ZFuzz

Release 1.2

Oct 07, 2019

Contents

ZFuzz is an opensource web fuzzer written in Python (See Wfuzz for more advanced features)

Usage exemple:

```
$ ./zfuzz.py -w /mywordlist -u https://example.com/^FUZZ^ --sc 200
     ___
 ___|  _|_ _ ___ ___
|- _|  _| | |- _|- _|
|___|_| |___|___|___| v1.2

[TARGET] https://example.com/<fuzz>

[27:58] [200]: admin
[27:58] [200]: robots.txt
[27:58] [200]: js
[27:58] [200]: css
[27:59] [200]: cgi-bin
[27:59] [200]: about
[28:00] [200]: accounts

[28:00] Total time: 3s
```

Features

- Multithreaded
- Allows fuzzing of HTTP headers, POST data, cookies, and different parts of URL
- Very simple architecture/code so you can easily contribute to the project
- Easy to use and a nice interface

ZFuzz needs Python v3 to work, and it must be run on Linux

CHAPTER 1

Documentation

1.1 Contributing

You can contribute to zfuzz depending on your python skills or your willingness to help as an user

1.1.1 Improve the documentation

You can improve this documentation by forking this repository, updating the contents and sending a pull request

1.1.2 Improve ZFuzz

If you know how to code in Python and have ideas to improve zfuzz or just ameliorate the code to make it better, you're very welcome to send a pull requests, just make sure that you respect theses rules:

- Keep the code clean
- Respect the PEP8 style
- Don't too change the project structure/codes
- That's all ;)

1.2 Installation

1.2.1 From source

You can easily install ZFuzz from Github by following these commands:

```
$ git clone https://github.com/z3pp/ZFuzz.git
$ cd ZFuzz
$ python setup.py install
```

1.2.2 Dependencies

ZFuzz only use colored for the colored output and requests for the HTTP requests

1.3 Basic Usage

1.3.1 ZFuzz Options

- -h/–help – Print the help banner
- -u/–url – URL to fuzz
- -w/–wordlist – wordlist
- -H/–headers – HTTP headers
- -d/–data – POST data
- -b/–cookies – Cookie to send for the requests
- -k/–keyword – Fuzzing keyword to use. Default ^FUZZ^
- -t/–threads – Number of threads. Default 35
- -s/–delay – Delay between requests
- –timeout – Requests timeout
- –hc/sc – HTTP Code(s) to hide/show
- –hs/ss – Response to hide/show with the given str

1.3.2 Fuzzing keyword

By default, the fuzzing keyword is ^FUZZ^ but you can change it by using the [-k/–keyword] option:

```
$ ./zfuzz.py -k #FUZZ# ...
```

To fuzz something, just add the ^FUZZ^ keyword in the options that you would like to fuzz, And zfuzz will replace this keyword by each word of the wordlist specified:

```
$ ./zfuzz.py -w /mywordlist -u https://example.com/^FUZZ^
$ ./zfuzz.py -w /mywordlist -u https://example.com/ -d "username=admin&password=^FUZZ^
↪"
$ ./zfuzz.py -w /mywordlist -u https://example.com/ -H "User-agent: ^FUZZ^" "Content-
↪Type: application/json"
$ ./zfuzz.py -w /mywordlist -u https://example.com/ -b cookie:^FUZZ^
```

1.3.3 Limiting requests

The fuzzer is multi-threaded and by default, has 35 threads, you can change this by using the [-t/–threads] option You also can specify a delay between the requests

- Safe mode (Sending requests each 0.2s):

```
$ ./zfuzz.py -w /mywordlist -u http://example.com/^FUZZ^ -t 1 --delay 0.2
```

1.3.4 Filters

You can easily filter the requests result with these filters:

Hide reponse

The following options can be used to hide certain HTTP responses

–hc (HTTP Code(s) to hide):

```
$ ./zfuzz.py -w /mywordlist -u http://example.com/^FUZZ^ --hc 500,404
```

–hs (Response to hide with the given str):

```
$ ./zfuzz.py -w /mywordlist -u http://example.com/^FUZZ^ --hs "home page"
```

Show reponse

The following options can be used to show certain HTTP responses

–sc (HTTP Code(s) to show):

```
$ ./zfuzz.py -w /mywordlist -u http://example.com/^FUZZ^ --sc 200,301
```

–hs (Response to show with the given str):

```
$ ./zfuzz.py -w /mywordlist -u http://example.com/^FUZZ^ --hs "home page"
```

1.4 How ZFuzz works

1.4.1 ZFuzz CLI

class zfuzz.cli.**ZFuzzCLI**

Handle zfuzz CLI

main(*argv*)

ZFuzz main method

Parameters **argv** – Command line arguments list

parse_args(*argv*)

ZFuzz Argument parser

Parameters **argv** – Command line arguments list

Returns Arguments parsed

print_banner()

Print the zfuzz banner

print_help()

Print the help banner

1.4.2 Argparse custom actions

class zfuzz.action.**DataAction**(*option_strings*, *dest*, *nargs=None*, *const=None*, *default=None*, *type=None*, *choices=None*, *required=False*, *help=None*, *metavar=None*)

Parse data values

class zfuzz.action.**DictAction**(*option_strings*, *dest*, *nargs=None*, *const=None*, *default=None*, *type=None*, *choices=None*, *required=False*, *help=None*, *metavar=None*)

Create a dict from an str

class zfuzz.action.**ListAction**(*option_strings*, *dest*, *nargs=None*, *const=None*, *default=None*, *type=None*, *choices=None*, *required=False*, *help=None*, *metavar=None*)

Convert items separated by commas to a list

class zfuzz.action.**RangeAction**(*mini*, *maxi*, **args*, ***kwargs*)

Check the range of an argument

class zfuzz.action.**UrlAction**(*option_strings*, *dest*, *nargs=None*, *const=None*, *default=None*, *type=None*, *choices=None*, *required=False*, *help=None*, *metavar=None*)

Check the format of an url

1.4.3 The Fuzzer

1.4.4 Utils

zfuzz.utils.**get_code_color**(*code*)

Return http code colors

Parameters **code** – HTTP Status code

Returns HTTP Code color

zfuzz.utils.**is_matching**(*code*, *hc*, *sc*, *content*, *hs*, *ss*)

Determinate if the given response match the given filters

Parameters

- **code** – HTTP Status code
- **hc** – HTTP Code(s) to hide
- **sc** – HTTP Code(s) to show
- **content** – Response content
- **hs** – Hide response with hs
- **ss** – Show response with ss

Returns True/False, depending of the filter

zfuzz.utils.**replace_kv_dict**(*d*, *keyword*, *string*)

Replace each key and value of a dict

Parameters

- **d** – The dict to replace
- **keyword** – The keyword to replace in the dict

- **`string`** – The string that will replace the keyword

Python Module Index

z

Index