GlobalConfusion: TrustZone Trusted Application 0-Days by Design

Marcel Busch, Philipp Mao, and Mathias Payer, *EPFL*

https://www.usenix.org/conference/usenixsecurity24/presentation/busch-globalconfusion

This artifact appendix is included in the Artifact Appendices to the Proceedings of the 33rd USENIX Security Symposium and appends to the paper of the same name that appears in the Proceedings of the 33rd USENIX Security Symposium.

August 14–16, 2024 • Philadelphia, PA, USA

978-1-939133-44-1

USENIX Security ’24 Artifact Appendix: GlobalConfusion: TrustZone Trusted Application 0-Days by Design

Marcel Busch Philipp Mao Mathias Payer
EPFL, Lausanne, Switzerland

A Artifact Appendix

A.1 Abstract

Trusted Execution Environments form the backbone of mobile device security architectures. The GlobalPlatform Internal Core API is the de-facto standard that unites the fragmented landscape of real-world implementations, providing compati- bility between different TEEs. Unfortunately, our research reveals that this API standard is prone to a design weakness. Manifestations of this weakness result in critical type-confusion bugs in real-world user-space applications of the TEE, called Trusted Applications (TAs). At its core, the design weakness consists of a fail-open design leaving an optional type check for untrusted data to TA devel- opers. The API does not mandate this easily forgettable check that in most cases results in arbitrary read-and-write exploitation primitives. To detect instances of these type-confusion bugs, we design and implement GPCheck, a static binary analysis system capable of vetting real-world TAs. We employ GPCheck to analyze 14,777 TAs deployed on widely used TEEs to investigate the prevalence of the issue. We reconfirm known bugs that fit this pattern and discover unknown instances of the issue in the wild. In total, we confirmed 9 known bugs, found 10 instances of silently-fixed bugs, and discovered a surprising amount of 14 critical 0-day vulnerabilities using our GPCheck prototype. Our findings affect mobile devices currently in use by billions of users.

This artifact aims to demonstrate GPCheck’s capabilities to detect the above mentioned type-confusion bugs.

A.2 Description & Requirements

A.2.1 Security, privacy, and ethical concerns

GPCheck is fully dockerized and should not harm the computer of the artifact evaluator.

A.2.2 How to access

The code and documentation are available on GitHub: `https://github.com/HexHive/GlobalConfusion/tree/sec-ae`.

A.2.3 Hardware dependencies

We do not have special hardware requirements. Our analyzer can be configured with a longer timeout to compensate for slow hardware.

A.2.4 Software dependencies

We tested our setup on Ubuntu 22.04.2 using Docker version 23.0.1 and the docker-compose plugin version 1.29.2.

A.2.5 Benchmarks

Our dataset consists of proprietary TA blobs. Due to intellectual property concerns, we cannot publicly share these blobs. For research purposes, we will share our dataset on demand.

A.3 Set-up

A.3.1 Installation

Follow the SETUP steps in our `README.md`.

A.3.2 Basic Test

Analyze the two pre-compiled examples mentioned in HOWTO in our `README.md`.

A.4 Evaluation workflow

A.4.1 Major Claims

(C1): GPCheck can detect type-confusion bugs resulting from the design weakness of the GP TEE Core Internal API in closed-source TAs (Table 3 and Table 4).

(C2): The type-confusion vulnerability resulting from the design weakness of the GP TEE Core Internal API is widely prevalent in the Android mobile device ecosystem.

A.4.2 Experiments

(E1): *[GP Function and Type-Confusion Bug Detection] [30 human-minutes + 2 compute-hours]*:
In this experiment, we provide the datasest from Table 3 and Table 4 to demonstrate GPCheck’s capability to

detect GP function in stripped binaries, and its type-confusion detection on a ground-truth dataset.

Preparation: Download the GP function detection ground-truth dataset and the GP type-confusion ground-truth dataset.

Execution: Run GPCheck as described on the `README.md` on these datasets. Make sure to select the correct `TEE` to trigger the correct function detection logic. For instance `oppo_kinibi` for TAs on Oppo Kinibi, `mitee` for MiTEE TAs, and `qualcomm` for QSEE TAs.

Results: The function detection for the TAs in the GP function detection ground-truth dataset should resemble the results in Table 3. When a TA is not GP-compliant, the report should say "*not GP compliant*".

The GP type-confusion detection for the GP type-confusion ground-truth dataset should resemble the results shown in Table 4. The results provide evidence for **C1**.

(E2): *[TA Type-Confusion Vulnerabilities] [30 human-minutes + 2 compute-hours]*:

In total, we found 33 unique vulnerable TAs. This experiment serves to confirm these vulnerable TAs.

Preparation: Download the vulnerable TAs dataset.

Execution: Run GPCheck as described on the `README.md` on this dataset.

Results: GPCheck should report all of these TAs to be vulnerable. These TAs are distributed across 5 reputable OEMs and affect 54 recent mobile devices employing the 5 dominant TEE implementation on the market. Given the wide spread of the type-confusion bug, we conclude that it is widely prevalent in the Android mobile ecosystem (**C2**).

A.5 Notes on Reusability

Our analyzer can be used to detect GP type-confusion bugs in future proprietary GP-compliant TAs.

A.6 Version

Based on the LaTeX template for Artifact Evaluation V20231005. Submission, reviewing and badging methodology followed for the evaluation of this artifact can be found at `https://secartifacts.github.io/usenixsec2024/`.