Maigret

Release 0.4.4

soxoj

Sep 09, 2023

SECTIONS

1 You may be interested in: 3

1.1 Command line options . . . 3
1.2 Extracting information from pages . . . 5
1.3 Features . . . 6
1.4 Philosophy . . . 7
1.5 Roadmap . . . 8
1.6 Supported identifier types . . . 8
1.7 Tags . . . 9
1.8 Usage examples . . . 9
1.9 Settings . . . 10
1.10 Development . . . 10

Maigret is an easy-to-use and powerful OSINT tool for collecting a dossier on a person by username only.

This is achieved by checking for accounts on a huge number of sites and gathering all the available information from web pages.

The project's main goal - give to OSINT researchers and pentesters a **universal tool** to get maximum information about a subject and integrate it with other tools in automatization pipelines.

CHAPTER

ONE

YOU MAY BE INTERESTED IN:

- *Command line options description* and *usage examples*
- *Features list*
- *Project roadmap*

1.1 Command line options

1.1.1 Usernames

```
maigret username1 username2 ...
```

You can specify several usernames separated by space. Usernames are **not** mandatory as there are other operations modes (see below).

1.1.2 Parsing of account pages and online documents

```
maigret --parse URL
```

Maigret will try to extract information about the document/account owner (including username and other ids) and will make a search by the extracted username and ids. *Examples*.

1.1.3 Main options

Options are also configurable through settings files, see *settings section*.

`--tags` - Filter sites for searching by tags: sites categories and two-letter country codes (**not a language!**). E.g. photo, dating, sport; jp, us, global. Multiple tags can be associated with one site. **Warning: tags markup is not stable now.**

`-n`, `--max-connections` - Allowed number of concurrent connections **(default: 100)**.

`-a`, `--all-sites` - Use all sites for scan **(default: top 500)**.

`--top-sites` - Count of sites for scan ranked by Alexa Top **(default: top 500)**.

`--timeout` - Time (in seconds) to wait for responses from sites **(default: 30)**. A longer timeout will be more likely to get results from slow sites. On the other hand, this may cause a long delay to gather all results. The choice of the right timeout should be carried out taking into account the bandwidth of the Internet connection.

`--cookies-jar-file` - File with custom cookies in Netscape format (aka cookies.txt). You can install an extension to your browser to download own cookies (Chrome, Firefox).

`--no-recursion` - Disable parsing pages for other usernames and recursive search by them.

`--use-disabled-sites` - Use disabled sites to search (may cause many false positives).

`--id-type` - Specify identifier(s) type (default: username). Supported types: gaia_id, vk_id, yandex_public_id, ok_id, wikimapia_uid. Currently, you must add `-a` flag to run a scan on sites with custom id types, sites will be filtered automatically.

`--ignore-ids` - Do not make search by the specified username or other ids. Useful for repeated scanning with found known irrelevant usernames.

`--db` - Load Maigret database from a JSON file or an online, valid, JSON file.

`--retries RETRIES` - Count of attempts to restart temporarily failed requests.

1.1.4 Reports

`-P`, `--pdf` - Generate a PDF report (general report on all usernames).

`-H`, `--html` - Generate an HTML report file (general report on all usernames).

`-X`, `--xmind` - Generate an XMind 8 mindmap (one report per username).

`-C`, `--csv` - Generate a CSV report (one report per username).

`-T`, `--txt` - Generate a TXT report (one report per username).

`-J`, `--json` - Generate a JSON report of specific type: simple, ndjson (one report per username). E.g. `--json ndjson`

`-fo`, `--folderoutput` - Results will be saved to this folder, `results` by default. Will be created if doesn't exist.

1.1.5 Output options

`-v`, `--verbose` - Display extra information and metrics. *(loglevel=WARNING)*

`-vv`, `--info` - Display service information. *(loglevel=INFO)*

`-vvv`, `--debug`, `-d` - Display debugging information and site responses. *(loglevel=DEBUG)*

`--print-not-found` - Print sites where the username was not found.

`--print-errors` - Print errors messages: connection, captcha, site country ban, etc.

1.1.6 Other operations modes

`--version` - Display version information and dependencies.

`--self-check` - Do self-checking for sites and database and disable non-working ones **for current search session** by default. It's useful for testing new internet connection (it depends on provider/hosting on which sites there will be censorship stub or captcha display). After checking Maigret asks if you want to save updates, answering y/Y will rewrite the local database.

`--submit URL` - Do an automatic analysis of the given account URL or site main page URL to determine the site engine and methods to check account presence. After checking Maigret asks if you want to add the site, answering y/Y will rewrite the local database.

