Global Platform®

May 2026

Attack Expert Sub-TF and Certification at GlobalPlatform

Jeremy O’Donoghue, Chair of Attack Expert Sub-TF

Attack Expert Sub-TF

Membership: Any GlobalPlatform member (no geographic or membership class restrictions)

- Hardware and software solutions developers
- Laboratories and Certification Bodies
- National Bodies and other schemes invited
- Liaisons with other working groups and standards
- Additional requirements for participation: 2 max. per member, suitable expertise, NDA and public key

Main goals of SAEWG

- Guidance to developers on attacks to be considered in the development of a solution
- Guidance to security assessors in laboratories who create vulnerability analysis campaigns for the security assessment of solutions.
  - Note – this is always guidance; the assessor must operate independently
- Support the security of solutions, and the quality and comparability of security assessments

Deliverables

Attack catalog for both SW and HW attacks

- Attacks to be considered with **guidance** for developers and evaluators to identify which is applicable per type of product, how and when (which context)
- All type of solutions and context

Attack ratings

- Method to rate successful vulnerability exploitation meaningfully considering **software attacks specificities**
- Associate to different **assurance levels**
- Method to handle when not possible to demonstrate the **exploitability**

Vulnerability assessment process

- Steps for software vulnerability **assessment** (as not sequential as classic ones)
- Method to handle in a limited time the important amount of code to analyze, the complexity of the code, the large list of public vulnerability to be checked, etc. (effort in relation with assurance levels)

Methodology

Attacks are rated using the JHAS methodology described in EUCC Scheme State-of-the-art Document “Application of Attack Potential to Smartcards and Similar Devices”

- This document draws heavily on Common Criteria CEM (Annexe B), but with changes in how attack rating is performed.
- Different rating tables and values for Attack Potential
- AAPS splits rating between “Identification Phase” and “Exploitation Phase”. CEM combines these.

CEM, Table B.3

Table B.3 — Rating of vulnerabilities and TOE resistance

Values	Attack potential required to exploit scenario:	Meets assurance components:	Failure of components:
0-9	Basic	-	AVA_VAN.1, AVA_VAN.2, AVA_VAN.3, AVA_VAN.4, AVA_VAN.5
10-13	Enhanced-Basic	AVA_VAN.1, AVA_VAN.2	AVA_VAN.3, AVA_VAN.4, AVA_VAN.5
14-19	Moderate	AVA_VAN.1, AVA_VAN.2, AVA_VAN.3	AVA_VAN.4, AVA_VAN.5
20-24	High	AVA_VAN.1, AVA_VAN.2, AVA_VAN.3, AVA_VAN.4	AVA_VAN.5
=>25	Beyond High	AVA_VAN.1, AVA_VAN.2, AVA_VAN.3, AVA_VAN.4, AVA_VAN.5	-

AAP Smartcard, Table 14

Values	ToE Resistant to attackers with attack potential of
0-15	No rating
16-20	Basic
21-24	Enhanced-Basic
25-30	Moderate
=> 31	High

Problems with the JHAS methodology

- Very Smartcard-focused (unsurprising, as this is what it was designed for)
  - Smartcards often protect shared symmetric keys – loss of key is catastrophic for a class of devices.
  - Strong focus on physical attacks (FI in particular) as SW is not complex.
  - Few other established/accepted methodologies means it is applied in places it was never intended to be used.
- Features of TEEs and many other SoC architectures
  - Unique device identity and use of public key crypto – loss of key breaks only one device and is not catastrophic.
    - Unless a method for then easily recovering keys from other devices is found.
    - Scaling of attacks is catastrophic
  - Generally, remote attacks of much greater interest than physical attacks (devices are connected)
- Smartcards often not updateable (or not easily); connected devices often regularly patched
- How to perform ”software-only” evaluations
- Multi-part attacks not really considered

