Documentation

Overview

The aim of this project is to gather files from Github, which were modified after a SonarQube's scanner highlighted an issue in them. The scripts are gathering files by using graphql and rest api provided by Github. Each step will store a csv file with the temporary results. Results will be stored in results/(folder)

Note: Rate limits might have changed since the creation of the project.

Requirements

- Python
    - pandas, javalang, unidecode, requests, pygithub
- SonarQube

Main steps

1. Instantiate TruePositiveSquidSearcher(user, password, token, date, word, patterns, datetimedate=(2020, 8, 27))
    - user, password, token – github related credentials
    - data, dictionary that specifies the start_year, end_year, start_month, end_month, e.g.:
    - word, search word, e.g.: squid
    - patterns, regular expressions: r'(squid:s\d+)', r'(squid:\w+)', r'(squid%3aS\d+)', r'(squid%3A\w+)'
    - datetimedate, used for folder creation
2. First search for pull requests:
    - tpss.gather_pull_requests()
    - this step is using GraphQL
3. Filter the results with the given regular expressions:

- tpss.filter_pull_requests()

4. Collect the commits from the pull requests:
    - tpss.gather_commits_from_pr()
    - this step is using GraphQL
5. Filter the commits with the given patterns:
    - tpss.filter_commits()
6. Collect files:
    - tpss.gather_files()
    - this step is using REST API
7. Search the location of the fix:
    - tpss.search_sonar_locations(),
    - this is using the downloaded files and patches
8. Check the downloaded files for parsing errors:
    - tpss.check_java_files()
9. Create a list of the usable files:
    - tpss.load_good_files_list()
10. Prepare project names with files for sonar scanner:
    - tpss.create_project_names_and_good_files_chunks(project_base='0827', size=10000)
    - project_base, project name
    - size, maximum file number for each project
11. Instantiate sonar scanner for multiple projects:
    - ms = MultiSonarScanner(token=sonar_token, projects_and_chunks=tpss.projects_and_chunks, lang=tpss.lang, good_files_csv=tpss.good_files_csv, log_file=tpss.log_txt)
    - token, can be obtained through sonar qube's administration panel
12. Run sonar scanner:

- ms.bulk_sonar_scan()
- sonar-project.propertios files will be automatically created

13. Get the column names of the created temporary csv files:
    - keys = KeyHolder()
14. Get project names and keys:
    - projects = {k + '_Key': v for k, v in tpss.projects_and_chunks.items()}
15. Instantiate ProjectScanner, this will search the files at the squid patch locations:
    - ps = ProjectScanner(token=sonar_token, projects_and_chunks=projects, patch_checked_csv=tpss.patch_checked_csv, patch_located_keys=keys.get_patched_keys(), result_keys=keys.get_project_scan_res_keys(), good_files_csv_path=tpss.good_files_csv, result_folder=tpss.squid_folder, result_file_base_name='0827_java')
16. Search for squids:
    - ps.search_projects_issues()

Notes on the content of the project

- main.py, Run the program using main.py
- binaries, necessary for sonar scanner
- check_files, contains scripts to check if the downloaded file contains parsing issues
- gather, contains script related for the main process
- repo_tools, contains scripts that are related to github
- results, temporary and final results will be stored here
- sonar_qube_api, contains script related to SonarQube
- sonar_tools, contains sonar related tools used by main process
- tp_utils, utils related to true positive search