2.2
Connection: close
```

Cookies can also be fuzzed:

```
$ wfuzz -z file,wordlist/general/common.txt -b cookie=FUZZ http://testphp.vulnweb.com/
```

4.2.5 Fuzzing Custom headers

If you'd like to add HTTP headers to a request, simply use the -H parameter (repeat for various headers):

```
$ wfuzz -z file,wordlist/general/common.txt -H "myheader: headervalue" -H "myheader2:␣
↪headervalue2" http://testphp.vulnweb.com/FUZZ
```

The command above will generate HTTP requests such as the one below:

```
GET /agent HTTP/1.1
Host: testphp.vulnweb.com
Accept: */*
Myheader2:  headervalue2
Myheader:  headervalue
Content-Type:  application/x-www-form-urlencoded
User-Agent:  Wfuzz/2.2
Connection: close
```

You can modify existing headers, for example, for specifying a custom user agent, execute the following:

```
$ wfuzz -z file,wordlist/general/common.txt -H "myheader: headervalue" -H "User-
↪Agent: Googlebot-News" http://testphp.vulnweb.com/FUZZ
```

The command above will generate HTTP requests such as the one below:

```
GET /asp HTTP/1.1
Host: testphp.vulnweb.com
Accept: */*
Myheader:  headervalue
Content-Type:  application/x-www-form-urlencoded
User-Agent:  Googlebot-News
Connection: close
```

Headers can also be fuzzed:

```
$ wfuzz -z file,wordlist/general/common.txt -H "User-Agent: FUZZ" http://testphp.
↪vulnweb.com/
```

4.2.6 Fuzzing HTTP Verbs

HTTP verbs fuzzing can be specified using the -X switch:

```
$ wfuzz -z list,GET-HEAD-POST-TRACE-OPTIONS -X FUZZ http://testphp.vulnweb.com/
********************************************************
* Wfuzz 2.2 - The Web Fuzzer                           *
********************************************************

Target: http://testphp.vulnweb.com/
Total requests: 5

==================================================================
ID      Response   Lines      Word         Chars          Request
==================================================================

00002:  C=200      0 L         0 W            0 Ch         "HEAD"
00004:  C=405      7 L        12 W          172 Ch         "TRACE"
00005:  C=405      7 L        12 W          172 Ch         "OPTIONS"
00001:  C=200    104 L       296 W         4096 Ch         "GET"
00003:  C=200    104 L       296 W         4096 Ch         "POST"

Total time: 1.030354
Processed Requests: 5
Filtered Requests: 0
Requests/sec.: 4.852696
```

If you want to perform the requests using a specific verb you can also use "-X HEAD".

4.2.7 Proxies

If you need to use a proxy, simply use the -p parameter:

```
$ wfuzz -z file,wordlist/general/common.txt -p localhost:8080 http://testphp.vulnweb.
↪com/FUZZ
```

In addition to basic HTTP proxies, Wfuzz also supports proxies using the SOCKS4 and SOCKS5 protocol:

```
$ wfuzz -z file,wordlist/general/common.txt -p localhost:2222:SOCKS5 http://testphp.
↪vulnweb.com/FUZZ
```

Multiple proxies can be used simultaneously by supplying various -p parameters:

```
$ wfuzz -z file,wordlist/general/common.txt -p localhost:8080 -p localhost:9090 http:/
↪/testphp.vulnweb.com/FUZZ
```

Each request will be performed using a different proxy each time.

4.2.8 Authentication

Wfuzz can set an authentication headers by using the –basic/ntlm/digest command line switches.

For example, a protected resource using Basic authentication can be fuzzed using the following command:

```
$ wfuzz -z list,nonvalid-httpwatch --basic FUZZ:FUZZ https://www.httpwatch.com/
↪httpgallery/authentication/authenticatedimage/default.aspx
********************************************************
* Wfuzz 2.2 - The Web Fuzzer                           *
********************************************************

Target: https://www.httpwatch.com/httpgallery/authentication/authenticatedimage/
↪default.aspx
Total requests: 2

==================================================================
ID      Response   Lines      Word         Chars          Request
==================================================================

00001:  C=401      0 L        11 W           58 Ch        "nonvalid"
00002:  C=200     20 L        91 W         5294 Ch        "httpwatch"

Total time: 0.820029
Processed Requests: 2
Filtered Requests: 0
Requests/sec.: 2.438938
```

If you want to fuzz a resource from a protected website you can also use "–basic user:pass".

4.2.9 Recursion

The -R switch can be used to specify a payload recursion's depth. For example, if you want to search for existing directories and then fuzz within these directories again using the same payload you can use the following command:

```
$ wfuzz -z list,"admin-CVS-cgi\-bin"  -R1 http://testphp.vulnweb.com/FUZZ
********************************************************
* Wfuzz 2.2 - The Web Fuzzer                           *
********************************************************

Target: http://testphp.vulnweb.com/FUZZ
Total requests: 3

==================================================================
ID      Response   Lines      Word         Chars          Request
```

(continues on next page)

(continued from previous page)

```
===================================================================

00003:  C=403     10 L        29 W          263 Ch        "cgi-bin"
00002:  C=301      7 L        12 W          184 Ch        "CVS"
|_ Enqueued response for recursion (level=1)
00001:  C=301      7 L        12 W          184 Ch        "admin"
|_ Enqueued response for recursion (level=1)
00008:  C=404      7 L        12 W          168 Ch        "admin - CVS"
00007:  C=404      7 L        12 W          168 Ch        "admin - admin"
00005:  C=404      7 L        12 W          168 Ch        "CVS - CVS"
00006:  C=404      7 L        12 W          168 Ch        "CVS - cgi-bin"
00009:  C=404      7 L        12 W          168 Ch        "admin - cgi-bin"
00004:  C=404      7 L        12 W          168 Ch        "CVS - admin"
```

4.2.10 Perfomance

Several options lets you fine tune the HTTP request engine, depending on the performance impact on the application, and on your own processing power and bandwidth.

You can increase or decrease the number of simultaneous requests to make your attack proceed faster or slower by using the -t switch.

You can tell Wfuzz to stop a given number of seconds before performing another request using the -s parameter.

4.2.11 Writing to a file

Wfuzz supports writing the results to a file in a different format. This is performed by plugins called "printers". The available printers can be listed executing:

```
$ wfuzz -e printers
```

For example, to write results to an output file in JSON format use the following command:

