Securing mobile applications using Trusted Execution Environment

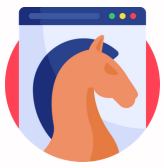
By Raja Ben Ali - 2020-2021 - AIRE M1 DiSc track
Remote, from 15-03-21 to 31-08-21, GIRY SAS startup
Supervisor: Researcher Myrto Arapinis

Increasing Convenience Heightens Threat Levels

5.22 billion
Unique mobile users worldwide in 2021.

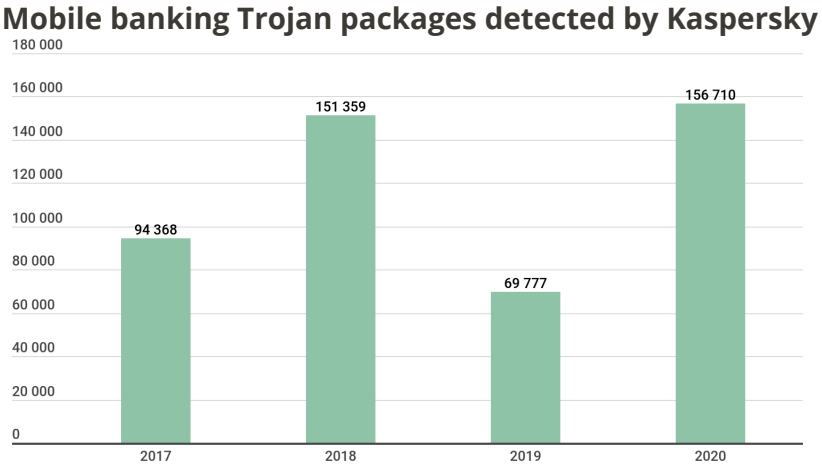
"The owner of a smartphone with a banking application on board is a WALKING WALLET"(1)

79 %
Smartphone owners have used their device for an online purchase in the past six month.



25,314 packages
Were related to mobile banking Trojans.

Mobile banking Trojan packages detected by Kaspersky (2)

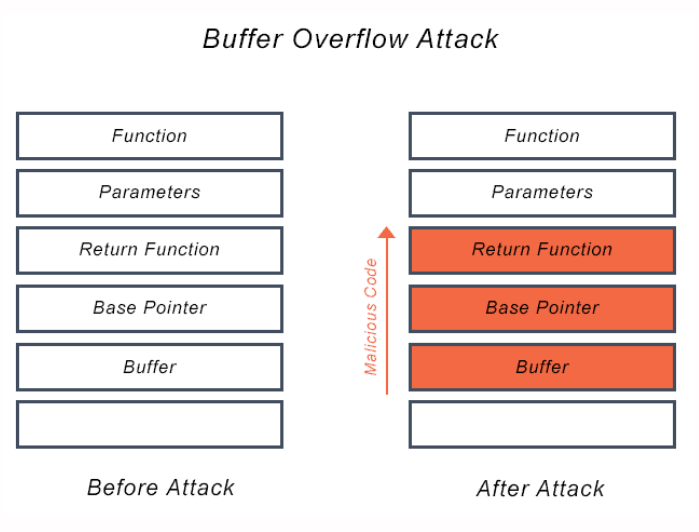


Summary: a security research project around users' security-sensitive operations inside mobile devices with a focus on the Samsung's implementation of the Trusted Execution Environment technology.

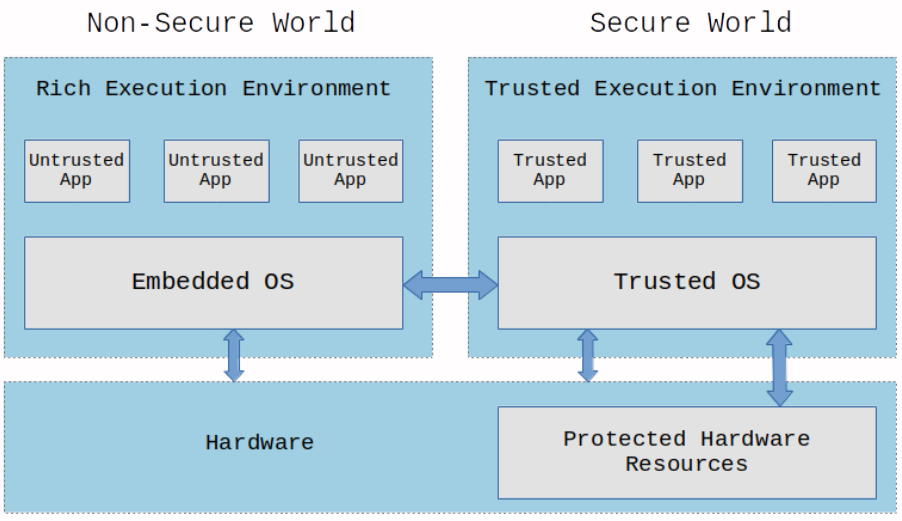
Hypothesis: Samsung's implementation of TEE is feature-rich so there might be more code vulnerabilities and architecture flaws than Google's. As Samsung's TEE is more popular, it will give more open doors for exploitation.

1. Bibliographic research: Privilege escalation & TEE apps

2. Code vulnerabilities experimentations in a virtual environment(3)



3. Experimentation on Samsung S20 device, TEEGRIS OS (4)



Results

- **For the project:**
  - Exploitation of C programs with buffer overflows.
  - 2 Trusted Applications in Google's TEE but +40 Trusted Applications in Samsung TEE.
  - Before Samsung S10, only one security measure (eXecute Never). S10 has numerous measures to bypass (XN, Canaries, ASLR & KASLR, ...). S20 might have more.
- **For myself:**
  - Deeper understanding of mobile phone security mitigations, memory security countermeasures.

Future results
After the exploitation of a Samsung S20 device.
Newer device = More mitigations to bypass = Need to chain vulnerabilities.

Organization

- **GIRY team & TEE projects:**

Google pixel TEE, Blockchain for finance, e-voting, and mobile phone security.
Weekly meetings and presentations. Google vs Samsung discussion.

Challenges

- **Theory vs practice gap:**

Understanding how to find and exploit vulnerabilities in stack memory.

- **Dense and continuous learning phase**

Why use the TEE? OS security, TEE implementation, and limitations.

Learning experience & Future plans

Background in C & C++ programming:
Syscalls, games, complex programs
+ using GNU debugger to exploit memory vulnerabilities

Future Research & Experimentations:
Memory vulnerabilities + Pen-testing methods

Certification:
Pen-testing tools + mobile phone security + Samsung certifications

Articles References
Security analysis of Samsung's TEEGRIS TEE OS

Bibliography

(1) https://www.tripwire.com/state-of-security/security-data-protection/android-banking-trojans-history-types-modus-operandi/
(2) https://securelist.com/mobile-malware-evolution-2020/101029/ (3) https://avinetworks.com/glossary/buffer-overflow/
(4) https://azeria-labs.com/trusted-execution-environments-tee-and-trustzone/