1.2 Extracting information from pages

Maigret can parse URLs and content of web pages by URLs to extract info about account owner and other meta information.

You must specify the URL with the option `--parse`, it's can be a link to an account or an online document. List of supported sites see here.

After the end of the parsing phase, Maigret will start the search phase by *supported identifiers* found (usernames, ids, etc.).

1.2.1 Examples

```
$ maigret --parse https://docs.google.com/spreadsheets/d/
↪1HtZKMLRXNsZ0HjtBmo0Gi03nUPiJIA4CC4jTYbCAnXw/edit\#gid\=0

Scanning webpage by URL https://docs.google.com/spreadsheets/d/
↪1HtZKMLRXNsZ0HjtBmo0Gi03nUPiJIA4CC4jTYbCAnXw/edit#gid=0...
org_name: Gooten
mime_type: application/vnd.google-apps.ritz
Scanning webpage by URL https://clients6.google.com/drive/v2beta/files/
↪1HtZKMLRXNsZ0HjtBmo0Gi03nUPiJIA4CC4jTYbCAnXw?fields=alternateLink
↪%2CcopyRequiresWriterPermission%2CcreatedDate%2Cdescription%2CdriveId%2CfileSize
↪%2CiconLink%2Cid%2Clabels(starred%2C%20trashed)%2ClastViewedByMeDate%2CmodifiedDate
↪%2Cshared%2CteamDriveId%2CuserPermission(id%2Cname%2CemailAddress%2Cdomain%2Crole
↪%2CadditionalRoles%2CphotoLink%2Ctype%2CwithLink)%2Cpermissions(id%2Cname
↪%2CemailAddress%2Cdomain%2Crole%2CadditionalRoles%2CphotoLink%2Ctype%2CwithLink)
↪%2Cparents(id)%2Ccapabilities(canMoveItemWithinDrive%2CcanMoveItemOutOfDrive
↪%2CcanMoveItemOutOfTeamDrive%2CcanAddChildren%2CcanEdit%2CcanDownload%2CcanComment
↪%2CcanMoveChildrenWithinDrive%2CcanRename%2CcanRemoveChildren
↪%2CcanMoveItemIntoTeamDrive)%2Ckind&supportsTeamDrives=true&enforceSingleParent=true&
↪key=AIzaSyC1eQ1xj69IdTMeii5r7brs3R90eck-m7k...
created_at: 2016-02-16T18:51:52.021Z
updated_at: 2019-10-23T17:15:47.157Z
gaia_id: 15696155517366416778
fullname: Nadia Burgess
email: nadia@gooten.com
image: https://lh3.googleusercontent.com/a-/
↪AOh14GheZe1CyNa3NeJInWAl70qkip4oJ7qLsD8vDy6X=s64
email_username: nadia
```

```
$ maigret.py --parse https://steamcommunity.com/profiles/76561199113454789
Scanning webpage by URL https://steamcommunity.com/profiles/76561199113454789...
steam_id: 76561199113454789
nickname: Pok
username: Machine42
```

1.3 Features

This is the list of Maigret features.

1.3.1 Personal info gathering

Maigret does the parsing of accounts webpages and extraction of personal info, links to other profiles, etc. Extracted info displayed as an additional result in CLI output and as tables in HTML and PDF reports. Also, Maigret use found ids and usernames from links to start a recursive search.

Enabled by default, can be disabled with `--no extracting`.

1.3.2 Recursive search

Maigret can extract some *common ids* and usernames from links on the account page (often people placed links to their other accounts) and immediately start new searches. All the gathered information will be displayed in CLI output and reports.

Enabled by default, can be disabled with `--no-recursion`.

1.3.3 Reports

Maigret currently supports HTML, PDF, TXT, XMind 8 mindmap, and JSON reports.

HTML/PDF reports contain:

- profile photo
- all the gathered personal info
- additional information about supposed personal data (full name, gender, location), resulting from statistics of all found accounts

Also, there is a short text report in the CLI output after the end of a searching phase.

Warning: XMind 8 mindmaps are incompatible with XMind 2022!

1.3.4 Tags

The Maigret sites database very big (and will be bigger), and it is maybe an overhead to run a search for all the sites. Also, it is often hard to understand, what sites more interesting for us in the case of a certain person.

Tags markup allows selecting a subset of sites by interests (photo, messaging, finance, etc.) or by country. Tags of found accounts grouped and displayed in the reports.

See full description *in the Tags Wiki page*.