Rating Factors – CEM vs AAP Smartcard

Factor	CEM	AAPS (Identification)	AAPS (Exploitation)	Notes
Elapsed Time	1 (< 1w) .. 19 (> 6m)	1 (<1 d) .. 6 (>4 m)	3 (<1d) .. 10 (>4m)	Also “Not practical” – cannot exploit in useful time.
Expertise	0 (layman) .. 8 (multiple experts)	0 (layman) .. 7 (multiple experts)	0 (laymen) .. 6 (multiple experts)	
Knowledge of ToE	0 (public) .. 11 (critical)	0 (public) .. 9 (critical+)	0 (public) .. 5 (critical)	
Window of Opportunity	0 (unlimited) .. 10 (difficult)			
Access to ToE		0 (<10 samples) .. 3 (>= 100 samples)	0 (<10 samples) .. 6 (>= 100 samples)	No of samples mainly about destructive physical attacks; not really relevant to remote attacks.
Package Removal		0 (low) .. 2 (high)	0 (low) .. 4 (high)	Added to “Access to ToE in practice”. Not relevant to remote attacks.
Equipment	0 (standard) .. 9 (multiple bespoke)	1 (standard) .. 7 (multiple bespoke)	2 (standard) .. 8 (multiple bespoke)	Standard “< 10k Euro”. Bespoke “>200k Euro)
Open Samples		0 (not needed) .. 9 (critical)		Basically, devices with known secrets

Rating applicability concern

Typical complex SW attack rating => will always end up with a Basic rating with JIL

Only access may vary

	SmartCard Attack Potential		CEM	
	Identification	**Exploitation**		
Elapse time	8 *(>1 month)*	**0** *(<1 hour)*	**19** *(> 6 month)*	Considering full exploit done and published
Expertise	5 *(expert)*	0 *(layman)*	6 *(expert)*	
TOE knowledge	**0** *(public / restrict.)*	**0** *(public)*	**0** *(public)*	Will mostly be public
TOE access / Window of opportunity	0 *(easy)*	0 *(easy)*	0 / 1 *(unlimited / easy)*	
Equipment	1 *(standard)*	2 *(standard)*	0 *(standard)*	Will mostly always be standard
Open samples / Secret knowlege	0 / 2 *(Public / Restricted)*	NA	-	Could be adapted?
Rating	14 / 16 **16 / 18** *(Basic)*	2	**26** *(High)*	

Even very complex attacks will be rated Basic, no granularity

Only the time reflects the attack complexity

GLOBALPLATFORM®

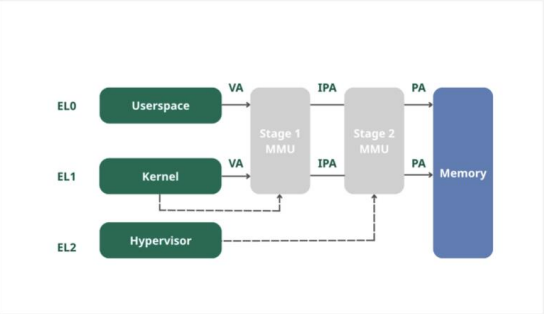
Global Platform®

Software Attacks

Software-only evaluation?

Operational Environment Assumptions

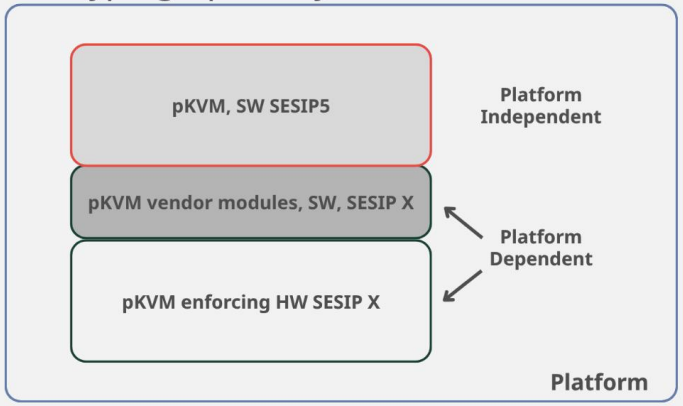
- HW: “The hardware where the platform is deployed shall provide security exception levels and memory management making it possible to implement memory and execution protections.”



