ViChecker User Manual

ViChecker aims to find security vulnerabilities in specific open source projects published on git by querying gpt for the commit history within a desired period.

Before running this code, you should install some Python libraries and write two text files.

Command	pip install os pip install git pip install json pip install subprocess pip install openai pip install time pip install pandas pip install nbimporter
api_key.txt	Write down your own gpt key
projectList.txt	Write down the repository links you want to check

This tool consists of five steps.

First, read the list of desired open source projects from projectList.txt and clone them locally. The git repository path that can be cloned must be stored in the projectList file. When executing the following command, it is cloned into the current directory, and if it has already been cloned, it is fetched with the latest commit. Gitignore the cloned directories.

Command	python _1_git_repo_cloner.py
Ressult	Cloned open source project, modified gitignore file

Second, run logger.sh and save the commit logs of the cloned repository in "{repository_name}-log.json". Sometimes the json format is broken, and in this case, you need to open the file directly and remove the cause of the breakage. Add a list of modified files to the saved log. The filter for selecting the modified file to add includes whether it is a java file or a 'test' file. Modify if necessary.

Command	python _2_git_commit_logger.py
Ressult	Create commit-logs/{repository_name}-log.json files

Third, using the commit log, you can load three files at each commit point: a diff file, a modified java file, and a java file before modification.

Command	python _3_git_file_tracker
Ressult	commit-files/{index}_after_{file-path}.java, commit-files/{index}_before_{file-path}.java, commit-files/{index}_diff_{file-path}.txt Three files are created for each file in the modified file list of each commit.

Fourth, read the API key from api_key.txt. We have three options. First, you can decide which of the three files (after, before, diff) created in the third step to query the gpt api. Second, you can enter which gpt api model to query in the gpt_api_model variable. Third, you can decide which prompt to use.

Command	python _4_gpt_responser.py
Ressult	Create commit-files/{index}_{first-option}_{file-path}_response.txt for each query file.

Finally, save it in Excel so that you can compare the contents of the query file with the response generated in the fifth and fourth steps. At this time, the column values in Excel are 'repository-name', 'commit hash', 'file path', '{first-option} file content', and 'gpt response'. In this case, first-option is the input value of the option selected among after, before, and diff in the fourth step.

Command	python _5_save2sheet.py
Ressult	commit-sheets/{repository-name}_{first-option}.xlsx

Through this process, you can obtain a closed project, its json format commit log, files at the time of each commit, gpt answers when querying those files, and an Excel file organizing them. If you want to proceed with these steps at once, run the following command.

Command	python _6_VIChecker.py