ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

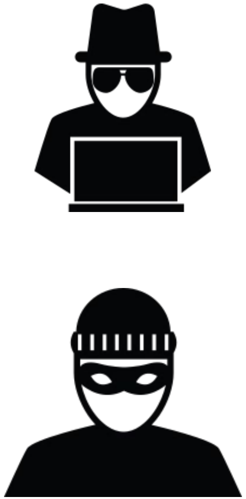
🌍😵‍💫GlobalConfusion: TrustZone Trusted Application 0-Days by Design

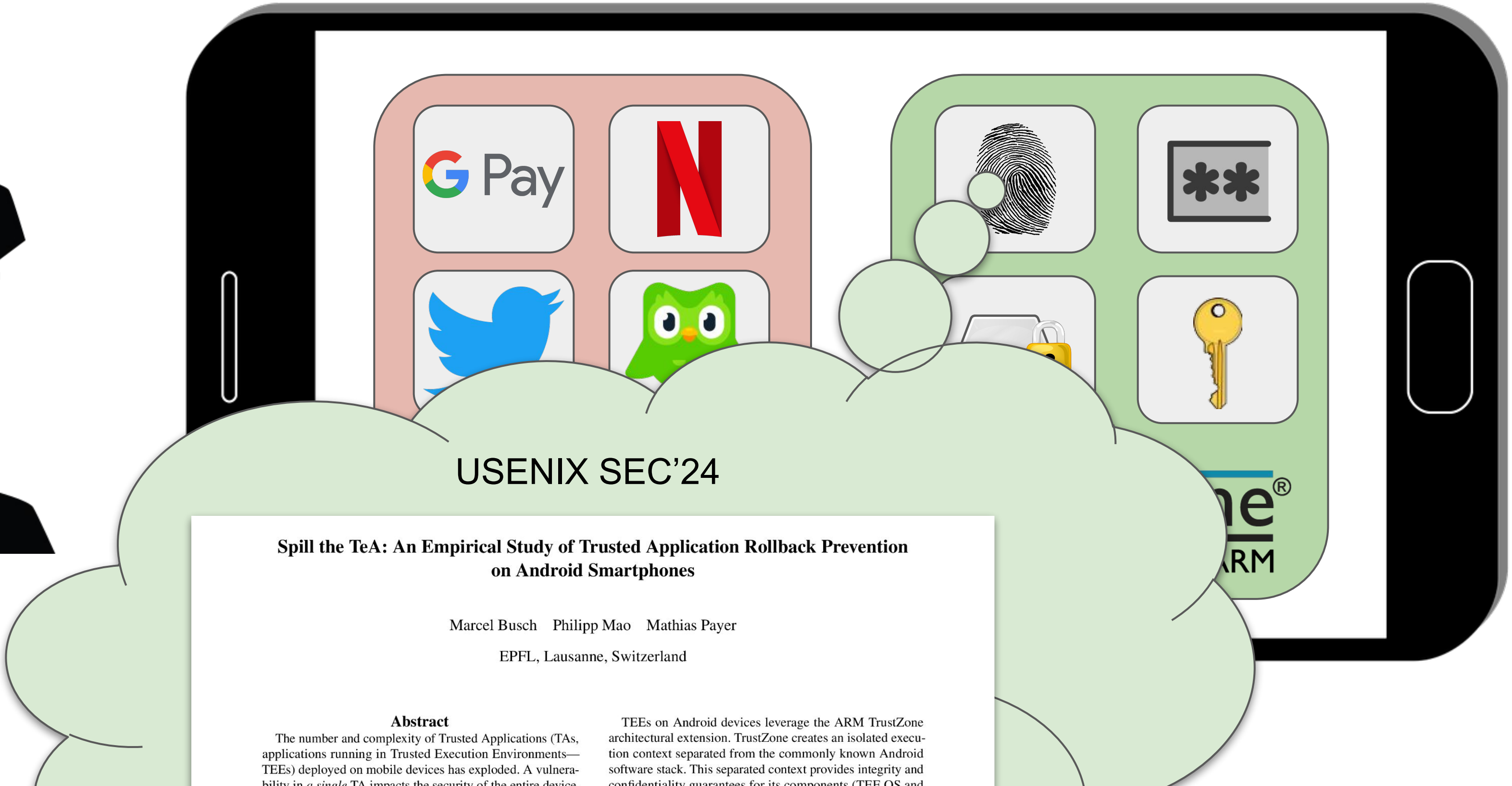
Marcel Busch, Philipp Mao, Mathias Payer
EPFL, Lausanne, Switzerland

