BlackRed Documentation

Release 3.0.0

Juergen Edelbluth

January 19, 2016

Contents

1 How does it work? 3

2 Requirements 5

3 Jump Start 7

4 Usage 9
- 4.1 Settings and Defaults . . . 9
- 4.2 Example Usage . . . 9

5 API Documentation 13
- 5.1 blackred package . . . 13

6 Links 17

7 Indices and tables 19

Python Module Index 21

BlackRed is a dynamic blacklisting library using Redis as a fast and reliable storage backend.

CHAPTER 1

How does it work?

Example: A user tries to log on a system and fails because of bad credentials or an inactivated account. This failure can be recorded with BlackRed. After three failures within a certain time the account gets locked for an extended period of time. This limits brute force attacks. All time periods are configurable.

In a desktop application you would record the username in question with BlackRed. In a web environment, the requester's IP address would be the perfect.

In the redis database, two lists are kept: A watchlist that records the failures, and the blacklist that contains blocked items.

CHAPTER 2

Requirements

BlackRed runs only under Python 3.3, 3.4, 3.5 and PyPy3. There's no support for Python 2.

The only thing BlackRed needs is the redis package >= 2.10.

CHAPTER 3

Jump Start

Installation can be done with `pip install blackred`. Usage is as easy, here an example for a simple user login:

```
import blackred

def login(username, password, request_ip):
    br = blackred.BlackRed()
    if br.is_blocked(request_ip):
        return False
    if not authenticate(username, password):
        br.log_fail(request_ip)
        return False
    return True
```

More examples: Example Usage.

CHAPTER 4

Usage

4.1 Settings and Defaults

The default settings for new instances of *BlackRed* can be defined with the attributes of the *Settings* subclass.

For all available options, see the *Settings* documentation at the blackred API docs.

4.2 Example Usage

BlackRed features a very simple and intuitive API. Go to API documentation.

4.2.1 Simple Use

With everything in defaults (Host is `localhost`, port is `6379` and so on), you can use BlackRed right out of the box:

```
import blackred

def login(username, password, request_ip):
    br = blackred.BlackRed()
    if br.is_blocked(request_ip):
        return False
    if not authenticate(username, password):
        br.log_fail(request_ip)
        return False
    return True
```

The *BlackRed constructor* offers a wide range of on-the-fly settings for the new instance. To keep your code nice and clean when you use the same settings most of the time, you can use the *Settings Predefinition*.

4.2.2 Item Selection

For a web or network based application, a good item to check and block is the client IP address of the requestor.

In other environments, you might choose anything else that identifies a request source uniqually. Most important: The item must be represented by a `str`.

4.2.3 Settings Predefinition

To make the using of BlackRed easier, you can change the default settings for new instances by changing the values of the attributes of the *BlackRed.Settings* class. Changes are only valid for new instances created after the changes. Already existing instances are not changed.

Note: BlackRed versions before 0.3.0 had a different settings approach. Settings where separated into global settings and settings for new instances. With 0.3.0, all settings are only valid for new instances. Keep that in mind when upgrading from an older version.

Example: Enable *Anonymization* by default:

```
import blackred

blackred.BlackRed.Settings.ANONYMIZATION = True

def login(username, password, request_ip):
    br = blackred.BlackRed()
    if br.is_blocked(request_ip):
        return False
    if not authenticate(username, password):
        br.log_fail(request_ip)
        return False
    return True
```

4.2.4 Use Unix Socket instead of TCP/IP connection

If you want to use a unix socket to connect, set the `redis_use_socket` constructor parameter to `True` and provide the absolute path to the socket with the `redis_host` parameter, example:

```
import blackred

br = blackred.BlackRed(redis_host='/var/run/redis/redis.sock', redis_use_socket=True)
```

As always, those settings can be predefined (see *Settings Predefinition*) as defaults for all new instances:

```
import blackred

blackred.BlackRed.Settings.REDIS_USE_SOCKET = True
blackred.BlackRed.Settings.REDIS_HOST = '/var/run/redis/redis.sock'
```

