Wfuzz Documentation

Release 2.1.4

Xavi Mendez

Jan 12, 2019

Contents

Wfuzz supports Python 3. The use of **Python 3** is preferred (and faster) over Python 2.

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
```

Wfuzz has been created to facilitate the task in web applications assessments and 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 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.
- 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 1

User Guide

1.1 Installation

1.1.1 Pip install Wfuzz

To install WFuzz using pip

```
$ pip install wfuzz
```

1.1.2 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 <https://github.com/xmendez/wfuzz/releases/latest>_*.

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
```

1.1.3 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.

1.2 Installation issues

1.2.1 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 --with-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/bin:$PATH"' >> ~/.bash_profile
   ```

4. Or temporary in the current shell:

   ```
   $ export PATH="/usr/local/opt/curl/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.

1.2.2 Pycurl on Windows

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

1.2.3 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.')
```

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'
```

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
```

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
6. sudo apt-get install libcurl4-openssl-dev
7. sudo dpkg-source -x pycurl_7.19.0-3build1.dsc
8. cd pycurl-7.19.0
9. edit debian/control file and replace all instances of "libcurl4-gnutls-dev" with "libcurl4-openssl-dev"
10. sudo PYCURL_SSL_LIBRARY=openssl dpkg-buildpackage -rfakeroot -b
11. sudo dpkg -i ../python-pycurl_7.19.0-3build1_i386.deb

If there is still the error:

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

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

1.3 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 analyze 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.

1.3.1 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.

1.3.2 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:
   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.
```

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

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
```

1.3.3 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.

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

=====================================================================
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"
```

Showing responses

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

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.

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 vulnerables 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
```

1.4 Basic Usage

1.4.1 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
```

1.4.2 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
```

1.4.3 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
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
```

1.4.4 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
```

(continues on next page)

(continued from previous page)

```
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/
```

1.4.5 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/
```

1.4.6 Fuzzing HTTP Verbs

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

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

(continues on next page)

(continued from previous page)

```
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".

1.4.7 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.

1.4.8 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
```

(continues on next page)

(continued from previous page)

```
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".

1.4.9 Recursion

The -R swith 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
=====================================================================

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"
```

1.4.10 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.

1.4.11 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
```

1.4.12 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
```

1.5 Advanced Usage

1.5.1 Wfuzz global options

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

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

[connection]
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.

1.5.2 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"
```

1.5.3 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
```

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:

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

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"
```

(continues on next page)

(continued from previous page)

```
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"
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"
```

(continues on next page)

(continued from previous page)

```
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"
```

1.5.4 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.

Thre are two general categories:

- passive: Passive scripts analyze 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"
```

(continues on next page)

(continued from previous page)

```
|_ 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
```

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.

1.5.5 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
```

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

1.5.6 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"

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/
```

(continues on next page)

(continued from previous page)

```
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.

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 bruteforce resources behind a proxy, ports, hostnames, virtual hosts, etc.

1.5.7 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".

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 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)

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
valuelunquote()	valuelun()	Unquotes the value
valuellower()	valuell()	lowercase of the value
valuelupper()		uppercase of the value
valuelencode('encoder', 'value')	valuele('enc', 'val')	Returns encoder.encode(value)
valueldecode('decoder', 'value')	valueld('dec', 'val')	Returns encoder.decode(value)
valuelreplace('what', 'with')	valuelr('what', 'with')	Returns value replacing what for with
valueunique(value)	valuelu(value)	Returns True if a value is unique.
valuelstartswith('value')	valuelsw('param')	Returns true if the value string starts with param

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

Name	Short version	Description
description		Wfuzz's result description
nres		Wfuzz's result identifier
code	c	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

Or FuzzRequest object's attribute such as:

Name	Description
url	HTTP request's value
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.request	HTTP request cookie
cookies.response	HTTP response cookie
cookies.request.<<name>>	HTTP request cookie
cookies.response.<<name>>	HTTP response cookie
headers.request	All HTTP request headers
headers.response	All HTTP response headers
headers.request.<<name>>	HTTP request given header
headers.response.<<name>>	HTTP response given header
params	All HTTP request GET and POST parameters
params.get	All HTTP request GET parameters
params.post	All HTTP request POST parameters
params.get/post.<<name>>	A given HTTP request GET/POST parameter

URL field is broken in smaller parts using the urlparse Python's module, which 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
url.scheme	http
url.netloc	www.google.com
url.path	/dir/test.php
url.params	
url.query	id=1
url.fragment	
url.domain	google.com
url.ffname	test.php
url.fext	.php
url.fname	test
url.pstrip	Returns a hash of the request using the parameter's names without values (useful for unique operations)
url.hasquery	Returns true when the URL contains a query string.
url.ispath	Returns true when the URL path refers to a directory.
url.isbllist	Returns true when the URL file extension is included in the configuration discovery's blacklist

Payload instrospection 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.

Filtering results

The –filter command line parameter in conjuntion with the described filter language allows you to peform 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 instrospection 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
```

Filtering a payload

Slice

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

An example is shown below:

```
$ wfuzz-cli.py -z list,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 the one in the example above does not have a full FuzzResult object context and therefore object fields cannot be used.

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.

1.5.8 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 keywork 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"
```

- 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 specyfing 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.

wfpayload

If you do not want to perform any request, just find some specific HTTP request you can use the wfpayload executable.

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' and url.
↪pstrip|u()"
```

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

CHAPTER 2

Library Guide

2.1 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	dryrun=True
-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
–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.

2.2 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 webpage 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.

2.3 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"
...
>>> s.close()
```

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"
```

2.4 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 programatically without using Wfuzz payloads plugins.

Generating a new payload and start fuzzing is really simple:

```
>>> import wfuzz
>>> for r in wfuzz.get_payload(range(5)).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
>>> for r in wfuzz.get_payloads([range(5), ["a","b"]]).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"
```

(continues on next page)

(continued from previous page)

```
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"
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"
>>>
```

2.5 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 programatically from a string representing CLI options:

```
$ python
>>> import wfuzz
>>> for r in wfuzz.get_session("-z range,0-10 http://testphp.vulnweb.com/FUZZ").
↪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"
```