HexHive

EPFL

Trusted Applications on Mobile Devices





USENIX SEC’24

Spill the TeA: An Empirical Study of Trusted Application Rollback Prevention on Android Smartphones

Marcel Busch Philipp Mao Mathias Payer

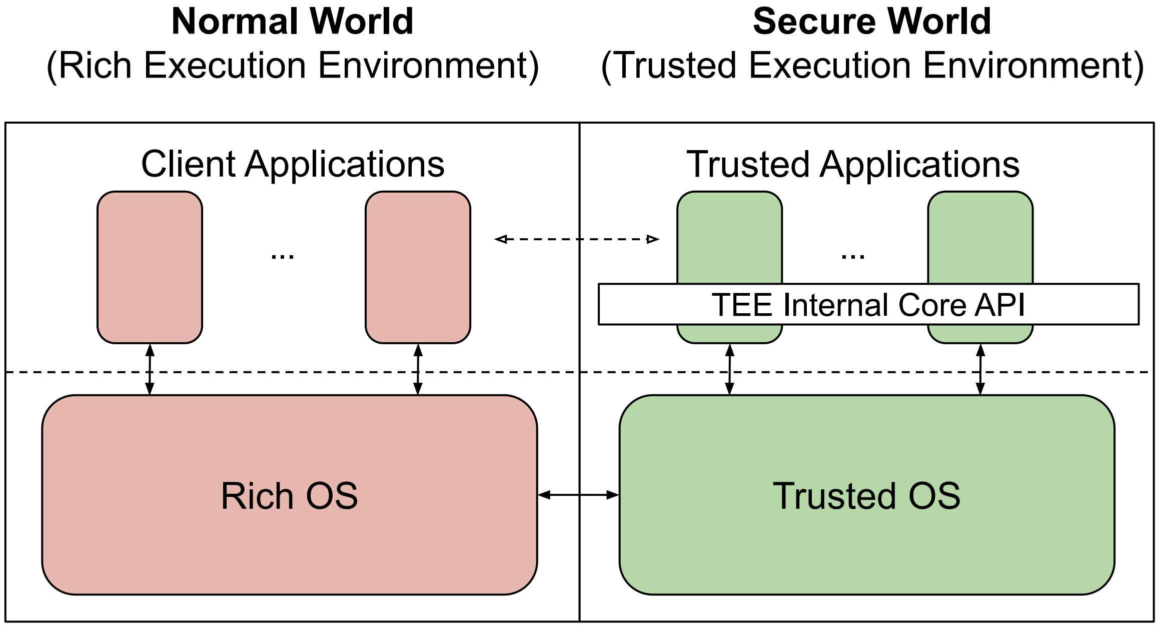
EPFL, Lausanne, Switzerland

Abstract

The number and complexity of Trusted Applications (TAs, applications running in Trusted Execution Environments—TEEs) deployed on mobile devices has exploded. A vulnerability in *a single* TA impacts the security of the entire device.

TEEs on Android devices leverage the ARM TrustZone architectural extension. TrustZone creates an isolated execution context separated from the commonly known Android software stack. This separated context provides integrity and confidentiality guarantees for its components (TEE OS and

TEE Fragmentation and the GlobalPlatform API



GLOBALPLATFORM®

THE STANDARD FOR SECURE DIGITAL SERVICES AND DEVICES

GlobalPlatform Technology

TEE Internal Core API Specification

Version 1.3.1

Public Release

July 2021

Document Reference: GPD_SPE_010

```
TEE_Result TA_InvokeCommandEntryPoint(void *sessCtx, uint32_t cmdId,
                                      uint32_t paramTypes, TEE_Param params[4])
{
```

