Playing Without Paying: Detecting Vulnerable Payment Verification in Native Binaries of Unity Mobile Games

Chaoshun Zuo and Zhiqiang Lin, *The Ohio State University*

https://www.usenix.org/conference/usenixsecurity22/presentation/zuo

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 31st USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 31st USENIX Security Symposium.

August 10–12, 2022 • Boston, MA, USA

978-1-939133-31-1

A Artifact Appendix

A.1 Abstract

Obligatory. Briefly describe your artifact including minimal hardware and software requirements, how it supports your paper, how it can be validated, and what is the expected result. At submission time, it will also be used to select appropriate reviewers. It will also help readers understand what was evaluated and how.

The artifact is the source code of the tool (i.e., PaymentScope) we proposed in the paper. It can detect payment bypass vulnerability in Unity mobile games. It is implemented atop **Ghidra**. To evaluate PaymentScope, we have attached 15 games in which 10 of them are vulnerable. PaymentScope can detect that 10 of them are vulnerable and it can tell the vulnerability type (i.e., local-verification or no-verification). To run PaymentScope, we have prepared a VirtualBox VM in which all the requirements have been setup. The VM needs 2 cores CPU and 8GB memory (mostly required by Ghidra)

A.2 Artifact check-list (meta-information)

Obligatory. Fill in whatever is applicable with some keywords and remove unrelated items.

- **Program:** the source code of PaymentScope
- **Run-time environment:** VirtualBox VM with 2 cores CPU and 8GB memory
- **Security, privacy, and ethical concerns:** please don't use the tool to attack any real games.
- **Output:** identify 5 local-verification and 5 no-verification games
- **Experiments:** run `/home/paymentscope/Desktop/runPaymentScopeOnTestData.py` in the VM
- **How much disk space required (approximately)?:** 30GB
- **How much time is needed to prepare workflow (approximately)?:** 1 hour depending on the Internet bandwidth
- **How much time is needed to complete experiments (approximately)?:** less than 1 hour

A.3 Description

Obligatory. For inapplicable subsections (e.g., the "How to access" subsection when not applying for the "Artifacts Available" badge), please specify 'N/A'.

A.3.1 How to access

N/A

A.3.2 Hardware dependencies

2 cores CPU and 8GB memory

A.3.3 Software dependencies

VirtualBox

A.3.4 Data sets

15 games. Among them, 10 are vulnerable. In particular, 5 are local-verification and 5 are no-verification games.

A.3.5 Security, privacy, and ethical concerns

Please don't use the tool to attack any real games.

A.4 Installation

Obligatory. Describe the setup procedures for your artifact targeting novice users (even if you use a VM image or access to a remote machine).

- Read the `README.md` file for the source code in `Source Code` folder
- Read the `README.md` file for the Virtual Machine in `Virtual Machine` folder
- Install the VirtualBox

A.5 Experiment workflow

Describe the high-level view of your experimental workflow and how it is implemented, invoked and customized (if needed), i.e. some OS scripts, IPython/Jupyter notebook, portable CK workflow, etc. This subsection is optional as long as the experiment workflow can be easily embedded in the next subsection.

- Login to VM
- Run `/home/paymentscope/Desktop/runPaymentScopeOnTestData.py` to conduct the experiments
- For each output folder, find the `isVulnerable` field in `analysisRes.json` file. The field indicates whether the game is vulnerable and the vulnerability type.

A.6 Evaluation and expected results

Obligatory. Start by listing the main claims in your paper. Next, list your key results and detail how they each support the main claims. Finally, detail all the steps to reproduce each of the key results in your paper by running the artifacts. Describe the expected results and the maximum variation of empirical results (particularly important for performance numbers).

Main claim: PaymentScope can detect payment bypass vulnerability in Unity mobile games. It is implemented by the guidance of the Algorithm 1 in the paper.

To support the claim, we have attached a PDF in the source code to map our implementation to the Algorithm 1. In addition, we have attached 15 games for testing, in which 5 are local-verification and 5 are no-verification games. We have manually verified the 10 games and they are indeed vulnerable. The games and the vulnerability types is explained in file `PaymenScope/Virtual Machine/README.md`.

After run `runPaymentScopeOnTestData.py`, the vulnerability type can be found in `isVulnerable` field in `analysisRes.json` file which is located in the output folder. 5 of them should be no-verification, 5 of them should be local-verification and the rest should be 'secure'.

A.7 Experiment customization

A.8 Notes

A.9 Version

Based on the LaTeX template for Artifact Evaluation V20220119.