Poster Abstract: SmartAppZoo: a Repository of SmartThings Apps for IoT Benchmarking

Zhaohui Wang
EECS/I2S, The University of Kansas
Lawrence, KS, United States
zhwang@ku.edu

Bo Luo
EECS/I2S, The University of Kansas
Lawrence, KS, United States
bluo@ku.edu

Fengjun Li
EECS/I2S, The University of Kansas
Lawrence, KS, United States
fli@ku.edu

ABSTRACT

A well-organized SmartApps dataset provides a valuable resource for researchers to evaluate their work on smart home automation systems. The IoTBench dataset created by Celik et al. [1] is a significant contribution to the IoT research community [1]. However, due to the fast growth of SmartApps and the retirement of some old apps, the IoTBench dataset becomes outdated. The research community is in need of a new large-scale and carefully cleaned benchmarking dataset. In this poster, we present a new repository, namely, SmartAppZoo, which contains 3,526 SmartApps collected from GitHub repositories, including 184 SmartThings official apps, 468 third-party apps from IoTBench, and 2,874 new third-party apps. SmartAppZoo is a manually-verified, comprehensive, clean, and diverse IoT benchmarking dataset. SmartAppZoo is available at: https://github.com/SmartAppZoo/SmartAppZoo.

CCS CONCEPTS

• **Software and its engineering → Software libraries and repositories**.

KEYWORDS

Dataset, IoT, SmartThings, SmartApps

ACM Reference Format:
Zhaohui Wang, Bo Luo, and Fengjun Li. 2023. Poster Abstract: SmartAppZoo: a Repository of SmartThings Apps for IoT Benchmarking. In *International Conference on Internet-of-Things Design and Implementation (IoTDI '23), May 09–12, 2023, San Antonio, TX, USA.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3576842.3589162

1 INTRODUCTION

Smart IoT devices and applications nowadays play a very important role in our daily lives. The SmartThings ecosystem has become the most widely adopted IoT platform worldwide, with 220 million registered users in 2022. SmartApps are used to manage SmartThings devices such as thermostats, locks, and switches. Research efforts have been devoted to various aspects of SmartApps, e.g., intelligent controls, device collaboration, and security and privacy. The IoTBench dataset [1] was shared with the research community in 2018. However, the 5-year-old dataset appears to be in need of an update, as: (1) the dataset scale does not reflect the significant growth of SmartApps and becomes insufficient for large-scale benchmarking needs; (2) the dataset is outdated that some apps are removed or replaced by new apps, while some new apps with new features or new vulnerabilities are not included; and (3) it contains duplicate apps and apps with syntax errors that they do not compile.

To address these issues, we devote efforts to collecting an up-to-date and clean repository of open-source SmartApps, namely the SmartAppZoo and share it with the research community. SmartAppZoo aims to achieve two objectives, *scalability* and *usability*. We expect to collect all open-source SmartApps that are available and provide a clean dataset without duplicates or meaningless apps that do not compile or do not provide any functionality. To the best of our knowledge, we have collected the largest number of SmartApps from real-world applications. In this poster, we explain the process to collect and clean the dataset and present the basic statistics of the dataset. We share the dataset with the research community at: https://github.com/SmartAppZoo/SmartAppZoo.

[1]https://github.com/IoTBench/IoTBench-test-suite

IoTDI '23, May 09–12, 2023, San Antonio, TX, USA
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0037-8/23/05.
https://doi.org/10.1145/3576842.3589162

2 DATA COLLECTION AND CLEANING

The data collection process consists of three phases: crawling, SmartThings app identification, and app cleaning.

App Crawling. SmartApp developers may share their apps on the SmartThings community website [2] or post open-source SmartApps in GitHub repositories. We first retrieved the 184 apps that are officially posted in SmartThings' public GitHub repository and labeled them as the *official apps* in SmartAppZoo. This set of apps is different from the official apps in the IoTBench dataset as five official apps were removed in 2020. Next, we collected all the SmartApps from GitHub to create a *candidate* third-party SmartApps dataset. To collect all available third-party apps, we defined a list of 15 SmartApp-related keywords such as "SmartThing", "SmartApp", and "IoT" and used the REST API of GitHub [3] to search for repositories with these keywords. We excluded all forked repositories to avoid duplications. The REST API returns only up to 1,000 results for each search. Therefore, we used non-overlapping time windows to construct the queries (e.g., from 01/01/2022 to 02/01/2022) and reduced the window size until the search returned less than 1,000 results. Eventually, the crawler covered all the repositories in the date range between January 1, 2013 and December 31, 2022. In this process, we avoided the hand-crafted malicious SmartApps from IoTBench or other third-party GitHub repositories used for adversarial SmartApp research. Finally, we extracted all the Groovy files from the repositories in this candidate SmartApps dataset. We appended the repository name and GitHub username in the form of "user_name@repo_name" to the end of the filename of all the crawled Groovy files so that we can resolve any potential conflict in file names and easily trace the origin of all the crawled apps.

App Identification. Not all Groovy files in the candidate app dataset are valid SmartApps. In particular, both SmartApps and device type handlers have "preferences" blocks, while device type handlers also have "metadata" blocks, which do not exist in SmartApps. We use regular expression "`preferences(\(.*\))?\s*{`" and "`metadata(\(.*\))?\s*{`" to check if a Groovy file contains a "metadata" or "preferences" block. If it contains neither, it is not a SmartApp or a device type handler. If it has only "metadata", it is likely a device type handler; otherwise, it is a SmartApp.