```
$ wfuzz -f /tmp/outfile,json -w wordlist/general/common.txt http://testphp.vulnweb.
↪com/FUZZ
```

4.2.12 Different output

Wfuzz supports showing the results in various formats. This is performed by plugins called "printers". The available printers can be listed executing:

```
$ wfuzz -e printers
```

For example, to show results in JSON format use the following command:

```
$ wfuzz -o json -w wordlist/general/common.txt http://testphp.vulnweb.com/FUZZ
```

When using the default or raw output you can also select additional FuzzResult's fields to show, using –efield, together with the payload description:

```
$ wfuzz -z range --zD 0-1 -u http://testphp.vulnweb.com/artists.php?artist=FUZZ --
↪efield r
...
000000001:   200        99 L     272 W    3868 Ch   0 | GET /artists.php?artist=0␣
↪HTTP/1.1
                                                    Content-Type: application/x-www-
↪form-urlencoded
                                                    User-Agent: Wfuzz/2.4
                                                    Host: testphp.vulnweb.com
...
```

The above command is useful, for example, to debug what exact HTTP request Wfuzz sent to the remote Web server.

To completely replace the default payload output you can use –field instead:

```
$ wfuzz -z range --zD 0-1 -u http://testphp.vulnweb.com/artists.php?artist=FUZZ --
↪field url
...
000000001:   200        104 L    364 W    4735 Ch     "http://testphp.vulnweb.com/
↪artists.php?artist=0"
...
```

–efield and –field can be repeated to show several fields:

```
$ wfuzz -z range --zD 0-1 -u http://testphp.vulnweb.com/artists.php?artist=FUZZ --
↪efield url --efield h
...
000000001:   200        104 L    364 W    4735 Ch     "0 | http://testphp.vulnweb.com/
↪artists.php?artist=0 | 4735"
...
```

The field printer can be used with a –efield or –field expression to list only the specified filter expressions without a header or footer:

```
$ wfuzz -z list --zD https://www.airbnb.com/ --script=links --script-args=links.
↪regex=.*js$,links.enqueue=False -u FUZZ -o field --field plugins.links.link | head -
↪n3
https://a0.muscache.com/airbnb/static/packages/4e8d-d5c346ee.js
https://a0.muscache.com/airbnb/static/packages/7afc-ac814a17.js
https://a0.muscache.com/airbnb/static/packages/7642-dcf4f8dc.js
```

The above command is useful, for example, to pipe wfuzz into other tools or perform console scripts.

–efield and –field are in fact filter expressions. Check the filter language section in the advance usage document for the available fields and operators.

4.3 Advanced Usage

4.3.1 Wfuzz global options

Wfuzz global options can be tweaked by modifying the "wfuzz.ini" at the user's home directory:

```
~/.wfuzz$ cat wfuzz.ini

[connection]
```

(continues on next page)

(continued from previous page)

```
concurrent = 10
conn_delay = 90
req_delay = 90
retries = 3
user-agent = Wfuzz/2.2

[general]
default_printer = raw
cancel_on_plugin_except = 1
concurrent_plugins = 3
encode_space = 1
lookup_dirs = .,/home/xxx/tools/fuzzdb
```

A useful option is "lookup_dirs". This option will indicate Wfuzz, which directories to look for files, avoiding to specify a full path in the command line. For example, when fuzzing using a dictionary.

4.3.2 Iterators: Combining payloads

Payloads can be combined by using the -m parameter, in wfuzz this functionality is provided by what is called iterators, the following types are provided by default:

```
$ wfuzz -e iterators

Available iterators:

Name     | Summary
-------------------------------------------------------------------------------------
↪--------
product  | Returns an iterator cartesian product of input iterables.
zip      | Returns an iterator that aggregates elements from each of the iterables.
chain    | Returns an iterator returns elements from the first iterable until it is␣
↪exhaust
         | ed, then proceeds to the next iterable, until all of the iterables are␣
↪exhausted
```

Below are shown some examples using two different payloads containing the elements a,b,c and 1,2,3 respectively and how they can be combined using the existing iterators.

- zip:

```
wfuzz -z list,a-b-c -z list,1-2-3 -m zip http://google.com/FUZZ/FUZ2Z

00001:  C=404      9 L       32 W         276 Ch        "a - 1"
00002:  C=404      9 L       32 W         276 Ch        "c - 3"
00003:  C=404      9 L       32 W         276 Ch        "b - 2"
```

- chain:

```
wfuzz -z list,a-b-c -z list,1-2-3 -m chain http://google.com/FUZZ

00001:  C=404      9 L       32 W         280 Ch        "b"
00002:  C=404      9 L       32 W         280 Ch        "a"
00003:  C=404      9 L       32 W         280 Ch        "c"
00004:  C=404      9 L       32 W         280 Ch        "1"
00006:  C=404      9 L       32 W         280 Ch        "3"
00005:  C=404      9 L       32 W         280 Ch        "2"
```

- product:

```
wfuzz -z list,a-b-c -z list,1-2-3 http://mysite.com/FUZZ/FUZ2Z

00001:  C=404      9 L       32 W          276 Ch          "a - 2"
00002:  C=404      9 L       32 W          276 Ch          "a - 1"
00005:  C=404      9 L       32 W          276 Ch          "b - 2"
00004:  C=404      9 L       32 W          276 Ch          "a - 3"
00008:  C=404      9 L       32 W          276 Ch          "c - 2"
00003:  C=404      9 L       32 W          276 Ch          "b - 1"
00007:  C=404      9 L       32 W          276 Ch          "c - 1"
00006:  C=404      9 L       32 W          276 Ch          "b - 3"
00009:  C=404      9 L       32 W          276 Ch          "c - 3"
```

4.3.3 Encoders

In Wfuzz, a encoder is a transformation of a payload from one format to another. A list of the available encoders can be obtained using the following command:

```
$ wfuzz -e encoders
```

4.3.3.1 Specifying an encoder

Encoders are specified as a payload parameter. There are two equivalent ways of specifying an encoder within a payload:

- The long way:

```
$ wfuzz -z file --zP fn=wordlist/general/common.txt,encoder=md5  http://testphp.
↪vulnweb.com/FUZZ
********************************************************
* Wfuzz 2.2 - The Web Fuzzer                           *
********************************************************

Target: http://testphp.vulnweb.com/FUZZ
Total requests: 950

==================================================================
ID      Response   Lines      Word         Chars          Request
==================================================================