Stores session state

Chooses TA cmd handler

Determines types of params

Four TEE_Param parameters

OPTIONAL

```
    u                    = TEE_PARAM_TYPES(
                           AM_TYPE_MEMREF_INPUT,
                     TEE_PARAM_TYPE_MEMREF_OUTPU
                     TEE_PARAM_TYPE_NONE,
                     TEE_PARA

    if (paramTypes !=        paramTypes)
                           RAMETERS;

    size_t in_buf_sz    = params[0].m
    char *in_buf        = params[0].m
                                      ef.size;
                                      ef.buffer;

                                      f_sz);
```

```
typedef union {
    struct {
        void *buffer;
        uint32_t size;
    } memref;
    struct {
        uint32_t a;
        uint32_t b;
    } value;
} TEE_Param;
```

Prevalence Study

Dataset

TOP 5

Android phone vendors

> 65% market share

545 firmware images

54 devices
5 different TEEs

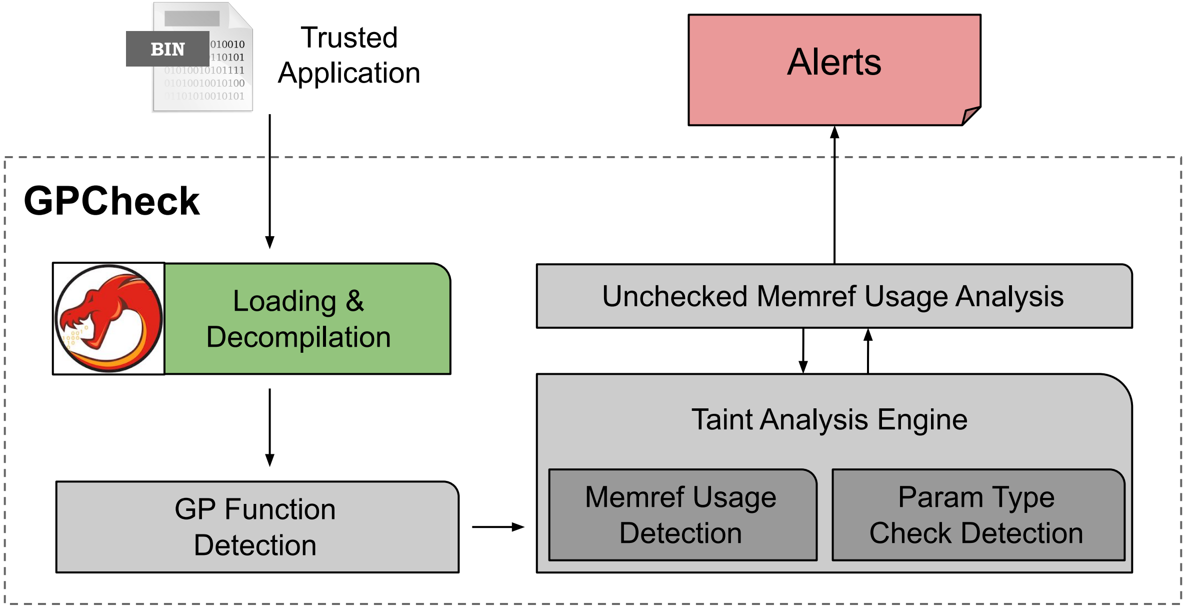
2016 - 2024

ELF

~14,500 proprietary TAs

GPCheck

- Ghidra-based
- Post-production vetting
- Open-Source



Trusted Application

Alerts

GPCheck

Loading & Decompilation

GP Function Detection

Unchecked Memref Usage Analysis

Taint Analysis Engine

Memref Usage Detection

Param Type Check Detection

Results

~6,900 TAs are GP-compliant (~131 unique TAs)

850 vulnerable TAs (33 unique vulnerable TAs)

