DroidCat Artifacts Evaluation Instructions

❖ Contents

Overview 1

Prerequisites 1

Source code and executable 2

Project website and documentation 2

Evaluation dataset 2

BENCHMARKS 2

TRAINING/TESTING SAMPLES 3

DYNAMIC CHARACTERIZATION RESULTS (MALWARE-BENIGN DIFFERENCES) 3

MALWARE DETECTION/CATEGORIZATION RESULTS 3

CASE STUDY RESULTS 4

Using DroidCat as a tool 4

Overview

This document gives the instruction for *accessing and reproducing* artifacts related to our research-paper submission on DroidCat, a unified dynamic approach to Android malware detection. The artifacts include source code, executable, project website and documentation for DroidCat, and evaluation dataset.

Prerequisites

To reproduce our artifacts, the following environments and tools are needed to be installed first:

- Python (used version: 2.7)
- Scikit-learn (used version: 0.18.1)
- JRE (used version: 1.8) and R for linux (used version: 3.3.2)
    - only needed for reproducing the dynamic characterization study related artifacts, not required for reproducing DoridCat classification results

We verified the artifact reproduction instructions under Ubuntu 15.

Source code and executable

The source code is included in the *droidcat*, directory, where the *src* subdirectory contains the Java code used for profiling and feature computation while the *ML* subdirectory contains the Python code for the multi-class classification and associated evaluation experiments.

The major executable is *droidcat/droidcat.jar*. The subdirectory *droidcat/libs* includes all libraries that are needed for compiling the source code and executing the DroidCat executable in order to reproduce our experimental dataset and final results as reported in the paper.

All these artifacts can be freely browsed and downloaded from our repository on bitbucket (*the link is withheld for now to comply with the double-blind review policy*).

Project website and documentation

The website content is placed in the *projectweb* directory. You may open *index.html* in there to start browsing the project website, where the documentation about this project can be found, including how to use DroidCat (*projectweb/page_usage.html*).

The detailed definition of each of the 122 metrics (features) can be found in the page projectweb/*metrics.htm*.

Evaluation dataset

BENCHMARKS

The dataset used in the paper includes the 271 Android benchmark apps (136 benign apps and 135 malware as identified as benign/malware by VirusTotal). Due to the large total volume of these apps, we did not upload the APKs themselves here. The benchmark names are provided in two list files: *data/benchmarks/used-benign-apps-droidcat.txt* and *data/benchmarks/used-malware-apps-droidcat.txt*, for benign and malware samples used in our study, respectively.

To download the benign-app apks from Google Play, a script has been prepared, to be used as:

droidcat/scripts/downloadapklist.sh the-apk-list-file (e.g., *data/used-benign-apps-droidcat.txt).*

We used the *Google Play API for Python* (contained in *droidcat/scripts/googleplay-api-master*) for downloading apps from Google Play. It needs google-protobuf for python be installed in order to work: you may install google-protobuf in Ubuntu using "*sudo apt-get install python-protobuf".*

The malware apps were downloaded from the Malware Genome Project.

TRAINING/TESTING SAMPLES

The per-app feature vectors each of 122 metric/feature values computed for the benchmarks are the actual data used for training and testing by DroidCat. These values are computed from the method-call and ICC traces as a result of our behavioral profiling of Android apps with random inputs from the Monkey tool.

The feature vectors used in the paper can be found in *droidcat/ML/features*.

For reproducing these feature values or generate them for a different set of apps, the usage page *(*projectweb/page_usage.html*)* of the project website provides the main instructions. The complete reproduction (starting from instrumenting the apps, and runtime tracing, followed by trace analysis) would need at least 46 hours (for tracing the 271 benchmarks each running 10min to gather the traces and for feature computation), as well as the setup of Android SDK and emulator. All the scripts mentioned on that usage page *(*projectweb/page_usage.html*)* are available in *droidcat/scripts.*

DYNAMIC CHARACTERIZATION RESULTS (MALWARE-BENIGN DIFFERENCES)

The original results of the dynamic characterization study as presented in the paper can be found in *data/characterization (*subdirectory *benign* and *malware* contains such raw results for benign and malware group, respectively). It is from these raw results where we summarized the statistics about the behavioral differences between malware and benign as listed in *data/characterization/behavioralDifferences.xlsx*, of which the highlights were plotted in Figure 2 of the paper. To reproduce the summary statistics shown in the Excel sheets, please follow the instructions in *data/characterization/ trace_statistics.html* (mainly using R thus needing R to be installed beforehand).

MALWARE DETECTION/CATEGORIZATION RESULTS

The original results for the unified detection (via the confusion matrix) shown in Table 3 of the paper are listed in *data/unifieddetection/confusion-matrix.xlsx*. To reproduce it, make sure *aapt has* been installed (if not, "*sudo apt-get install aapt*") and then run:

```
cd droidcat/ML && python confusionMatrix_multipleModels_loo_forMajorFamilyOnly.py
```

The original results of the conventional detection (binary classification) results reported in 5.3 of the paper can be found in *data/unifieddetection/classificationResults-binary.xlsx*. To reproduce it, run:

```
cd droidcat/ML && python multipleModels_tab.py true true|false

The last argument switches output between accuracy (true) and precision/recall/F1 (false).
```

- The instructions for reproducing the above results on unified malware detection can also be found in the usage page of the project website (projectweb/page_usage.html).

The original efficiency results can be found in *data/unifieddetection/efficiency.xlsx.*

Finally, the original results on the effects of feature set and learning algorithm choices as plotted in Figures 3 and 4 of the paper can be found in *data/unifieddetection/Effects-of-algs-and-feature-sets.xlsx.* To reproduce it, run:

```
cd droidcat/ML && python confusionMatrix_multipleModels_loo_forMajorFamilyOnly_all.py
```

Case Study Results

The original results of case studies can be found in *data/unifieddetection/case-studies.xlsx*.

Specifically, the results corresponding to our findings in Section 5.3 of our paper are in the sheet '*malwareAsBenign'*, which can be reproduced by running

```
cd droidcat/ML && python case_study_malwareasbenign.py
```

The results corresponding to our findings in the "*A Case Study of DroidCat's Effectiveness*" in Section 5.2 of the paper are in the sheet '*topfeatureValuesByGoodBadGroup',* which can be reproduced by running

```
cd droidcat/ML && python case_study_family_topmetrics.py
```

Using DroidCat as a tool

The detailed instructions for using our tool on an unknown app for unified malware detection are found in the usage page of the project website (projectweb/page_usage.html).

```
cd droidcat/ML && python DroidCat_detect.py true|false

The last argument is true for binary detection and false (default) for the unified detection.
```