App Cleaning. We observed a large number of duplicate apps in the candidate app set. Moreover, a significant number of similar apps were found. For instance, some apps have identical code (called "code-identical apps") but have different comments or definition blocks, e.g. different authors. Finally, some apps have highly similar codes to fulfill highly similar functions (called "near-duplicate apps"). Removing them avoids unnecessary overhead when using SmartAppZoo for testing while not affecting its utility. The app cleaning pipeline involves five steps to eliminate identical, code-identical, and near-duplicate apps as well as non-functional and invalid apps. Here, we introduce the details of the process.

(1) Clean identical apps. First, we removed all the apps that are exact duplicates, i.e., Groovy files that produce identical hashes. In practice, many apps were found in multiple repositories. In this case, we kept only one app in SmartAppZoo and gave priorities to official SmartThings apps. Approximately 75% of the collected apps were removed in this step.

(2) Clean non-functional apps. We scanned the codebase of each app to remove the non-functional ones that do not have any user-defined code/functions. For instance, apps that only contain headers or system-defined methods such as "installed" and "updated".

(3) Clean invalid apps. We compiled all the remaining apps and excluded the ones that failed to compile. A small number of apps, for example, with syntax errors in their codebase were removed.

(4) Clean code-identical apps. To identify apps with identical function codebase, we extracted the code, excluded the definition blocks, and removed all comments. If multiple apps had identical cleaned code, we kept only one of them in SmartAppZoo.

(5) Clean near-duplicate apps. We calculated a fuzzy hash of the cleaned Groovy code using TLSH [5], computed the pair-wise distances, and adopted an agglomerative clustering algorithm to group the apps into clusters. The pair-wise distance of apps in the same cluster is less than a preset threshold (30 in our approach) [4]. We used a combination of regular expressions and parenthesis matching to extract the descriptions, inputs, subscriptions, and actions from each app. When two apps in the same cluster have *different* inputs, subscriptions, or actions, they were both preserved. We removed all other near-duplicate apps.

3 STATISTICS

We collected 51,336 Groovy files from 7,841 GitHub repositories, which included 22,421 SmartApps, 27,273 device type handlers, and 1,642 Groovy files that were neither SmartApps nor device type handlers. In particular, 405 Groovy files in SmartThings' official repository and 13 IoTBench third-party apps were device-type handlers. To ensure the accuracy of our categorization, we randomly selected Groovy files from each category for manual verification.

Table 1: Statistics of SmartApps datasets.

Datasets	Official	IoTBench	SmartAppZoo
number of apps	184	652	3526
number of descriptions	170	444	2566
number of capabilities	38	54	118

We used the app cleaning pipeline described in Section 2 to remove non-functional and invalid SmartApps and eliminate identical and near-duplicate apps. We removed 17,309 duplicate apps, 277 non-functional and invalid apps, 669 code-identical apps, and 640 near-duplicate apps. Finally, SmartAppZoo contains 3,526 SmartApps, which are all valid, functional, and unique. Table 1 lists the total number of apps, descriptions, and capabilities used in the official SmartThings dataset, the IoTBench dataset, and the SmartAppZoo dataset. As shown in the table, SmartAppZoo contains significantly more SmartApps than the other two datasets.

4 CONVERSION AND LICENSE

The SmartThings platform has retired Groovy and introduced a new API using Node.js SDK. The transition is ongoing. With searches of SmartThings Community and Github, we found that the official repositories are almost the only sources for Node.js SmartApps. We believe that the SmartThings community still has a long way to go in migrating all the apps from Groovy to Node.js, which means SmartAppZoo will remain useful.

The main logic of Groovy and Node.js SmartApps is very similar, making it relatively easy to convert Groovy apps into Node.js apps. We have developed a conversion tool that accurately translates the definition blocks, preference blocks, subscription functions, schedule functions, and device commands from Groovy to Node.js. However, translating other user-defined functions requires additional work. In SmartAppZoo, we include the converter software and 50 sample Node.js SmartApps generated by the converter.

According to Github's terms of service, we can fork the repositories and create our own copies of SmartApps. Hence, we forked the SmartApps and acknowledged the original developers.

5 CONCLUSION

In this paper, we present a dataset of deduplicated and cleaned real-world SmartApps to support the benchmarking of Smart home automation systems. Our dataset captures a rich set of SmartApps, including 184 SmartThings official apps, 468 IoTBench third-party apps, and 2,874 GitHub third-party apps. With further updates and expansion, the dataset will continue to be a valuable resource for testing and evaluating research projects related to SmartApps.

REFERENCES

[1] Z Berkay Celik, Leonardo Babun, Amit Kumar Sikder, Hidayet Aksu, Gang Tan, Patrick McDaniel, and A Selcuk Uluagac. 2018. Sensitive information tracking in commodity IoT. In *USENIX Security Symposium*.

[2] SmartThings Community. 2023. *SmartApps & Automations*. Retrieved Feb 2, 2023 from https://community.smartthings.com/c/smartapps/6

[3] Github. 2023. *GitHub Docs*. Retrieved Feb 2, 2023 from https://docs.github.com/en/rest/search?apiVersion=2022-11-28#about-the-search-api

[4] Amanda Lee and Travis Atkison. 2017. A comparison of fuzzy hashes: evaluation, guidelines, and future suggestions. In *Proceedings of the SouthEast Conference*.

[5] Jonathan Oliver, Chun Cheng, and Yanggui Chen. 2013. TLSH–a locality sensitive hash. In *Cybercrime and Trustworthy Computing Workshop*.