- Physical attacks: “The platform shall only be deployed in environments where physical attacks protections are not required.”

Only Logical Attacks Considered

- SW target, “platform independent”.
- Main security functions are isolation and access control.
- No TOE assets, such as cryptographic keys.



Slide credit: ICCC ‘25 – pKVM SESIP 5 Evaluation: Where do we go from here?

“Meaningful” software attack levels (1)

What is needed to find a vulnerability and build an exploit?

- Time
  - Complexity of implementation (language, obfuscation technics, etc.) => for extrapolation?
  - Integrate reverse engineering so no need of knowledge of the TOE? (not exactly that in CC)
  - Agentic AI (new topic)
- Expertise, 2 aspects
  - Global expertise in software analysis (techniques and tools) for the targeted field (reverse, web, etc.)
  - Prior specific expertise on the type of product (e.g. specific developer product architecture/countermeasures)
    Different from knowledge of the TOE (availability of information)
- Tooling (see next slide) – different from HW equipment
- Environment pre-conditions => is this really to be part of rating?
  - Local access constraint (e.g. USB, etc.) => attack easiness/probability
  - Privileges for vulnerability “access” => attack easiness
  - Victim action dependency (e.g. user authentication, application/file opening, etc.) => attack probability
  - Other resources dependency (e.g. specific software, specific CPU, etc.) => product hypothesis
- Risk analysis - Not really a factor in JHAS

“Meaningful” software attack levels (2)

Resistance to low attacks: low expertise on target, limited time

- Basic reverse engineering/code analysis
- Basic fuzzing (e.g. random/mutation based) of interfaces, protocols, formats, etc.
- Publicly accessible tools and scripts

⇒ Obvious vulnerabilities (e.g. simple memory corruptions / protocols issues/ race conditions, etc.)

Resistance to medium attacks: significant expertise on target, limited time

- Advanced analysis: limited time on memory leaks, race conditions, obvious design issues, etc.
- Advanced fuzzing (e.g. model based)
- Home made/adapted tools and scripts

⇒ Vulnerabilities requiring good implementation understanding (e.g. complex memory corruptions…)

Resistance to advance attacks: high expertise on target, significant time

- Deeper analysis: more time, more expertise in analysis and tools development

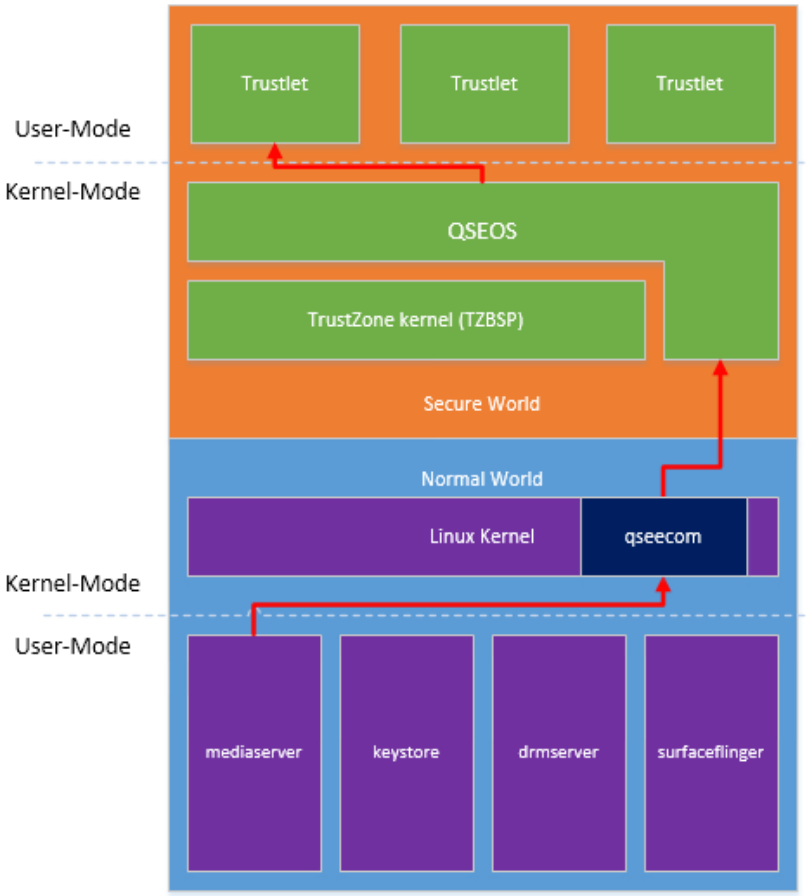
⇒ Vulnerabilities requiring deep implementation understanding