4.2.5 Anonymization

Sometimes it's necessary to hash the item's values to ensure privacy of the requester. BlackRed can easily support you. Just set the *ANONYMIZATION* attribute to `True`:

```
import blackred

blackred.BlackRed.Settings.ANONYMIZATION = True
```

4.2.6 Use a Redis password (aka Redis AUTH feature)

To activate authentication, you can use the `redis_auth` parameter of the constructor:

```
import blackred

br = blackred.BlackRed(redis_auth='my_password')
```

As always, this settings can be predefined (see *Settings Predefinition*) as default for all new instances:

```
import blackred

blackred.BlackRed.Settings.REDIS_AUTH = 'my_password'
```

CHAPTER 5

API Documentation

5.1 blackred package

The complete *blackred.blackred* module is imported in the `__init__.py` of the `blackred` module. So you can use `blackred` instead of `blackred.blackred` in your code.

For usage examples, please see the Examples Page.

5.1.1 blackred.blackred module

class `blackred.blackred.`**`BlackRed`**(*redis_host: str=None*, *redis_port: int=None*, *redis_db: int=None*, *redis_use_socket: bool=None*, *redis_auth: str=None*)

Bases: `object`

Create an Instance of BlackRed with configuration for Redis connection

Parameters

- **`redis_host`** (*str*) – Hostname, IP Address or Socket
- **`redis_port`** (*int*) – Port Number
- **`redis_db`** (*int*) – DB Number
- **`redis_use_socket`** (*bool*) – True, when a socket should be used instead the TCP/IP connection
- **`redis_auth`** (*str*) – If set, the redis AUTH command will be used with the given password

class `Settings`

Bases: `object`

All the attributes of this class are used as default settings for new BlackRed instances.

ANONYMIZATION = False
If required, you can turn this on (set it to `True`) when BlackRed should not store IP addresses, usernames etc. as plain text. A hash value is used instead.

This setting might be necessary to get data protection policy compliant.
Type bool

BLACKLIST_KEY_TEMPLATE = 'BlackRed:BlackList:{:s}'
Template for generating blacklist entries.

Defaults to `'BlackRed:BlackList:{:s}'` and should only be changed when this collides with one of your namespaces.
Type str

BLACKLIST_REFRESH_TTL_ON_HIT = True
If an item is already on the blacklist and is checked with *BlackRed.is_blocked* or *BlackRed.is_not_blocked* while on the blacklist, the time to live for the blacklist entry is reset to *BlackRed.Settings.BLACKLIST_TTL_SECONDS*.

So if this is set to `True` (that's the default value) and a blocked user tries to login after 12 hours after blacklisting, his blacklist time is increased to another 24 hours.
Type bool

BLACKLIST_TTL_SECONDS = 86400
Time in seconds for an item to stay on the blacklist.

Defaults to 86400 (24 hours). After that time, the entry is deleted automatically.
Type int

REDIS_AUTH = None
If the redis connection requires authentication, the password can be predefined here.

If set to `None` (which is the default), authentication will not be used.
Type str

REDIS_DB = 0
The Redis database number, defaults to `0`
Type int

REDIS_HOST = 'localhost'
Hostname, IP Address or socket, defaults to `'localhost'`
Type str

REDIS_PORT = 6379
TCP-Port for Redis, defaults to `6379`
Type int

REDIS_USE_SOCKET = False
Tell the `BlackRed` class to use a unix socket instead of a TCP/IP connection.

Defaults to `False`
Type bool

SALT_KEY = 'BlackRed:AnonymizationListSecret'
The key for saving the individual salt for anonymization.

Defaults to `BlackRed:AnonymizationListSecret` and should only be changed when it collides with one of your namespaces.
Type str

WATCHLIST_KEY_TEMPLATE = 'BlackRed:WatchList:{:s}'
Template for generating watchlist entries.

Defaults to `'BlackRed:WatchList:{:s}'` and should only be changed when this collides with one of your namespaces.

Type str

WATCHLIST_TO_BLACKLIST_THRESHOLD = 3

Number of fails for an item to get from the watchlist to the blacklist. Defaults to 3.

Type int

WATCHLIST_TTL_SECONDS = 180

Time in seconds for an item to stay on the watchlist.

