Advanced Mechanism for Logging Method Invocations in Android Applications

1 Introduction

We use a tool named Droidmon to log method invocations in Android applications [1]. Droidmon accepts as input a list of monitored methods. Droidmon logs the methods whenever they are invoked and prints many information about them (e.g., their arguments, the time when they are invoked). Droidmon typically produces a relatively huge amount of logs. When reading the logs afterwards, we have to filter them because we require only a portion of the logged information. It would be better if we can have a more concise log. In this work, we propose an approach to generate such log.

2 Proposed Approach

To minimize the log size, we propose an advanced mechanism that can generate a rich and fine-grained log succinctly while also reducing the overall log generation time. To achieve this goal, we use a combination of two techniques: *Targeted Logging* and *Method Name Encoding*. The first enforces the log to record only the required information. The second encodes method names in the log to numbers, which typically have a much shorter length than method names.

2.1 Targeted Logging

As mentioned before, Droidmon logs many information about a monitored method whenever the method is invoked. There are two issues that we encountered in this logging process. First, we do not need to log the method when it is invoked by an application that we are not monitoring. Second, we only require a portion of the logged information. To fix these issues, we make the logging process targets only relevant contexts and information. Specifically, we modify Droidmon as follows:

1. We add an input to Droidmon that accepts a list of monitored applications. We enable logging only for these input applications.

2. We inject a method filter to Droidmon to make it print only the required information for each monitored method.

The above modifications shorten the generated log and allow us to simplify our log reading process.

2.2 Method Name Encoding

In a log generated by Droidmon, method names are usually written frequently. To reduce log size, we encode method names to numbers using a dictionary which is defined from the input list of monitored methods. Each method is mapped to its order in the list. For example, the first method in the list is encoded to 1, the second method is encoded to 2, and so on. This makes the dictionary implicit, thereby saving some spaces as we have no need to print it along with the log.

3 Experiments

We experiment on five benign applications taken from Le et al. work [3]. These applications are listed in Table 1. We run each application in a Google Nexus 5X 4.4.4 emulator from Genymotion [2] that is installed in a MacBook Pro (13-inch, 2017) with 2.3 GHz Intel Core i5 processor and 16GB RAM. We use DroidBot [4] with default settings to generate 100 input events and interact with the applications. For each application, we select the monitored methods following Le et al. work. We record the time required for executing the generated events and measure the size of the generated log when using Droidmon with and without our logging mechanism.

Table 1: Android Applications Dataset

Application Package	**Functionality**
appinventor.ai_rintoadi.RadioICBB	Radio Islamic Centre Bin Baz
com.nextminute.app	Job management
cz.kinst.jakub.clockq	A simple digital clock widget
com.pommedeterresautee.angrybirdsseasonunlock	AngryBirds Seasons unlock
lysesoft.andftp	An FTP client for Android

Table 2: Execution Time when using Droidmon with and without Our Logging Mechanism (in seconds)

Application Package	**Without**	**With**	**Improv.**
appinventor.ai rintoadi.RadioICBB	271.62	172.812	36.36%
com.nextminute.app	332.363	320.104	3.69%
cz.kinst.jakub.clockq	581.241	506.826	12.80%
com.pommedeterresautee.angrybirdsseasonunlock	994.613	885.002	11.02%
lysesoft.andftp	906.981	634.876	30.00%
Total	3086.818	2519.62	18.37%

Table 2 and Table 3 show the execution time and log size for the five experimented applications when using Droidmon with and without our logging mechanism. They show that our logging mechanism consistently improves both the execution time and the generated log size. In total, our logging mechanism reduce the execution time by 18.37% and the log size by 99.47%. Notice that the improvement for log size is high. This is due to the fact that the logs generated when running Droidmon without our logging mechanism mostly contain method invocations from non-monitored applications. The non-monitored applications include DroidBot, which is used to interact with the monitored applications.

Table 3: Log Size when using Droidmon with and without Our Logging Mechanism (in kibibytes)

Application Package	**Without**	**With**	**Improv.**
appinventor.ai rintoadi.RadioICBB	5,184	256	95.06%
com.nextminute.app	1,024	20	98.05%
cz.kinst.jakub.clockq	15,424	12	99.92%
com.pommedeterresautee.angrybirdsseasonunlock	16,448	12	99.93%
lysesoft.andftp	19,520	8	99.96%
Total	57,600	308	99.47%

References

[1] Droidmon - Dalvik Monitoring Framework for CuckooDroid. `https://github.com/idanr1986/droidmon`. Last Accessed: 8 Nov. 2018.

[2] Genymotion Android Emulator. `https://www.genymotion.com/`. Last Accessed: 8 Nov. 2018.

[3] Tien-Duy B. Le, Lingfeng Bao, David Lo, Debin Gao, and Li Li. Towards Mining Comprehensive Android Sandboxes. In *The 23rd International Conference on Engineering of Complex Computer Systems (ICECCS)*, 2018.

[4] Yuanchun Li, Ziyue Yang, Yao Guo, and Xiangqun Chen. Droid-Bot: a lightweight UI-guided test input generator for Android. In *EEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C)*, pages 23–26. IEEE, 2017.