USENIX'23 Artifact Appendix: Credit Karma: Understanding Security Implications of Exposed Cloud Services through Automated Capability Inference

Xueqiang Wang
University of Central Florida

Yuqiong Sun
Meta

Susanta Nanda
ServiceNow

XiaoFeng Wang
Indiana University Bloomington

A Artifact Appendix

A.1 Abstract

The submission pertains to PrivRuler, a tool utilized in the research paper titled "Credit Karma: Understanding Security Implications of Exposed Cloud Services through Automated Capability Inference." PrivRuler comprises two key components. The first one, AppAnalysis, is a static app analysis component that extracts cloud service credentials and usages from mobile applications. The second component, CloudProbe, takes the output of AppAnalysis as input and probes the associated cloud services to infer the additional capabilities granted to mobile applications.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

The AppAnalysis component runs as a Java program on the local machine, and it does not alter any system settings that might impact security, nor does it transmit any data to backend servers.

The CloudProbe component operates within an Android emulator or a test Android device, allowing it to probe cloud services and infer the capabilities associated with a specific cloud credential. In this submission, we established a test credential on our AWS accounts, ensuring that running the CloudProbe component for functional testing would not compromise the privacy of any third parties. Additionally, we minimized the risks associated with cloud service probing using multiple strategies, as detailed in Section 3.6 of the paper.

A.2.2 How to access

URL: https://github.com/privruler/PrivRuler-Public
Commit: 8ff0ae9c8d2611072fde0b112e71b8f662fb2507

A.2.3 Hardware dependencies

There is no hardware dependencies to run PrivRuler.

A.2.4 Software dependencies

- PrivRuler runs on basically all operating systems, including Windows, MacOS, and Linux. We recommend MacOS or Linux as these are the operating systems we test more often.
- Recommend JDK Version 8
- Android Studio
- Android platform tools (e.g., adb)
- Android emulator or device of API Level < 30 (Recommended 28).

A.2.5 Benchmarks

None.

A.3 Set-up

A.3.1 Installation

Install AppAnalysis. AppAnalysis is written in Java, and we have created a script to automate the compiling and dependency management process. Users can follow the below steps to compile it.

- `cd $dir/PrivRuler-Public/AppAnalysis`
- `./compile.sh`

Install CloudProbe. CloudProbe is delivered as an Android app. Therefore, we need an Android emulator and Android Studio to install it.

- Create an Android Emulator with API Level 28.
- Import CloudProbe into Android Emulator, and hit "`Run app`" button to install CloudProbe on the emulator.

A.3.2 Basic Test

We created a test app folder at "$dir/PrivRuler-Public/AppAnalysis/apks/" for basic testing.

Run AppAnalysis. Use the below commands to analyze test apps using AppAnalysis:

- `cd $dir/PrivRuler-Public/AppAnalysis`
- `./analyze apks apks`

The output will be stored in `output/app-debug.output` file. A successful run of AppAnalysis will generate a line that contains "`cloudAPIs`" keyword in the output file.

Transfer AppAnalysis result to Android emulator. Launch the Android emulator, and run the below command to transfer the output of AppAnalysis to the emulator.

- `grep -Rh 'appName.*cloudAPIs' $dir/PrivRuler-Public/AppAnalysis/output » summary`
- `adb shell mkdir /sdcard/cloudassets`
- `adb push summary /sdcard/cloudassets`

Run CloudProbe. In Android Studio, click the "`Run app`" button to run CloudProbe. Once the app is launched in the emulator, click the three buttons on the app UI. A successful run will generate analysis results under emulator folder `/sdcard/AWSSummaries/`, with each app has a file named `summary_⟨packagename⟩.json`. Users may check presence of this file by running: `adb shell ls -alh $file_path`.

A.4 Evaluation workflow

We do not request for a complete evaluation since it will require analyzing over 1.3M apps (over 30TB) in storage, and re-probe the cloud backends of 12K apps.

A.5 Notes on Reusability

In addition to inferring over-privileges in cloud services, the artifact has the potential to be used in several other ways:

- **Analyzing mobile apps for sensitive information beyond cloud service credentials.** While the artifact's primary focus is detecting cloud service credentials within mobile apps, it can be customized to scan and identify other sensitive information present within mobile apps with ease.
- **Identifying obfuscated APIs.** As a part of the AppAnalysis component, the code extracts fingerprints for obfuscated APIs by examining invariant information like the number of arguments. The obfuscated APIs fingerprinting module can be employed to analyze other obfuscated APIs aside from cloud APIs.
- **Enhancing mobile app security.** By employing the PrivRuler tool, app developers can evaluate the security of their mobile apps and identify any potential vulnerabilities in regard to cloud services, thereby enhancing their app's security posture and safeguarding against cyber attacks such as data leaks.

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20220926. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at `https://secartifacts.github.io/usenixsec2023/`.