00002:  C=404      7 L        12 W          168 Ch
↪"b4b147bc522828731f1a016bfa72c073"
00003:  C=404      7 L        12 W          168 Ch
↪"96a3be3cf272e017046d1b2674a52bd3"
00004:  C=404      7 L        12 W          168 Ch
↪"a2ef406e2c2351e0b9e80029c909242d"
...
```

- The not so long way using the zE command line switch:

```
$ wfuzz -z file --zD wordlist/general/common.txt --zE md5 http://testphp.vulnweb.
↪com/FUZZ
```

- The not so long way:

```
$ wfuzz -z file,wordlist/general/common.txt,md5 http://testphp.vulnweb.com/FUZZ
```

4.3.3.2 Specifying multiple encoders

- Several encoders can be specified at once, using "-" as a separator:

```
$ wfuzz -z list,1-2-3,md5-sha1-none http://webscantest.com/FUZZ
********************************************************
* Wfuzz 2.2 - The Web Fuzzer                           *
********************************************************

Target: http://webscantest.com/FUZZ
Total requests: 9

==================================================================
ID      Response   Lines      Word         Chars          Request
==================================================================

00000:  C=200      38 L       121 W        1486 Ch
↪"da4b9237bacccdf19c0760cab7aec4a8359010b0"
00001:  C=200      38 L       121 W        1486 Ch
↪"c4ca4238a0b923820dcc509a6f75849b"
00002:  C=200      38 L       121 W        1486 Ch        "3"
00003:  C=200      38 L       121 W        1486 Ch
↪"77de68daecd823babbb58edb1c8e14d7106e83bb"
00004:  C=200      38 L       121 W        1486 Ch        "1"
00005:  C=200      38 L       121 W        1486 Ch
↪"356a192b7913b04c54574d18c28d46e6395428ab"
00006:  C=200      38 L       121 W        1486 Ch
↪"eccbc87e4b5ce2fe28308fd9f2a7baf3"
00007:  C=200      38 L       121 W        1486 Ch        "2"
00008:  C=200      38 L       121 W        1486 Ch
↪"c81e728d9d4c2f636f067f89cc14862c"

Total time: 0.428943
Processed Requests: 9
Filtered Requests: 0
Requests/sec.: 20.98180
```

- Encoders can also be chained using the "@" char:

```
$ wfuzz -z list,1-2-3,sha1-sha1@none http://webscantest.com/FUZZ
********************************************************
* Wfuzz 2.2 - The Web Fuzzer                           *
********************************************************

Target: http://webscantest.com/FUZZ
Total requests: 6

==================================================================
ID      Response   Lines      Word         Chars          Request
==================================================================

00000:  C=200      38 L       121 W        1486 Ch
↪"356a192b7913b04c54574d18c28d46e6395428ab"
00001:  C=200      38 L       121 W        1486 Ch
↪"356a192b7913b04c54574d18c28d46e6395428ab"
```

(continues on next page)

(continued from previous page)

```
00002:  C=200      38 L        121 W          1486 Ch
↪"77de68daecd823babbb58edb1c8e14d7106e83bb"
00003:  C=200      38 L        121 W          1486 Ch
↪"da4b9237bacccdf19c0760cab7aec4a8359010b0"
00004:  C=200      38 L        121 W          1486 Ch
↪"da4b9237bacccdf19c0760cab7aec4a8359010b0"
00005:  C=200      38 L        121 W          1486 Ch
↪"77de68daecd823babbb58edb1c8e14d7106e83bb"
```

The above "sha1@none" parameter specification will encode the payload using the sha1 encoder and the result will be encoded again using the none encoder.

- Encoders are grouped by categories. This allows to select several encoders by category, for example:

```
$ wfuzz -z list,1-2-3,hashes http://webscantest.com/FUZZ

00000:  C=200      38 L        121 W          1486 Ch         "Mw=="
00001:  C=200      38 L        121 W          1486 Ch
↪"c81e728d9d4c2f636f067f89cc14862c"
00002:  C=200      38 L        121 W          1486 Ch
↪"77de68daecd823babbb58edb1c8e14d7106e83bb"
00003:  C=200      38 L        121 W          1486 Ch
↪"da4b9237bacccdf19c0760cab7aec4a8359010b0"
00004:  C=200      38 L        121 W          1486 Ch
↪"c4ca4238a0b923820dcc509a6f75849b"
00005:  C=200      38 L        121 W          1486 Ch
↪"356a192b7913b04c54574d18c28d46e6395428ab"
00006:  C=200      38 L        121 W          1486 Ch         "MQ=="
00007:  C=200      38 L        121 W          1486 Ch         "Mg=="
00008:  C=200      38 L        121 W          1486 Ch
↪"eccbc87e4b5ce2fe28308fd9f2a7baf3"
```

4.3.4 Scan/Parse Plugins

Wfuzz is more than a Web Content Scanner. Wfuzz could help you to secure your web applications by finding and exploiting web application vulnerabilities.

Wfuzz's web application vulnerability scanner is supported by plugins. A list of scanning plugins can be obtained using the following command:

```
$ wfuzz -e scripts
```

Scripts are grouped in categories. A script could belong to several categories at the same time.

There are two general categories:

- passive: Passive scripts analyse existing requests and responses without performing new requests.
- active: Active scripts perform new requests to the application to probe it for vulnerabilities.

Additional categories are:

- discovery: Discovery plugins help crawling a website by automatically enqueuing discovered content to wfuzz request's pool.

The default category groups the plugins that are run by default.

Scanning mode is indicated when using the –script parameter followed by the selected plugins. Plugins could be selected by category or name, wildcards can also be used.

The -A switch is an alias for –script=default.

Script's detailed information can be obtained using –scrip-help, for example:

```
$ wfuzz --script-help=default
```

An example, parsing a "robots.txt" file is shown below:

```
$ wfuzz --script=robots -z list,robots.txt http://www.webscantest.com/FUZZ
********************************************************
* Wfuzz 2.2 - The Web Fuzzer                           *
********************************************************

Target: http://www.webscantest.com/FUZZ
Total requests: 1

==================================================================
ID      Response   Lines      Word         Chars          Request
==================================================================

00001:  C=200      6 L        10 W          101 Ch        "robots.txt"
|_ Plugin robots enqueued 4 more requests (rlevel=1)
00002:  C=200     40 L       117 W         1528 Ch        "/osrun/"
00003:  C=200     55 L       132 W         1849 Ch        "/cal_endar/"
00004:  C=200     40 L       123 W         1611 Ch        "/crawlsnags/"
00005:  C=200     85 L       197 W         3486 Ch        "/static/"