Global Platform®

Worked example

TEE Exploit

Example: Multi-phase TEE exploit (CVE-2016-24131)

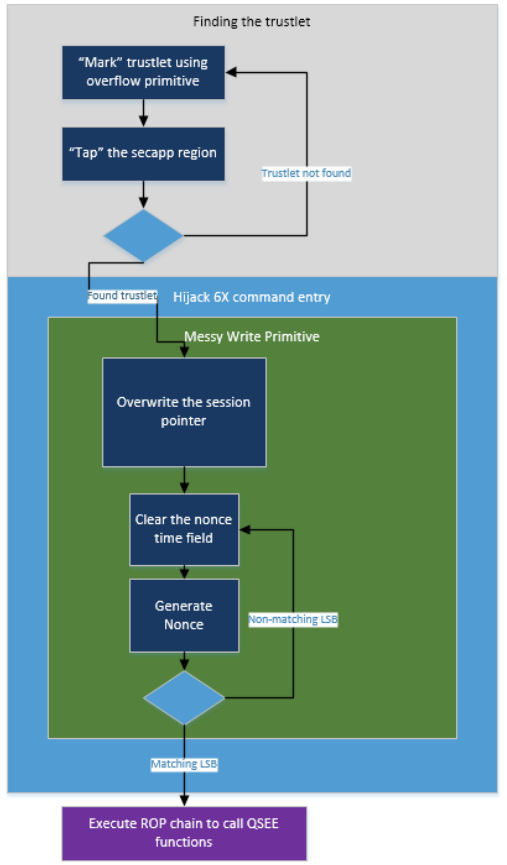


Exploit for Qualcomm QTEE.

- QSEECom is a device driver allowing user-mode Android processes to invoke operations in the TEE.
- Client Applications use daemon-like functionality to communicate with QSEECom (c.f. TEE Client API)
- Access to QSEECom restricted by SELinux policy to four processes.
- Assume that “mediaserver” is compromised (see CVE-2015-6639)
- The exploit depends on a TA (“Trustlet”) vulnerability to take control of the kernel.