1.3.5 Censorship and captcha detection

Maigret can detect common errors such as censorship stub pages, CloudFlare captcha pages, and others. If you get more them 3% errors of a certain type in a session, you've got a warning message in the CLI output with recommendations to improve performance and avoid problems.

1.3.6 Retries

Maigret will do retries of the requests with temporary errors got (connection failures, proxy errors, etc.).

One attempt by default, can be changed with option `--retries N`.

1.3.7 Archives and mirrors checking

The Maigret database contains not only the original websites, but also mirrors, archives, and aggregators. For example:

- Reddit BigData search
- Picuki, Instagram mirror
- Twitter shadowban checker

It allows getting additional info about the person and checking the existence of the account even if the main site is unavailable (bot protection, captcha, etc.)

1.3.8 Simple API

Maigret can be easily integrated with the use of Python package maigret.

Example: the official Telegram bot

1.4 Philosophy

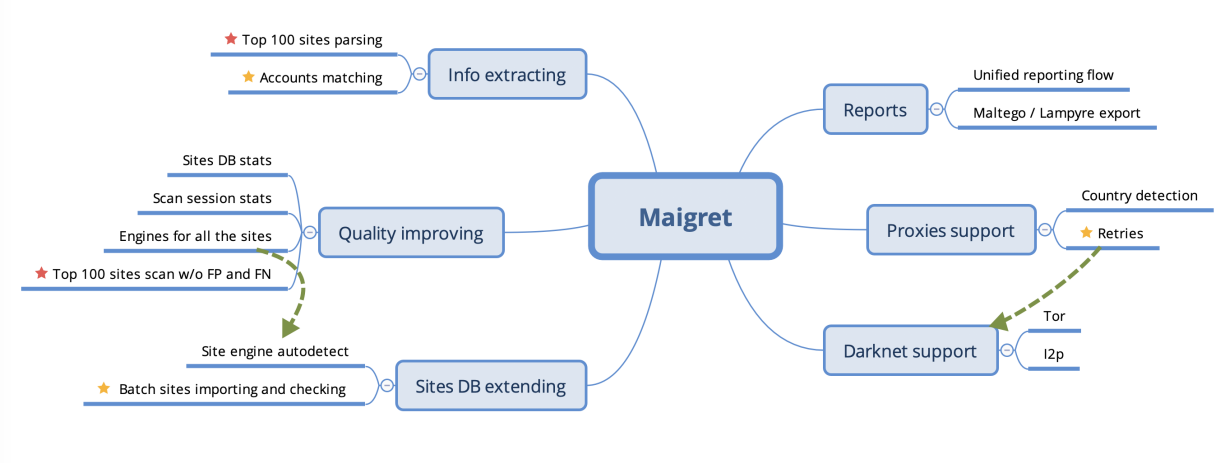
TL;DR: Username => Dossier

Maigret is designed to gather all the available information about person by his username.

What kind of information is this? First, links to person accounts. Secondly, all the machine-extractable pieces of info, such as: other usernames, full name, URLs to people's images, birthday, location (country, city, etc.), gender.

All this information forms some dossier, but it also useful for other tools and analytical purposes. Each collected piece of data has a label of a certain format (for example, `follower_count` for the number of subscribers or `created_at` for account creation time) so that it can be parsed and analyzed by various systems and stored in databases.

1.5 Roadmap



1.5.1 Current status

- Sites DB stats - ok
- Scan sessions stats - ok
- Site engine autodetect - ok
- Engines for all the sites - WIP
- Unified reporting flow - ok
- Retries - ok

1.6 Supported identifier types

Maigret can search against not only ordinary usernames, but also through certain common identifiers. There is a list of all currently supported identifiers.

- **gaia_id** - Google inner numeric user identifier, in former times was placed in a Google Plus account URL.
- **steam_id** - Steam inner numeric user identifier.
- **wikimapia_uid** - Wikimapia.org inner numeric user identifier.
- **uidme_uguid** - uID.me inner numeric user identifier.
- **yandex_public_id** - Yandex sites inner letter user identifier. See also: YaSeeker.
- **vk_id** - VK.com inner numeric user identifier.
- **ok_id** - OK.ru inner numeric user identifier.
- **yelp_userid** - Yelp inner user identifier.

1.7 Tags

The use of tags allows you to select a subset of the sites from big Maigret DB for search.

Warning: tags markup is not stable now.