Total time: 0
Processed Requests: 5 (1 + 4)
Filtered Requests: 0
Requests/sec.: 0
```

In order to not scan the same requests (with the same parameters) over an over again, there is a cache,the cache can be disabled with the –no-cache flag.

For example, if we target a web server with the same URL but different parameter values, we get:

```
$ wfuzz -z range --zD 0-3 -z list --zD "'" -u http://testphp.vulnweb.com/artists.php?
↪artist=FUZZFUZ2Z -A

000000004:   0.195s       200         101 L     287 W    3986 Ch     nginx/1.4.1       ␣
↪                                                "3 - '"
|_  Error identified: Warning: mysql_fetch_array()
000000001:   0.198s       200         101 L     287 W    3986 Ch     nginx/1.4.1       ␣
↪                                                "0 - '"
000000002:   0.198s       200         101 L     287 W    3986 Ch     nginx/1.4.1       ␣
↪                                                "1 - '"
000000003:   0.198s       200         101 L     287 W    3986 Ch     nginx/1.4.1       ␣
↪                                                "2 - '"
```

But, if we do the same but disabling the cache:

```
$ wfuzz -z range --zD 0-3 -z list --zD "'" -u http://testphp.vulnweb.com/artists.php?
↪artist=FUZZFUZ2Z -A --no-cache

000000004:   1.170s       200         101 L     287 W    3986 Ch     nginx/1.4.1       ␣
↪                                                "3 - '"
```

(continues on next page)

(continued from previous page)

```
|_  Error identified: Warning: mysql_fetch_array()
000000002:   1.173s       200        101 L    287 W    3986 Ch     nginx/1.4.1        ␣
↪                                                   "1 - '"
|_  Error identified: Warning: mysql_fetch_array()
000000001:   1.174s       200        101 L    287 W    3986 Ch     nginx/1.4.1        ␣
↪                                                   "0 - '"
|_  Error identified: Warning: mysql_fetch_array()
000000003:   1.173s       200        101 L    287 W    3986 Ch     nginx/1.4.1        ␣
↪                                                   "2 - '"
|_  Error identified: Warning: mysql_fetch_array()
```

4.3.4.1 Custom scripts

If you would like to create customs scripts, place them in your home directory. In order to leverage this feature, a directory named "scripts" must be created underneath the ".wfuzz" directory.

4.3.5 Recipes

You could save Wfuzz command line options to a file for later execution or for easy distribution.

To create a recipe, execute the following:

```
$ wfuzz --script=robots -z list,robots.txt --dump-recipe /tmp/recipe http://www.
↪webscantest.com/FUZZ
```

Then, execute Wfuzz using the stored options by using the "–recipe" option:

```
$ wfuzz --recipe /tmp/recipe
********************************************************
* Wfuzz 2.2 - The Web Fuzzer                           *
********************************************************

Target: http://www.webscantest.com/FUZZ
Total requests: 1

==================================================================
ID      Response   Lines      Word         Chars          Request
==================================================================

00001:  C=200      6 L        10 W          101 Ch        "robots.txt"
|_ Plugin robots enqueued 4 more requests (rlevel=1)
00002:  C=200     40 L       117 W         1528 Ch        "/osrun/"
00003:  C=200     55 L       132 W         1849 Ch        "/cal_endar/"
00004:  C=200     40 L       123 W         1611 Ch        "/crawlsnags/"
00005:  C=200     85 L       197 W         3486 Ch        "/static/"

Total time: 1.341176
Processed Requests: 5 (1 + 4)
Filtered Requests: 0
Requests/sec.: 3.728071
```

You can combine a recipe with additional command line options, for example:

```
$ wfuzz --recipe /tmp/recipe -b cookie1=value
```

Several recipes can also be combined:

```
$ wfuzz --recipe /tmp/recipe --recipe /tmp/recipe2
```

In case of repeated options, command line options have precedence over options included in the recipe. Last recipe has precedence.

4.3.6 Connect to an specific host

The –ip option can be used to connect to a specific host and port instead of the URL's host and port:

```
$ wfuzz -z range,1-1 --ip 127.0.0.1 http://www.google.com/anything/FUZZ
```

This useful, for example, to test if a reverse proxy can be manipulated into misrouting requests to a destination of our choice.

4.3.7 Scan Mode: Ignore Errors and Exceptions

In the event of a network problem (e.g. DNS failure, refused connection, etc.), Wfuzz will raise an exception and stop execution as shown below:

```
$ wfuzz -z list,support-web-none http://FUZZ.google.com/
********************************************************
* Wfuzz 2.2 - The Web Fuzzer                           *
********************************************************

Target: http://FUZZ.google.com/
Total requests: 3

==================================================================
ID      Response   Lines      Word         Chars          Request
==================================================================


Fatal exception: Pycurl error 6: Could not resolve host: none.google.com
```

You can tell Wfuzz to continue execution, ignoring errors by supplying the -Z switch. The latter command in scan mode will get the following results:

```
$ wfuzz -z list,support-web-none -Z http://FUZZ.google.com/
********************************************************
* Wfuzz 2.2 - The Web Fuzzer                           *
********************************************************

Target: http://FUZZ.google.com/
Total requests: 3

==================================================================
ID      Response   Lines      Word         Chars          Request
==================================================================

00002:  C=404     11 L        72 W        1561 Ch         "web"
00003:  C=XXX      0 L         0 W           0 Ch         "none! Pycurl error 6:␣
↪Could not resolve host: none.google.com"
00001:  C=301      6 L        14 W         224 Ch         "support"
```

(continues on next page)

(continued from previous page)

```

Total time: 1.064229
Processed Requests: 3
Filtered Requests: 0
Requests/sec.: 2.818939
```

Errors are shown as a result with the XXX code, the payload used followed by an exclamation mark and the companion exception message. Error codes can be filtered using the “XXX” expression. For example:

```
$ wfuzz -z list,support-web-none -Z --hc XXX http://FUZZ.google.com/
********************************************************
* Wfuzz 2.2 - The Web Fuzzer                           *
********************************************************

Target: http://FUZZ.google.com/
Total requests: 3

==================================================================
ID      Response   Lines      Word         Chars          Request
==================================================================

00002:  C=404     11 L        72 W         1561 Ch        "web"
00001:  C=301      6 L        14 W          224 Ch        "support"

