TEEP Architecture
draft-ietf-teep-architecture-16

Dave Thaler (presenting)

Ming Pei, David Wheeler, Hannes Tschofenig

Timeline

- JAN 2020: WGLC completed
- JUL 2021: Draft-15 submitted to IESG
- JAN 2022: AD (Ben) feedback
  - https://mailarchive.ietf.org/arch/msg/teep/QJCjyUP-0vErQODB5-_PkvrKMXc/
- FEB 2022: Updated draft-16 to address AD feedback
  - https://mailarchive.ietf.org/arch/msg/teep/eW5OokuMPYGIIdRGKn1vbF14uQs/

“Double-edged sword” (1/3)

Ben’s comment:

- The whole architecture is in some sense a "double-edged sword". It provides protection for users and device owners against malicious apps running on the device, at the cost of the owner having to trust that the TAM is providing the stated code.
- The owner doesn't necessarily have good mechanisms for getting visibility into what code is actually running in the TEE without being involved in the TAM operation, and in some cases **the user will have no rights at all**. The latter **runs risk of conflicting with RFC 8890 [“The Internet is for End Users”]**, so I really think we want some discussion of how there are ways for TEEP to provide value to users, acknowledge that there are cases where the **main value of TEEP is to device owners potentially at risk of harm to users, and discuss the trade-offs involved**.
- This architecture basically requires that the TAM know the device identity in all transactions (device identifying information is a required claim in §7); this has privacy implications that should be documented. The TAM is a trusted party in the ecosystem, but can still be a different party than device owner or administrator, so we need to **document the new privacy exposure** (and possibly the assumption that contractual controls will be available for it).

Added intro text to clarify use cases (2/3)

- **TEEs are typically used in cases where a software or data asset needs to be protected from unauthorized entities that may include the owner (or pwner) or possesser of a device**.
- This situation arises for example in:
  - gaming consoles where anti-cheat protection is a concern,
  - devices such as ATMs or IoT devices placed in locations where attackers might have physical access,
  - cell phones or other devices used for mobile payments, and
  - hosted cloud environments.
- Such environments can be thought of as hybrid devices where **one user or administrator controls the REE where a different (remote) user or administrator controls a TEE in the same physical device**.
- It may also be the case in some constrained devices that there is no REE (only a TEE) and **there may be no local "user" per se, only a remote TEE administrator**. For further discussion of such confidential computing use cases and threat model, see [CC-Overview] and [CC-Technical-Analysis].

Privacy considerations (3/3)

- The TEEP architecture is applicable to cases where devices have a **TEE that**
- **protects data and code from the REE administrator**. In such cases, the TAM
- administrator, not the REE administrator, controls the TEE in the devices.   As
- some examples:
-
- * a cloud hoster may be the REE administrator where a customer
-    administrator controls the TEE hosted in the cloud.
- * a device manufacturer might control the TEE in a device purchased by a
-    customer
-
- The privacy risk is that **data in the REE might be susceptible to disclosure to**
- **the TEE administrator**. This risk is not introduced by the TEEP architecture,
- but is inherent in most uses of TEEs. This risk can be mitigated by making
- sure the **REE administrator is aware of and explicitly chooses to have a TEE**
- **that is managed by another party**. In the cloud hoster example, this choice
- is made by explicitly offering a service to customers to provide TEEs for
- them to administer. In the device manufacturer example, this choice is
- made by the customer choosing to buy a device made by a given manufacturer.

DoS possible by REE (orig. on transport draft)

- "Some implementations might rely on (due to lack of any available alternative) the use of an untrusted timer or other event to call the RequestPolicyCheck API (Section 6.2.1), which means that **a compromised REE can cause a TEE to not receive policy changes and thus be out of date with respect to policy**. The same can potentially be done by any other man-in-the-middle simply by blocking communication with a TAM. Ultimately such outdated compliance **could be addressed by using attestation** in secure communication, where the attestation evidence reveals what state the TEE is in, so that communication (other than remediation such as via TEEP) from an out-of-compliance TEE can be rejected.
- Similarly, in most implementations the REE is involved in the mechanics of installing new TAs. However, the authority for what TAs are running in a given TEE is between the TEEP Agent and the TAM. While **a TEEP Broker can in effect make suggestions, it cannot decide or enforce what runs where**. The TEEP Broker can also control which TEE a given installation request is directed at, but a TEEP Agent will only accept TAs that are actually applicable to it and where installation instructions are received by a TAM that it trusts.
- The authorization model for the UnrequestTA operation is, however, weaker in that it expresses the removal of a dependency from an application that was untrusted to begin with. This means that **a compromised REE could remove a valid dependency from an Untrusted Application on a TA. Normal REE security mechanisms should be used to protect the REE and Untrusted Applications**."

TEEP Agent validation of TAM HTTPS cert

- Ben: “ Might OCSP (including stapling) or other non-CRL mechanisms be in scope? Is it worth mentioning RFC 6960 or 6961 as well as 5280 here?”
- IETF 111 discussion about OCSP in TEEP, from minutes:
  - Russ made the argument that OCSP stappling might be difficult for constrained node. He was arguing that there are more lightweight solutions.
  - Ben: OCSP does not really make sense with COSE. You might just be using hard-coded keys and you might be updating keys with software updates. It is probably still worth to mention that there is a need for revocation.
- Added in draft-16:
  - “… See Section 6 of [RFC5280] for details. Such validation generally includes checking for certificate revocation, **but certificate status check protocols may not scale down to constrained devices that use TEEP**.”

Updated bundling cases

Different ways to package Untrusted App, TA, and Personalization Data:

1. [Untrusted App + TA + Personalization Data]
2. [Untrusted App + TA], [Personalization Data]
3. [Untrusted App], [TA], [Personalization Data]
4. **[Untrusted App], [TA + Personalization Data]**
5. **[Untrusted App + Personalization Data], [TA]**

Ben recommended adding 4-5 for completeness. Did so, and updated SGX and TrustZone sections to discuss applicability (or not). Roughly:

SGX: 2 is typical, 1/5 very complex no known cases

TZ: 2-4 applicable but 1/5 are complex

Other notable feedback

- ProcessConnect vs ProcessTeepMesage abstract API
  - Clarified: request to request a new session, vs process msg in existing session
- Use of same key for both signing and encryption
  - Ben: “joint security of encryption and signature with a single key remains to some extent an open question in academic cryptography”

```
> Typically the same key TEE pair is used for both signing and
> encryption, though separate key pairs might also be used in the
> future, as the joint security of encryption and signature with a
> single key remains to some extent an open question in academic
> cryptography.
```

    (and same point for TAM key pair)
- Various other editorial wordsmithings accepted as Ben suggested