User Manual

December 8, 2021
proSECO

1 Introduction

This is the User Manual for the Controller program (`https://github.com/SecureSECO/SearchSECOController`). This manual assumes the software is installed as explained in the readme file of the GitHub repository.

2 Usage

A command can be run from the commandline using:

```
searchseco command [flag <argument>?]*
```

Or from Visual Studio using:

```
command [flag <argument>?]*
```

The implemented commands are described in section 5. To use the program to its fullest, be sure to also read about section 6.

3 Config file

The `config.txt` file located at `cfg/config.txt` contains the default values used for the flags. There are some extra flags that can be set in the config file that can not be set in the command line: `github_username` and `github_token`. These are used by the Crawler to get project meta data and to crawl urls. A token can be generated on the github site by going to your user settings `->` Developer settings `->` personal access tokens. In this file you can also set the name of the worker node for better statistical analysis.
Comments can be written in the config file by starting the line with a `#`. A flag can be set in the config file by starting with the long hand version of the flag, followed by a colon `:` and then the default value you want to give it.

4 Environment file .env

The .env file should contain the IP and ports of all servers you want to connect to. This file will be read and used to determine where requests have to be send to. This file should contain a API_IPS entry. This entry should be in the format `API_IPS=ip1?port1,ip2?port2,ip3?port3`. Each ip port combination should start with the ip of the server, followed by a question mark. After that should be the port. Each ip port combination should be separated by a comma.

5 Commands

start

`searchseco start [flags]`

Starts a worker node. This worker node will contribute with finding projects, processing them and sending the found information to the database. The user does not need to provide any input after starting this command. If the user wants the process to stop, they can type `stop` in the command line. The program will finish the project it is currently working on and terminate when it is done.

check

`searchseco check <url> [flags]`

Checks for occurrences of code from the given url in the database, after which the found results will be displayed.

upload

`searchseco upload <url> [flags]`

Uploads code from the given url to the database, including relevant metadata.

checkupload

`searchseco checkupload <url> [flags]`

Executes the check and upload command as specified above.

6 Flags

All flags can be passed after calling `searchseco command` to change the way the programme behaves. Individual flags can be run without a command. All flags have a short and a long version. The short version is passed with a single dash followed by a single character (e.g. `searchseco command -v`). The long version is passed with two dashes followed by the name of the flag `searchseco command --version`. Some flags require an argument, this is passed directly after the flag tag (e.g. `searchseco command -- branch master`).

6.1 Individual Flags

`-v`
`-version`

Displays the version of the SearchSECO system, and the version of each of its subsystems. If this flag is used with a command, the command will be ignored and the version will be printed.

`-h`
`-help`

Displays the full help message. If this flag is used with a command, the help message for that specific command will be displayed.

6.2 Additional Flags

`-V`
`-verbose`

Use this flag to set the verbosity level of the console output. It can be used in combination with any command. Allowed values are 1 (Nothing), 2 (Errors), 3 (Warnings & Errors), 4 (Everything except debug), 5 (Debug).
Default value: 4 (Everything excep debug).

`-c`
`--cpu`

The amount of CPU cores to make available to the worker node.
Minimum value: 2
Default value: half the amount of cores available on the machine.

`-l`
`--lines`

For the uploading of vulnerabilities the lines to upload. Takes an argument in the form of `file1:line1,line2?file2:line3,line4,line5`, where the files are the file names including the location in the project with the lines in that file to upload.

`-C`
`--code`

For the uploading of vulnerabilities the vulnerability code to add to the methods.

`-p`
`--commit`

For uploading vulnerabilities and checking a project the commit to use.

7 Errors

7.1 Ranges

For overview of the user errors are put in ranges per submodule as follows.
E001-E099 - Controller Errors
E100-E199 - Crawler Errors
E200-E299 - Spider Errors
E300-E399 - Parser Errors
E400-E499 - Database API response Errors.

7.2 Error List

E001 - The specified flag does not exist.
E002 - The specified flag does not exist in the config file.
E003 - The specified argument is invalid for its corresponding flag.
E004 - The specified argument is invalid for its corresponding flag in the config file.
E005 - A flag was entered with an unexpected amount of arguments.
E006 - GitHub authentication data is missing in the config file.
E007 - A command got an unexpected amount of arguments.
E008 - No command was entered.
E009 - The entered command does not exist.
E010 - The call as a whole as it was entered in the command line could not be parsed.
E011 - A flag was incorrectly entered as if it were a full-length flag, while it actually is a shorthand flag (e.g. `--c` instead of `-c`).
E012 - A flag was incorrectly entered as if it were a shorthand flag, while it actually is a full-length flag (e.g. `-cpu` instead of `--cpu`).
E013 - A flag could not be parsed at all (e.g. too many leading hyphens).
E014 - The URL parameter was invalid.
E015 - Controller terminated after Crawler threw a fatal error.
E016 - Controller terminated after Spider threw a fatal error.
E017 - Controller terminated after Parser threw a fatal error.
E018 - No .env file was found in the `SearchSECOController` folder.
E019 - The .env does not contain data to connect to a Database API instance.
E020 - **(Debug only)** An unimplemented function was called.
E101 - GitHub responded with a JSON error.
E102 - No valid GitHub credentials were provided.
E103 - The rate limit of the used GitHub token was exceeded.
E104 - You do not have access to the specified GitHub repository.
E105 - GitHub responded with a Bad Gateway error.
E106 - The specified url was not found.
E107 - GitHub responded with an unknown error.
E108 - Could not index on the given key (the key does not exist in the JSON variable).
E109 - Parsing the given string as a JSON variable went wrong.
E110 - A type (conversion) error occurred in a JSON variable.
E111 - A field was empty in a JSON variable while trying to get from that field. Difference between this and **E108** is that this error is thrown when using a `get()` function.
E112 - Out of range error while trying to index on a key.
E200 - (not in use) - Git cloning failed.
E201 - Opening a stream to read from or write to the command prompt failed.
E202 - (not in use) - A file extension file was not found.
E400 - No connection could be made with a Database API.
E401 - The Database API received a bad request.
E402 - Something went wrong in the Database API.
E403 - The Database API responded in an unexpected way: the HTTP statuscode was not recognised.
E404 - The database gave an invalid Job.