9 publicly known

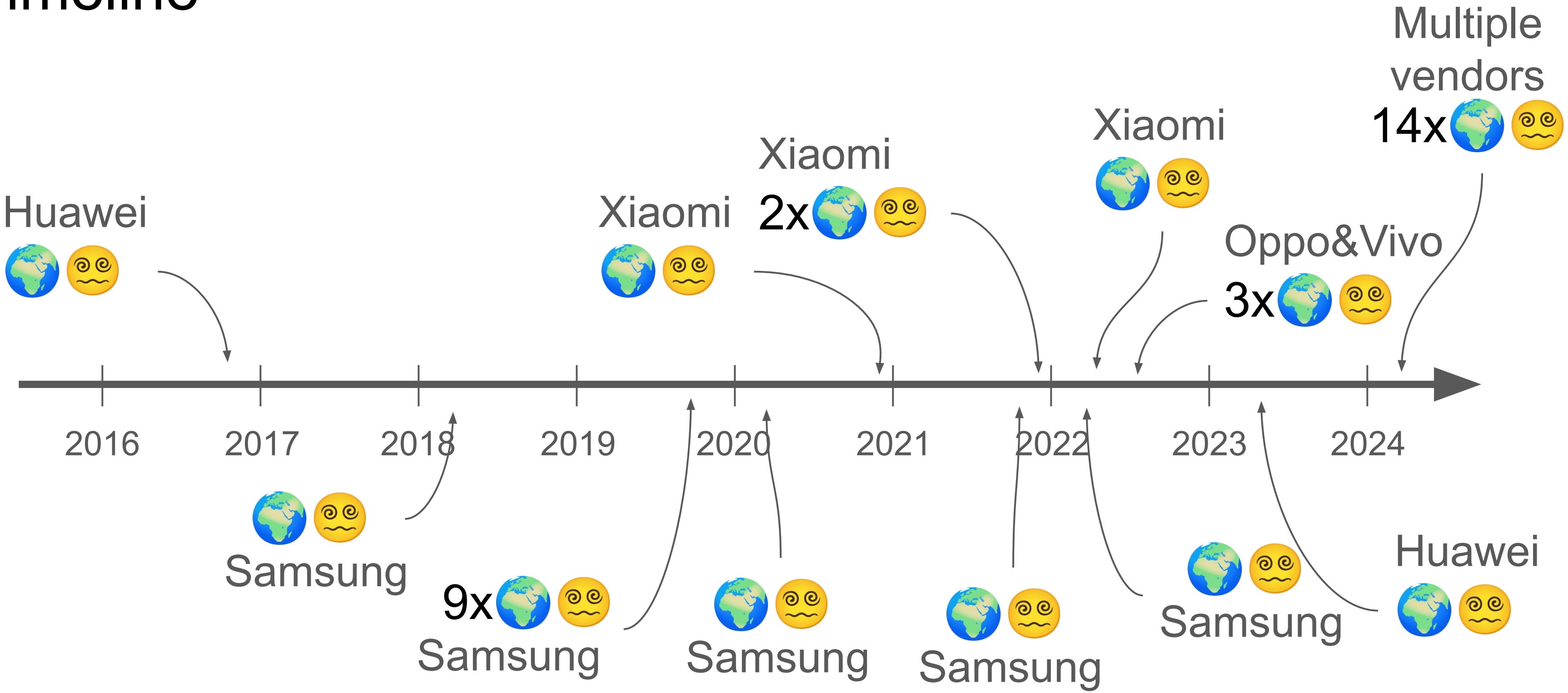
10 silently patched

14 0-days

CVE-2023-32835, CVE-2023-32834, CVE-2023-32848, CVE-2024-20078, …

> $ 12k bug bounty

Timeline



Huawei 🌍😵‍💫

🌍😵‍💫 Samsung

9x🌍😵‍💫 Samsung

Xiaomi 🌍😵‍💫

🌍😵‍💫 Samsung

Xiaomi 2x🌍😵‍💫

🌍😵‍💫 Samsung

Xiaomi 🌍😵‍💫

🌍😵‍💫 Samsung

Oppo&Vivo 3x🌍😵‍💫

Huawei 🌍😵‍💫

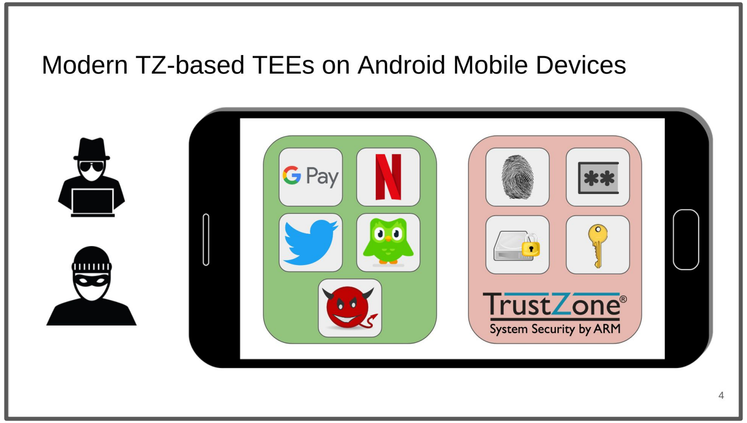
Multiple vendors 14x🌍😵‍💫

Mitigation

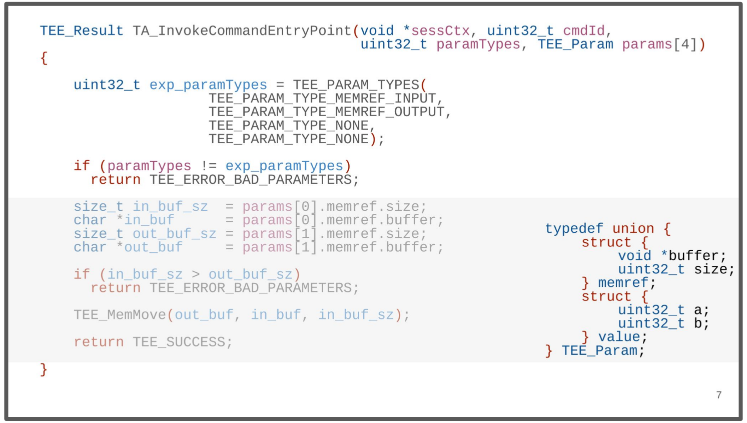
- Change fail-open to fail-close design
  - Mandatory type check
  - Fail-safe abort without proper check
- Sent proposal to GP; Draft for API update in progress
- No changes to external API (backwards compatible)

- Open-source and based on OPTEE

🌍😵‍💫 GlobalConfusion



Modern TZ-based TEEs on Android Mobile Devices



```
TEE_Result TA_InvokeCommandEntryPoint(void *sessCtx, uint32_t cmdId,
                                      uint32_t paramTypes, TEE_Param params[4])
{

    uint32_t exp_paramTypes = TEE_PARAM_TYPES(