There are several types of tags:

1. **Country codes**: `us`, `jp`, `br`... (ISO 3166-1 alpha-2). These tags reflect the site language and regional origin of its users and are then used to locate the owner of a username. If the regional origin is difficult to establish or a site is positioned as worldwide, *no country code is given*. There could be multiple country code tags for one site.
2. **Site engines**. Most of them are forum engines now: `uCoz`, `vBulletin`, `XenForo` et al. Full list of engines stored in the Maigret database.
3. **Sites' subject/type and interests of its users**. Full list of "standard" tags is present in the source code only for a moment.

1.7.1 Usage

`--tags us,jp` – search on US and Japanese sites (actually marked as such in the Maigret database)

`--tags coding` – search on sites related to software development.

`--tags ucoz` – search on uCoz sites only (mostly CIS countries)

1.8 Usage examples

Start a search for accounts with username `machine42` on top 500 sites from the Maigret DB.

```
maigret machine42
```

Start a search for accounts with username `machine42` on **all sites** from the Maigret DB.

```
maigret machine42 -a
```

Start a search [...] and generate HTML and PDF reports.

```
maigret machine42 -a -HP
```

Start a search for accounts with username `machine42` only on Facebook.

```
maigret machine42 --site Facebook
```

Extract information from the Steam page by URL and start a search for accounts with found username `machine42`.

```
maigret --parse https://steamcommunity.com/profiles/76561199113454789
```

Start a search for accounts with username `machine42` only on US and Japanese sites.

```
maigret machine42 --tags en,jp
```

Start a search for accounts with username `machine42` only on sites related to software development.

```
maigret machine42 --tags coding
```

Start a search for accounts with username `machine42` on uCoz sites only (mostly CIS countries).

```
maigret machine42 --tags ucoz
```

1.9 Settings

Options are also configurable through settings files. See settings JSON file for the list of currently supported options.

After start Maigret tries to load configuration from the following sources in exactly the same order:

```
# relative path, based on installed package path
resources/settings.json

# absolute path, configuration file in home directory
~/.maigret/settings.json

# relative path, based on current working directory
settings.json
```

Missing any of these files is not an error. If the next settings file contains already known option, this option will be rewrited. So it is possible to make custom configuration for different users and directories.

1.10 Development

1.10.1 Testing

It is recommended use Python 3.7/3.8 for test due to some conflicts in 3.9.

Install test requirements:

```
pip install -r test-requirements.txt
```

Use the following commands to check Maigret:

```
# run linter and typing checks
# order of checks%
# - critical syntax errors or undefined names
# - flake checks
# - mypy checks
make lint

# run testing with coverage html report
# current test coverage is 60%
make text

# open html report
open htmlcov/index.html
```

1.10.2 How to publish new version of Maigret

Collaborats rights are requires, write Soxoj to get them.

For new version publishing you must create a new branch in repository with a bumped version number and actual changelog first. After it you must create a release, and GitHub action automatically create a new PyPi package.

- New branch example: https://github.com/soxoj/maigret/commit/e520418f6a25d7edacde2d73b41a8ae7c80ddf39
- Release example: https://github.com/soxoj/maigret/releases/tag/v0.4.1

1. Make a new branch locally with a new version name. Check the current version number here: https://pypi.org/project/maigret/. **Increase only patch version (third number)** if there are no breaking changes.

```
git checkout -b 0.4.0
```

2. Update Maigret version in three files manually:
   - setup.py
   - maigret/__version__.py
   - docs/source/conf.py
3. Create a new empty text section in the beginning of the file *CHANGELOG.md* with a current date:

```
## [0.4.0] - 2022-01-03
```

4. Get auto-generate release notes:
   - Open https://github.com/soxoj/maigret/releases/new
   - Click *Choose a tag*, enter *v0.4.0* (your version)
   - Click *Create new tag*
   - Press + *Auto-generate release notes*
   - Copy all the text from description text field below
   - Paste it to empty text section in *CHANGELOG.txt*
   - Remove redundant lines *## What's Changed* and *## New Contributors* section if it exists
   - *Close the new release page*
5. Commit all the changes, push, make pull request

```
git add -p
git commit -m 'Bump to YOUR VERSION'
git push origin head
```

6. Merge pull request
7. Create new release
   - Open https://github.com/soxoj/maigret/releases/new again
   - Click *Choose a tag*
   - Enter actual version in format *v0.4.0*
   - Also enter actual version in the field *Release title*
   - Click *Create new tag*

- Press + *Auto-generate release notes*
- **Press "Publish release" button**

8. That's all, now you can simply wait push to PyPi. You can monitor it in Action page: https://github.com/soxoj/maigret/actions/workflows/python-publish.yml