Total time: 0.288635
Processed Requests: 3
Filtered Requests: 1
Requests/sec.: 10.39374
```

When Wfuzz is used in scan mode, HTTP requests will take longer time due to network error timeouts. These can be tweaked using the –req-delay and –conn-delay command line parameters.

4.3.7.1 Timeouts

You can tell Wfuzz to stop waiting for server to response a connection request after a given number of seconds –conn-delay and also the maximum number of seconds that the response is allowed to take using –req-delay parameter.

These timeouts are really handy when you are using Wfuzz to brute force resources behind a proxy, ports, hostnames, virtual hosts, etc.

4.3.8 Filter Language

Wfuzz’s filter language grammar is build using pyparsing, therefore it must be installed before using the command line parameters “–filter, –prefilter, –slice, –field and –efield”.

The information about the filter language can be also obtained executing:

```
wfuzz --filter-help
```

A filter expression must be built using the following symbols and operators:

- Boolean Operators

“and”, “or” and “not” operators could be used to build conditional expressions.

- Expression Operators

Expressions operators such as “= != < > >= <=” could be used to check values. Additionally, the following operators for matching text are available:

Operator	Description
=~	True when the regular expression specified matches the value.
~	Equivalent to Python’s “str2” in “str1” (case insensitive)
!~	Equivalent to Python’s “str2” not in “str1” (case insensitive)

Also, assignment operators:

Operator	Description
:=	Assigns a value
=+	Concatenates value at the left
=-	Concatenates value at the right

Where values could be:

- Basic primitives:

Long Name	Description
‘string’	Quoted string
0..9+	Integer values
XXX	HTTP request error code
BBB	Baseline

- Values can also be modified using the following operators:

Name	Short version	Description
value|unquote()	value|un()	Unquotes the value
value|lower()	value|l()	lower-case of the value
value|upper()		upper-case of the value
value|encode(‘encoder’, ‘value’)	value|e(‘enc’, ‘val’)	Returns encoder.encode(value)
value|decode(‘decoder’, ‘value’)	value|d(‘dec’, ‘val’)	Returns encoder.decode(value)
value|replace(‘what’, ‘with’)	value|r(‘what’, ‘with’)	Returns value replacing what for with
value|unique()	value|u()	Returns True if a value is unique.
value|startswith(‘value’)	value|sw(‘value’)	Returns true if the value string starts with param
value|gregex(‘expression’)	value|gre(‘exp’)	Returns first regex group that matches in value
value|diff(expression)		Returns diff comparison between value and expression

- When a FuzzResult is available, you could perform runtime introspection of the objects using the following symbols

Name	Short version	Description
url		Wfuzz's result HTTP request url
description		Wfuzz's result description
nres		Wfuzz's result identifier
code	c	Wfuzz's result HTTP response's code
chars	h	Wfuzz's result HTTP response chars
lines	l	Wfuzz's result HTTP response lines
words	w	Wfuzz's result HTTP response words
md5		Wfuzz's result HTTP response md5 hash
history	r	Wfuzz's result associated FuzzRequest object
plugins		Wfuzz's plugins scan results

FuzzRequest object's attribute (you need to use the r. prefix) such as:

Name	Description
url	HTTP request's url
urlp	HTTP request's parsed url (see section below).
method	HTTP request's verb
scheme	HTTP request's scheme
host	HTTP request's host
content	HTTP response's content
raw_content	HTTP response's content including headers
cookies.all	All HTTP request and response cookies
cookies.request	HTTP requests cookieS
cookies.response	HTTP response cookies
cook-ies.request.<<name>>	Specified HTTP request cookie
cook-ies.response.<<name>>	Specified HTTP response cookie
headers.all	All HTTP request and response headers
headers.request	HTTP request headers
headers.response	HTTP response headers
head-ers.request.<<name>>	Specified HTTP request header case insensitive
head-ers.response.<<name>>	Specified HTTP response header insensitive
params.all	All HTTP request GET and POST parameters
params.get	All HTTP request GET parameters
params.post	HTTP request POST parameters in returned as a dictionary
params.raw_post	HTTP request POST parameters payload
params.get.<<name>>	Spcified HTTP request GET parameter
params.post.<<name>>	Spcified HTTP request POST parameter
pstrip	Returns a signature of the HTTP request using the parameter's names without values (useful for unique operations)
is_path	Returns true when the HTTP request path refers to a directory.
reqtime	Returns the total time that HTTP request took to be retrieved

It is worth noting that Wfuzz will try to parse the POST parameters according to the specified content type header. Currently, application/x-www-form-urlencoded, multipart/form-dat and application/json are supported. This is prone to error depending on the data format, raw_post will not try to do any processing.

FuzzRequest URL field is broken in smaller (read only) parts using the urlparse Python's module in the urlp attribute.

Urlparse parses a URL into: scheme://netloc/path;parameters?query#fragment. For example, for the "http://www.google.com/dir/test.php?id=1" URL you can get the following values:

Name	Value
urlp.scheme	http
urlp.netloc	www.google.com
urlp.path	/dir/test.php
urlp.params	
urlp.query	id=1
urlp.fragment	
urlp.ffname	test.php
urlp.fext	.php
urlp.fname	test
urlp.hasquery	Returns true when the URL contains a query string.
urlp.isbllist	Returns true when the URL file extension is included in the configuration discovery's blacklist

Payload introspection can also be performed by using the keyword FUZZ:

Name	Description
FUZnZ	Allows to access the Nth payload string
FUZnZ[field]	Allows to access the Nth payload attributes

Where field is one of the described above.

4.3.8.1 Filtering results

The –filter command line parameter in conjunction with the described filter language allows you to perform more complex result triage than the standard filter switches such as "–hc/hl/hw/hh", "–sc/sl/sw/sh" and "-ss/hs".

An example below:

```
$ wfuzz -z range,0-10 --filter "c=200 and l>97" http://testphp.vulnweb.com/
↪listproducts.php?cat=FUZZ
********************************************************
* Wfuzz 2.2 - The Web Fuzzer                           *
********************************************************

Target: http://testphp.vulnweb.com/listproducts.php?cat=FUZZ
Total requests: 11

==================================================================
ID      Response   Lines      Word         Chars          Request
==================================================================

00003:  C=200      99 L       302 W        4442 Ch        "2"
00002:  C=200     102 L       434 W        7011 Ch        "1"

Total time: 1.452705
Processed Requests: 11
Filtered Requests: 9
Requests/sec.: 7.572076
```

Using result and payload introspection to look for specific content returned in the response:

```
$ wfuzz -z list,echoedback -d searchFor=FUZZ --filter "content~FUZZ" http://testphp.
↪vulnweb.com/search.php?test=query
```

Which is equivalent to:

```
$ wfuzz -z list,echoedback -d searchFor=FUZZ --ss "echoedback" http://testphp.vulnweb.
↪com/search.php?test=query
```

A more interesting variation of the above examples could be:

```
$ wfuzz -w fuzzdb/attack/xss/xss-rsnake.txt -d searchFor=FUZZ --filter "content~FUZZ"␣
↪http://testphp.vulnweb.com/search.php?test=query
```

You can use the fields as boolean values as well. For example, this filter will show only the requests with parameters:

```
$ wfuzz -z range --zD 0-1 -u http://testphp.vulnweb.com/artists.php?artist=FUZZ --
↪filter 'r.params.all'
```

Results with plugin issues can be filter as well:

```
$ wfuzz -z list --zD index -u http://testphp.vulnweb.com/FUZZ.php --script headers --
↪filter "plugins~'nginx'"
```

4.3.8.2 Payload mangling

Slicing a payload

The –slice command line parameter in conjunction with the described language allows you to filter a payload. The payload to filter, specified by the -z switch must precede –slice command line parameter.

The specified expression must return a boolean value, an example, using the unique operator is shown below:

```
$ wfuzz -z list --zD one-two-one-one --slice "FUZZ|u()" http://localhost:9000/FUZZ

********************************************************
* Wfuzz 2.2 - The Web Fuzzer                           *
********************************************************

Target: http://localhost:9000/FUZZ
Total requests: <<unknown>>

==================================================================
ID      Response   Lines      Word         Chars          Request
==================================================================

00001:  C=404      9 L        32 W          277 Ch        "one"
00002:  C=404      9 L        32 W          277 Ch        "two"

Total time: 0.031817
Processed Requests: 2
Filtered Requests: 0
Requests/sec.: 62.85908
```

It is worth noting that, the type of payload dictates the available language symbols. For example, a dictionary payload such as in the example above does not have a full FuzzResult object context and therefore object fields cannot be used.

When slicing a FuzzResult payload, you are accessing the FuzzResult directly, therefore given a previous session such as:

```
$ wfuzz -z range --zD 0-0 -u http://www.google.com/FUZZ --oF /tmp/test1
...
000000001:   404        11 L     72 W       1558 Ch     "0"
...
```

this can be used to filter the payload:

```
$ wfpayload -z wfuzzp --zD /tmp/test1 --slice "c=404"
...
000000001:   404        11 L     72 W       1558 Ch     "0"
...

$ wfpayload -z wfuzzp --zD /tmp/test1 --slice "c!=404"
...
wfuzz.py:168: UserWarning:Fatal exception: Empty dictionary! Please check payload or␣
↪filter.
...
```

In fact, in this situation, FUZZ refers to the previous result (if any):

```
$ wfuzz -z wfuzzp --zD /tmp/test1 -u FUZZ --oF /tmp/test2
...
000000001:   404        11 L     72 W       1558 Ch     "http://www.google.com/0"
...

$ wfpayload -z wfuzzp --zD /tmp/test2 --efield r.headers.response.date --efield␣
↪FUZZ[r.headers.response.date]
...
000000001:   404        11 L     72 W       1558 Ch     "http://www.google.com/0 |␣
↪Mon, 02 Nov 2020 19:29:03 GMT | Mon, 02 Nov 2020 19:27:27 GMT"
...
```

Re-writing a payload

The slice command parameter also allows to re-write a payload. Any value, other than a boolean, returned by the specified expression will be interpreted not to filter the source payload but to change its value.

For example:

```
$ ./wfuzz -z list --zD one-two-three --slice "FUZZ|upper()" -u https://www.wfuzz.io/
↪FUZZ
000000001:   404        11 L     72 W     1560 Ch     "ONE"
000000003:   404        11 L     72 W     1562 Ch     "THREE"
000000002:   404        11 L     72 W     1560 Ch     "TWO"
```

Prefilter

The –prefilter command line parameter is similar to –slice but is not associated to any payload. It is a general filtering performed just before any HTTP request is done.

In this context you are filtering a FuzzResult object, which is the result of combining all the input payloads, that is has not been updated with the result of performing its associated HTTP request yet and therefore lacking some information.

The –prefilter command cannot be used to re-write a payload. The assignment operators can be used to modify the FuzzResult object's fields but expressions other booleans will be ignored.

4.3.9 Reutilising previous results

Previously performed HTTP requests/responses contain a treasure trove of data. Wfuzz payloads and object introspection (explained in the filter grammar section) exposes a Python object interface to requests/responses recorded by Wfuzz or other tools.

This allows you to perform manual and semi-automatic tests with full context and understanding of your actions, without relying on a web application scanner underlying implementation.

Some ideas:

- Replaying individual requests as-is
- Comparing response bodies and headers of fuzzed requests against their original
- Looking for requests with the CSRF token exposed in the URL
- Looking for responses with JSON content with an incorrect content type

To reutilise previous results, a payload that generates a full FuzzResult object context should be used.

- wfuzzp payload:

Wfuzz results can be stored using the –oF option as illustrated below:

```
$ wfuzz --oF /tmp/session -z range,0-10 http://www.google.com/dir/test.php?id=FUZZ
```

- burpstate and burplog payloads:

Wfuzz can read burp's (TM) log or saved states. This allows to filter or reutilise burp proxy requests and responses.

Then, you can reutilise those results by using the denoted payloads. To repeat a request exactly how it was stored, you must use the FUZZ keyword on the command line:

```
$ wfuzz -z burpstate,a_burp_state.burp FUZZ

$ wfuzz -z burplog,a_burp_log.burp FUZZ

$ wfuzz -z wfuzzp,/tmp/session FUZZ
```

Previous requests can also be modified by using the usual command line switches. Some examples below:

- Adding a new header:

  ```
  $ wfuzz -z burpstate,a_burp_state.burp -H "addme: header" FUZZ
  ```

- Using new cookies specified by another payload:

  ```
  $ wfuzz -z burpstate,a_burp_state.burp -z list,1-2-3 -b "cookie=FUZ2Z" FUZZ
  ```

- The stored HTTP requests can be printed using the –prev flag for comparing old vs new results:

  ```
  $ wfuzz -z burpstate,testphp.burp --slice "cookies.request and url|u()" --filter
  ↪"c!=FUZZ[c]" -b "" --prev FUZZ
  ...
  000076:  C=302      0 L        3 W          14 Ch        "http://testphp.vulnweb.
  ↪com/userinfo.php"
    |__    C=200    114 L      373 W        5347 Ch        "http://testphp.vulnweb.
  ↪com/userinfo.php"
  ```

(continues on next page)

(continued from previous page)