                TEE_PARAM_TYPE_MEMREF_INPUT,
                TEE_PARAM_TYPE_MEMREF_OUTPUT,
                TEE_PARAM_TYPE_NONE,
                TEE_PARAM_TYPE_NONE);

    if (paramTypes != exp_paramTypes)
      return TEE_ERROR_BAD_PARAMETERS;

    size_t in_buf_sz  = params[0].memref.size;
    char *in_buf      = params[0].memref.buffer;
    size_t out_buf_sz = params[1].memref.size;
    char *out_buf     = params[1].memref.buffer;

    if (in_buf_sz > out_buf_sz)
      return TEE_ERROR_BAD_PARAMETERS;

    TEE_MemMove(out_buf, in_buf, in_buf_sz);

    return TEE_SUCCESS;
}
```

```
typedef union {
    struct {
        void *buffer;
        uint32_t size;
    } memref;
    struct {
        uint32_t a;
        uint32_t b;
    } value;
} TEE_Param;
```

Paper

Code

ARTIFACT EVALUATED usenix ASSOCIATION AVAILABLE

ARTIFACT EVALUATED usenix ASSOCIATION FUNCTIONAL

ARTIFACT EVALUATED usenix ASSOCIATION REPRODUCED

marcel.busch@epfl.ch

𝕏@0ddc0de

EPFL

This presentation has been designed using resources from Flaticon.com