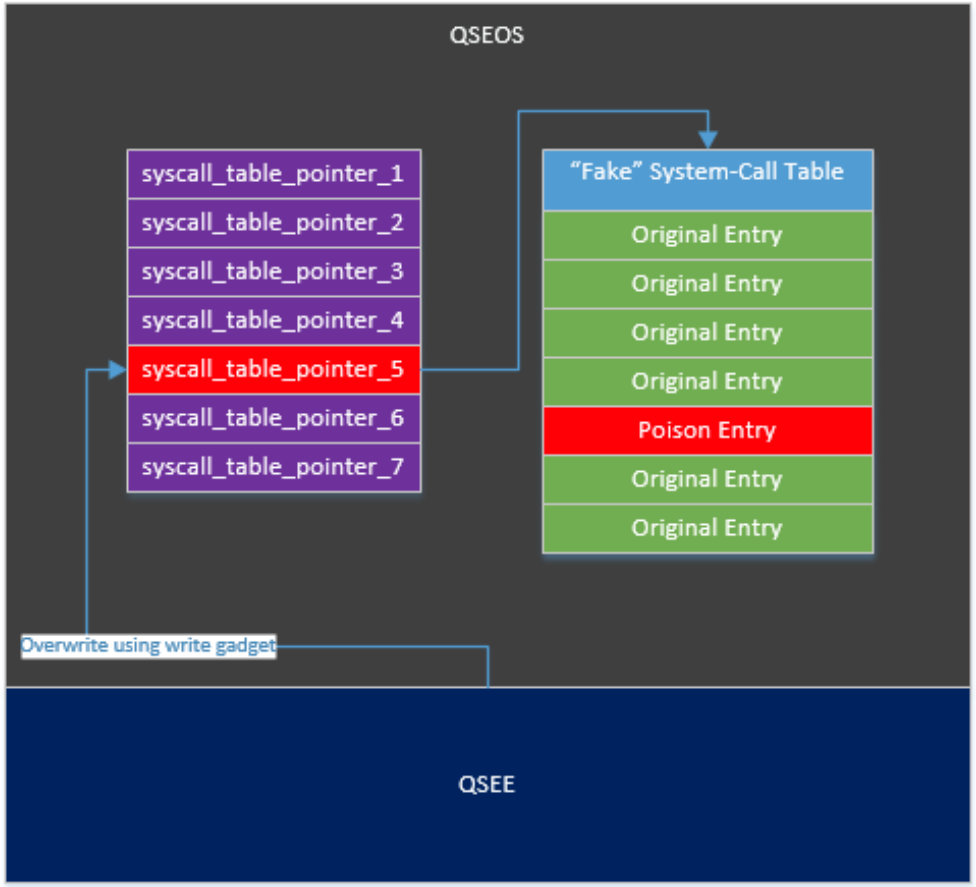
“Shellcode” machine in secure userland

- Find Widevine TA by repeatedly "tapping" the secapp region and "listening”
  - **Vulnerability #1**
- Create (uncontrolled) write gadget using the nonce-generation command. Allows write of partially-controlled data at controlled address
  - **Vulnerability #2**
- Overwrite unused 6X command entry using write gadget to direct it to a stack-pivot
  - **Vulnerability #3**
- Execute any arbitrary code using a small ROP chain under the "Universal Shellcode Machine” – requires attacker control of CPSR to read Thumb instructions as Arm



Take control of QTEE Kernel

- Allocate a "fake" syscall table in QSEE
  - **Vulnerability #4**
- Use the write primitive to overwrite the syscall table pointer to point to our crafted "fake" syscall table
- Set the single "poison" syscall entry in the "fake" syscall table to point to the DACR-modifying function in the TrustZone kernel
- Invoke the "poison" syscall in order to call the DACR-modifying function in the TrustZone kernel - thus setting the DACR to 0xFFFFFFFF
- Use the write gadget to write our shellcode directly to a code page in QSEE belonging to our QSEE application
- Invalidate the instruction cache (to avoid conflicts with the newly written code)
- Set the single "poison" syscall entry in the "fake" syscall table to point to the written shellcode
- Invoke the "poison" syscall in order to jump to our newly-written shellcode from the context of the TrustZone kernel!



Attack complexity?

Identification phase

- 4 vulnerabilities identified and exploited
  - Individually they reach 6-8 points for identification phase (<= 1 week or 1 month elapsed time, 1 sample, 1 expert and standard equipment)

Factor	Identification	Exploitation
Elapsed time	5 (> one month)	0 (<= one hour)
Access to TOE	0 (Only 1 sample)	0 (Only 1 sample)
Expertise	5 (Expert)	0 (Layman)
Knowledge of the TOE	0 (Public)	0 (Public)
Equipment	1 (Standard)	2 (Standard)
Open samples	0 (Public)	NA
Total	11	2
Grand total	13	

Table 36: CVE-2015-6639 attack rating

Successful remote exploits of TEEs in the field are rare. Almost every example exploits multiple vulnerabilities.

- Most vulnerabilities turn out not be be exploitable.
- “Useful” vulnerabilities are used to construct Attack Primitives.

In general, Attack Primitives offer a clear path to an exploit.

- Unless the attacker can control what is executing, the likely consequence of a vulnerability is a crash, which does not expose sensitive data.

Global Platform®

Securing the digital future

→globalplatform.org