If after a logged failure a new failure appears within this time frame, the fail count is increased. If not, the entry is deleted automatically. Defaults to 180 (3 minutes).

Type int

BlackRed.**get_blacklist_ttl**(*item: str*) → int

Get the amount of time a specific item will remain on the blacklist.

Parameters **item** (*str*) – The item to get the TTL for on the blacklist

Returns Time in seconds. Returns None for a non-existing element.

Return type int

BlackRed.**get_watchlist_ttl**(*item: str*) → int

Get the amount of time a specific item will remain on the watchlist.

Parameters **item** (*str*) – The item to get the TTL for on the watchlist

Returns Time in seconds. Returns None for a non-existing element

Return type int

BlackRed.**is_blocked**(*item: str*) → bool

Check if an item is on the blacklist

Parameters **item** (*str*) – The item to check

Returns True, when the item is on the blacklist

Return type bool

BlackRed.**is_not_blocked**(*item: str*) → bool

Check if an item is _not_ already on the blacklist

Parameters **item** (*str*) – The item to check

Returns True, when the item is _not_ on the blacklist

Return type bool

BlackRed.**log_fail**(*item: str*) → None

Log a failed action for an item. If the fail count for this item reaches the threshold, the item is moved to the blacklist.

Parameters **item** (*str*) – The item to log

BlackRed.**unblock**(*item: str*) → None

Unblock an item and/or reset it's fail count

Parameters **item** (*str*) – The item to unblock

blackred.blackred.**create_salt**(*length: int=128*) → bytes

Create a new salt

Parameters **length** (*int*) – How many bytes should the salt be long?

Returns The salt

Return type bytes

CHAPTER 6

Links

- Author: Juergen Edelbluth, https://juergen.rocks/, @JuergenRocks
- Build Status: https://travis-ci.org/edelbluth/blackred
- Project Homepage: https://github.com/edelbluth/blackred
- PyPi Page: https://pypi.python.org/pypi/blackred
- German Description (for 0.2 version): https://juergen.rocks/art/mit-blackred-benutzer-logins-absichern.html
- Documentation (this one): https://blackred.readthedocs.org/index.html

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

b

blackred.blackred, 13

A

ANONYMIZATION (blackred.blackred.BlackRed.Settings attribute), 13

B

BLACKLIST_KEY_TEMPLATE (blackred.blackred.BlackRed.Settings attribute), 14
BLACKLIST_REFRESH_TTL_ON_HIT (blackred.blackred.BlackRed.Settings attribute), 14
BLACKLIST_TTL_SECONDS (blackred.blackred.BlackRed.Settings attribute), 14
BlackRed (class in blackred.blackred), 13
blackred.blackred (module), 13
BlackRed.Settings (class in blackred.blackred), 13

C

create_salt() (in module blackred.blackred), 15

G

get_blacklist_ttl() (blackred.blackred.BlackRed method), 15
get_watchlist_ttl() (blackred.blackred.BlackRed method), 15

I

is_blocked() (blackred.blackred.BlackRed method), 15
is_not_blocked() (blackred.blackred.BlackRed method), 15

L

log_fail() (blackred.blackred.BlackRed method), 15

R

REDIS_AUTH (blackred.blackred.BlackRed.Settings attribute), 14
REDIS_DB (blackred.blackred.BlackRed.Settings attribute), 14
REDIS_HOST (blackred.blackred.BlackRed.Settings attribute), 14
REDIS_PORT (blackred.blackred.BlackRed.Settings attribute), 14
REDIS_USE_SOCKET (blackred.blackred.BlackRed.Settings attribute), 14

S

SALT_KEY (blackred.blackred.BlackRed.Settings attribute), 14

U

unblock() (blackred.blackred.BlackRed method), 15

W

WATCHLIST_KEY_TEMPLATE (blackred.blackred.BlackRed.Settings attribute), 14
WATCHLIST_TO_BLACKLIST_THRESHOLD (blackred.blackred.BlackRed.Settings attribute), 15
WATCHLIST_TTL_SECONDS (blackred.blackred.BlackRed.Settings attribute), 15