```
```

- Same request against another URL:

```
$ wfuzz -z burpstate,a_burp_state.burp -H "addme: header" -u http://www.otherhost.
↪com FUZZ
```

If you do not want to use the full saved request:

- Accessing specific HTTP object fields can be achieved by using the attr payload's parameter:

```
$ wfuzz -z wfuzzp,/tmp/session --zP attr=url FUZZ
```

- Or by specifying the FUZZ keyword and a field name in the form of FUZZ[field]:

```
$ wfuzz -z wfuzzp,/tmp/session FUZZ[url]
```

This could be used, for example, to perform new requests based on stored values:

```
$ wfuzz -z wfuzzp,/tmp/session -p localhost:8080 http://testphp.vulnweb.com/FUZZ[url.
↪path]?FUZZ[url.query]
00001:  C=200     25 L       155 W         1362 Ch        "/dir/test.php - id=0"
...
00002:  C=200     25 L       155 W         1362 Ch        "/dir/test.php - id=1"
```

The above command will generate HTTP requests such as the following:

```
GET /dir/test.php?id=10 HTTP/1.1
Host: testphp.vulnweb.com
Accept: */*
Content-Type:  application/x-www-form-urlencoded
User-Agent:  Wfuzz/2.2
Connection: close
```

You can filter the payload using the filter grammar as described before.

4.3.10 Reutilising previous results

Plugins results contain a treasure trove of data. Wfuzz payloads and object introspection (explained in the filter grammar section) exposes a Python object interface to plugins results. This allows you to perform semi-automatic tests based on plugins results or compile a set of results to be used in another tool.

4.3.10.1 Request mangling

The assignment operators can be used to modify previous requests, for example, let's add a quote to every string parameter prior of performing the HTTP request:

```
$ wfuzz -z range,1-5 --oF /tmp/session http://testphp.vulnweb.com/artists.php?
↪artist=FUZZ
000003:  C=200    118 L      455 W         5326 Ch        "3"
...
000004:  C=200     99 L      272 W         3868 Ch        "4"

$ wfuzz -z wfuzzp,/tmp/session --prefilter "r.params.get=+'\''" -A FUZZ
```

(continues on next page)

(continued from previous page)

```
00010:  0.161s   C=200  101 L  287 W    3986 Ch    nginx/1.4.1  "http://testphp.
↪vulnweb.com/artists.php?artist=1'"
|_  Error identified: Warning: mysql_fetch_array()
...
```

The above command looks for simple SQL injection issues.

4.4 wfpayload

wfpayload uses the same motor as wfuzz but instead of performing HTTP requests, uses wfuzz's payload plugins to generate new content or analyse saved sessions.

4.4.1 Generating new dictionaries

You can use wfpayload to create new dictionaries:

```
$ wfpayload -z range --zD 0-10
0
1
2
3
4
5
6
7
8
9
10
```

The same wfuzz's syntax can be used, for example:

```
$ wfpayload -z range --zD 0-10 --filter "FUZZ<3"
0
1
2
```

4.4.2 Analysing saved sessions

Previously performed HTTP requests/responses contain a treasure trove of data. You can use wfpayload to filter and analyse previously saved sessions. Wfpayload can also read sessions from external tools, such as burp.

This allows you to look for new vulnerabilities or understand the underlying target without performing new HTTP requests.

For example, the following will return a unique list of HTTP requests including the authtoken parameter as a GET parameter:

```
$ wfpayload -z burplog,a_burp_log.log --slice "params.get~'authtoken'"
```

Authtoken is the parameter used by BEA WebLogic Commerce Servers (TM) as a CSRF token, and therefore the above will find all the requests exposing the CSRF token in the URL.

You can also look for specific parameters or headers, for example, the following will look for HTTP responses accepting any CORS origin:

```
$ wfpayload -z burplog --zD burp_log_05032020.log --prefilter "r.headers.response.
↪Access-Control-Allow-Origin='*'"
```

It is worth noting that, if the header is not present in the response it will be return an empty value, not raising any error.

You can also select the fields to show with –efield and –field, for example:

```
$ wfpayload -z wfuzzp --zD /tmp/session --field r.params.get
artist=5
...
```

Or:

```
$ wfpayload -z wfuzzp --zD /tmp/session --efield r.params.get
000000006:   200        99 L     272 W    3868 Ch     "5 | artist=5"
...
```

4.4.3 Running plugins against saved sessions

Plugins can be run against a saved session. For example:

```
$ ./wfpayload -z burplog --zD ./burp_log_05032020.log  --script=headers --filter
↪"plugins~'akamai'"
...
000000124:   302        0 L      0 W      0 Ch        "https://trial-eum-clientnsv4-s.
↪akamaihd.net/eum/getdns.txt?c=pjq71x1r7"
|_  New Server header - AkamaiGHost
000000913:   200        10 L     6571 W   289832 Ch   "https://assets.adobedtm.com/
↪2eed2bf00c8bca0c98d97ffee50a306922bc8c98/satelliteLib-
↪27b81756e778cc85cc1a2f067764cd3abf072aa9.js"
|_  New Server header - AkamaiNetStorage
...
```

4.4.4 Re-writing saved sessions

The content of a saved session can be re-written. For example, let's say there is a session with a bunch of 404/400 results that you want to remove:

```
$ wfpayload -z burplog --zD ./burp_log_05032020.log  --hc 404 --oF /tmp/no404
```

and then:

```
$ wfpayload -z wfuzzp --zD /tmp/no404
```

CHAPTER 5

Library Guide

5.1 Python library

Wfuzz's Python library allows to automate tasks and integrate Wfuzz into new tools or scripts.

5.1.1 Library Options

All options that are available within the Wfuzz command line interface are available as library options:

CLI Option	Library Option
<URL>	url=”url”
–recipe <filename>	recipe=[“filename”]
–oF <filename>	save=”filename”
-f filename,printer	printer=(“filename”, “printer”)
–dry-run	transport=”dryrun”
-p addr	proxies=[(“ip”,”port”,”type”)]
-t N	concurrent=N
-s N	delay=0.0
-R depth	rleve=depth
–follow	follow=True
-Z	scanmode=True
–req-delay N	req_delay=0
–conn-delay N	conn_delay=0.0
–no-cache	no_cache=True
–script=<plugins>	script=”plugins”
–script-args n1=v1,. . .	script_args={n1: v1}
-m iterator	iterator=”iterator”
-z payload	payloads=[(“name”,{default=”“,encoder=[“md5”]},slice=”“),]
-V alltype	allvars=”alltype”
-X method	method=”method”
–hc/hl/hw/hh N[,N]+	hc/hl/hw/hh=[N,N]
–sc/sl/sw/sh N[,N]+	sc/sl/sw/sh=[N,N]
–ss/hs regex	ss/hs=”regex”
–filter <filter>	filter=”filter exp”
–prefilter <filter>	prefilter=[“prefilter exp”]
-b cookie	cookie=[“cookie1=value1”,]
-d postdata	postdata=”postdata”
-H header	headers=[(“header1”, “value1”),]
–basic/ntlm/digest auth	auth=(“basic”, “user:pass”)

These options can be used in the main library interfaces: fuzz, payload or session indistinctly.

5.1.2 Fuzzing a URL

Fuzzing a URL with wfuzz library is very simple. Firstly, import the wfuzz module:

```
>>> import wfuzz
```

Now, let’s try to fuzz a web page to look for hidden content, such as directories. For this example, let’s use Acunetix’s testphp (http://testphp.vulnweb.com/):

```
>>> import wfuzz
>>> for r in wfuzz.fuzz(url="http://testphp.vulnweb.com/FUZZ", hc=[404], payloads=[(
↪"file",dict(fn="wordlist/general/common.txt"))]):
...     print r
...
00060:  C=301      7 L        12 W          184 Ch         "admin"
00183:  C=403     10 L        29 W          263 Ch         "cgi-bin"
00429:  C=301      7 L        12 W          184 Ch         "images"
...
```

Now, we have a FuzzResult object called r. We can get all the information we need from this object.

5.1.3 FuzzSession object

A FuzzSession object has all the methods of the main wfuzz API.

The FuzzSession object allows you to persist certain parameters across fuzzing sessions:

```
>>> import wfuzz
>>> s = wfuzz.FuzzSession(url="http://testphp.vulnweb.com/FUZZ")
>>> for r in s.fuzz(hc=[404], payloads=[("file",dict(fn="wordlist/general/common.txt
↪"))]):
...     print r
...
00060:  C=301      7 L        12 W          184 Ch        "admin"
00183:  C=403     10 L        29 W          263 Ch        "cgi-bin"
...
```

FuzzSession can also be used as context manager:

```
>>> with wfuzz.FuzzSession(url="http://testphp.vulnweb.com/FUZZ", hc=[404],␣
↪payloads=[("file",dict(fn="wordlist/general/common.txt"))]) as s:
...     for r in s.fuzz():
...             print r
...
00295:  C=301      7 L        12 W          184 Ch        "admin"
00418:  C=403     10 L        29 W          263 Ch        "cgi-bin"
```

5.1.4 Get payload

The get_payload function generates a Wfuzz payload from a Python iterable. It is a quick and flexible way of getting a payload programmatically without using Wfuzz payloads plugins.

Generating a new payload and start fuzzing is really simple:

```
>>> import wfuzz
>>> s = wfuzz.get_payload(range(5))
>>> for r in s.fuzz(url="http://testphp.vulnweb.com/FUZZ"):
...     print r
...
00012:  C=404      7 L        12 W          168 Ch        "0"
00013:  C=404      7 L        12 W          168 Ch        "1"
00014:  C=404      7 L        12 W          168 Ch        "2"
00015:  C=404      7 L        12 W          168 Ch        "3"
00016:  C=404      7 L        12 W          168 Ch        "4"
```

The get_payloads method can be used when various payloads are needed:

```
>>> import wfuzz
>>> s = wfuzz.get_payloads([range(5), ["a","b"]])
>>> for r in s.fuzz(url="http://testphp.vulnweb.com/FUZZ/FUZ2Z"):
...     print r
...
00028:  C=404      7 L        12 W          168 Ch        "4 - b"
00027:  C=404      7 L        12 W          168 Ch        "4 - a"
00024:  C=404      7 L        12 W          168 Ch        "2 - b"
00026:  C=404      7 L        12 W          168 Ch        "3 - b"
00025:  C=404      7 L        12 W          168 Ch        "3 - a"
00022:  C=404      7 L        12 W          168 Ch        "1 - b"
```

(continues on next page)

(continued from previous page)

```
00021:  C=404      7 L        12 W          168 Ch        "1 - a"
00020:  C=404      7 L        12 W          168 Ch        "0 - b"
00023:  C=404      7 L        12 W          168 Ch        "2 - a"
00019:  C=404      7 L        12 W          168 Ch        "0 - a"
```

5.1.5 Get session

The get_session function generates a Wfuzz session object from the specified command line. It is a quick way of getting a payload programmatically from a string representing CLI options:

```
$ python
>>> import wfuzz
>>> s = wfuzz.get_session("-z range,0-10 http://testphp.vulnweb.com/FUZZ")
>>> for r in s.fuzz():
...     print r
...
00002:  C=404      7 L        12 W          168 Ch        "1"
00011:  C=404      7 L        12 W          168 Ch        "10"
00008:  C=404      7 L        12 W          168 Ch        "7"
00001:  C=404      7 L        12 W          168 Ch        "0"
00003:  C=404      7 L        12 W          168 Ch        "2"
00004:  C=404      7 L        12 W          168 Ch        "3"
00005:  C=404      7 L        12 W          168 Ch        "4"
00006:  C=404      7 L        12 W          168 Ch        "5"
00007:  C=404      7 L        12 W          168 Ch        "6"
00009:  C=404      7 L        12 W          168 Ch        "8"
00010:  C=404      7 L        12 W          168 Ch        "9"
```

5.1.6 Interacting with the results

Once a Wfuzz result is available the grammar defined in the filter language can be used to work with the results' values. For example:

```
$ python
>>> import wfuzz

>>> with wfuzz.get_session("-z list --zD test -u http://testphp.vulnweb.com/userinfo.
↪php -d uname=FUZZ&pass=FUZZ") as s:
...     for r in s.fuzz():
...             print(r.history.cookies.response)
...             print(r.history.params.all)
...             print(r.history.params.post)
...             print(r.history.params.post.uname)
...             print(r.history.params.post['pass'])
{'login': 'test%2Ftest'}
{'uname': 'test', 'pass': 'test'}
{'uname': 'test', 'pass': 'test'}
test
test
>>>
```

The result object has also a method to evaluate a language expression:

```
>> print(r.eval("r.cookies.response"))
login=test%2Ftest
```