Replication and Reproduction of Watchman

ICSE 2020 Technical Track, ID#702 "Watchman: Monitoring Dependency Conflicts for Python Library Ecosystem"

Original authors:
Ying Wang ♥(Northeastern University, Email: wangying@swc.neu.edu.cn, Github ID: NeolithEra)
Ming Wen (Huazhong University of Science and Technology, Email: mwenaa@cse.ust.hk, Github ID: justinwm)
Yepang Liu (Southern University of Science and Technology, Email: liuyp1@sustech.edu.cn, Github ID: yepangliu)
Yibo Wang (Northeastern University, Email: wyb_neu@163.com) **Zhenming Li** (Northeastern University, Email: lzm_neu@163.com)
Chao Wang (Northeastern University, Email: wangc_neu@163.com) **Hai Yu** (Northeastern University, Email: yuhai@mail.neu.edu.cn)
Shing-Chi Cheung (The Hong Kong University of Science and Technology, Email: scc@cse.ust.hk, Github ID: sccust)
Chang Xu (Nanjing University, Email: changxu@nju.edu.cn) **Zhiliang Zhu** (Northeastern University, Email: ZHUZhiLiang_NEU@163.com)

The following person replicated this artifact:
Xiaoning Chang (University of the Chinese Academy of Sciences, Email: changxiaoning17@otcaix.iscas.ac.cn, Github ID: ChangXiaoning)

The following people reproduced this artifact:
Xue Yang (Neusoft Co. Ltd (SSE: 600718), Email: yangxue@neusoft.com) **Jie Liang** (Pinduoduo Co. Ltd (Nasdaq: PDD), Email: chiyuan@pinduoduo.com)

Replication and reproduction reports and materials are available at: http://www.watchman-pypi.com/reports

Abstract: The version of a library installed for a Python project can vary over time. For each required library, `pip` (i.e., the official client-side Python library installer) automatically installs the latest version of the library that satisfies the concerned constraint. Such automation comes with the risk of potential dependency conflict (DC) issues, which can cause build failures when the installed version of a library violates certain version constraints on the library. Diagnosing DC issues is a challenging task in the Python world, because the impact of any library updates on `PyPI` could be propagated transitively to a wide range of downstream projects. `Watchman` (**http://www.watchman-pypi.com/**) is developed to help Python developers deal with DC issues. It performs a holistic analysis of the entire `PyPI` ecosystem, continuously monitors the DC issues for millions of Python projects, and automatically submits issue reports and pull requests to open-source projects when it finds DC issues for the projects. We implemented `Watchman` as an online service for Python developers to check their projects to pinpoint DC issues. Our work can be replicated. We explain how to replicate it below.

(a) **Dataset of our empirical study in Section 3:** One can manually investigate the 235 collected DC issues in open-source Python projects on `GitHub` and check the correctness of our categorization of the issues.

(b) **Verifying the functional correctness of Watchman**: The replication includes building an FDG for a given Python project that simulates the process of installing dependencies and diagnosing DC issues for the project. To ease replication, we provide three example Python projects. One can verify the diagnosis information of three types of DC issues generated by `Watchman` for the three example projects. One can also use `Watchman` to analyze any Python projects released on `PyPI` to check their FDGs and corresponding diagnosis results. For illustration, we provide a video demo at https://youtu.be/3EOz9g3Bw70.

(c) **Verifying the experiment results reported in Section 5.1:** For replication, we provide scripts and raw data. The scripts help to automatically play back the evolution history of the collected 2,067 Python projects on `PyPI` (from 1 January, 2017 to 30 June, 2019) and detect DC issues during the above time period. The values of our proposed metrics **resolving ratio** and **lasting time** can be calculated by checking the outputted evolution information of each collected library.

(d) **Verifying the experiment results reported in Section 5.2:** For replication, one can check the following data: 1) the daily library update information on `PyPI` captured by `Watchman` during two time periods (from 1 July, 2019 to 10 August, 2019, and 1 December, 2019 to 31 December, 2019), and the corresponding downstream projects affected by the library updates identified by `Watchman`; and 2) diagnosis information and the statuses of the 279 real issues **(Watchman found and reported 162 more DC issues since our paper submission)** reported by `Watchman` to the open-source projects, during the two time periods. Then, the **confirmation rate** and **fixing rate** of these issues can be calculated easily.

Recently, we promoted our technique, `Watchman`, in the open source community via communicating with developers in the comments section of issue reports submitted by us. We also invited several experienced developers from two Chinese software companies, `Neusoft` Co. Ltd (SSE: 600718) and `Pinduoduo` Co. Ltd (Nasdaq: PDD), to apply `Watchman` to their company projects and asked for their feedback. We observed that 467 users from 15 different countries visited `Watchman` website from 9 December, 2019 and 17 January, 2020 (we tried our best to filter out the visits by robots/crawlers). And Watchman has successfully generated online diagnosis reports for 2,590 Python projects. In addition, the developers of the two software companies sent us reproduction reports via emails.

Our dataset of the 235 collected DC issues and scripts for replaying the evolution history of the libraries released on `PyPI` can help more researchers to understand/replicate/reproduce/improve the proposed diagnosis approach for DC issues in the Python ecosystem. The online service provided by us can help developers analyze their own projects to avoid potential DC issues.

♥ Corresponding author of this artifact