TEEP Architecture draft-ietf-teep-architecture-08

Dave Thaler (presenting)

Ming Pei, David Wheeler, Hannes Tschofenig

April 6, 2020

Timeline

```
DEC 23, 2019 – WGLC started
JAN 20, 2020 – WGLC completed
FEB  8, 2020 – Draft -06 posted
FEB 10, 2020 – TEEP virtual interim meeting
MAR  8, 2020 – Draft -07 posted

Two new issues filed in github since -07
No new list discussion since -07
```

Next steps [slide from Feb interim meeting]

- Incorporate feedback from this meeting
- Post -07 before March 9 (ideally before end of Feb)
- Do a 2nd WGLC to finish before/at IETF 107?
- Goal is existing milestone:
  - “Apr 2020 - Progress Architecture document to the IESG for publication”

#158: TEE definition and "authorized" code

- Feedback came from Confidential Computing Consortium
- OLD (-07):
  - A Trusted Execution Environment (TEE) is an environment that enforces that **only authorized code can execute within that environment**, and that any data used by such code cannot be read or tampered with by any code outside that environment.
- NEW (-08):
  - A Trusted Execution Environment (TEE) is an environment that enforces that **any code within that environment cannot be tampered with,** and that any data used by such code cannot be read or tampered with by any code outside that environment.

#150: Devices with no REE, which are mentioned in Section 4.1 of Draft06

- Kuniyasu Suzaki (AIST) filed:
  - "Why does the TEEP protocol need to support such a device?
  - I cannot [imagine] the device which has no REE.
    I think the TEE just turns to be REE, and it is a normal device."
- Editors believe this is a continuation of issue #139 (Contradiction about whether a device must have an REE to use TEEP)
  - 139 was discussed in Feb interim meeting with agreed resolution that it should be supported
- Draft -07:
  - … In devices with no REE **(e.g., a microcontroller where all code runs in an environment that meets the definition of a Trusted Execution Environment in Section 2)**, the TEEP Broker would be absent and instead the TEEP protocol transport would be implemented inside the TEE itself.

#146: Section 9.1: Broker Trust Model does not mention DoS by dropping messages

Filed by Nicolae Paladi pointed out this omission, now fixed

Section 9.1 (Broker Trust Model):

- … As such, all TAM messages are signed and sensitive data is encrypted such that the TEEP Broker cannot modify or capture sensitive data**, but the TEEP Broker can still conduct DoS attacks as discussed in Section 9.3**. …

Section 9.3 (Compromised REE):

- It is possible that the REE of a device is compromised. If the REE is compromised, several DoS attacks may be launched. The compromised REE may terminate the TEEP Broker such that TEEP transactions cannot reach the TEE**, or might drop or delay messages between a TAM and a TEEP Agent**. However, while a DoS attack cannot be prevented, the REE cannot access anything in the TEE if it is implemented correctly. Some TEEs may have some watchdog scheme to observe REE state and mitigate DoS attacks against it but most TEEs don't have such a capability.

#148: TA attestation check frequency policy for compromised status check (1/3)

- Ming filed based on discussion at Feb. interim:
  - In the virtual interim TEEP meeting today, it is asked how often should the Attestation service should be checked. If we have a frequency as a policy, say, a TA attestation is valid for 1 week, or 1 month, who defines the policy: TAM that distributed the TA, or the TEEP agent in TEE or another entity that can define or update the policy.
- Discussion in Feb. interim:
  - - Should file an issue in RATS architecture, about how long an attestation result should be used.
  - HB: there will always be a delay, and the evidence may have changed during the evaluation.

#148: TA attestation check frequency policy for compromised status check (2/3)

- Section 9.6 (Malicious TA Removal) in draft-07 added:
  - … Furthermore the policy in the Verifier in an attestation process can be updated so that any evidence that includes the malicious TA would result in an attestation failure. **There is, however, a time window during which a malicious TA might be able to operate successfully, which is the validity time of the previous attestation result. For example, if the Verifier in Figure 5 is updated to treat a previously valid TA as no longer trustworthy, any attestation result it previously generated saying that the TA is valid will continue to be used until the attestation result expires. As such, the TAM's Verifier should take into account the acceptable time window when generating attestation results. See [I-D.ietf-rats-architecture] for further discussion.**
- RATS arch issue: https://github.com/ietf-rats-wg/architecture/issues/42

#148: TA attestation check frequency policy for compromised status check (3/3)

- Ming commented:
  - Just note that the question wasn't about how long an attestation result will be valid. The question is how often some entity reaches out a TA attestation service that the TA in the devices have not been compromised.
- Draft -08:
  - It may happen that a TA was previously considered trustworthy but is later found to be buggy or compromised. In this case, the TAM can initiate the removal of the TA by notifying devices to remove the TA (and potentially the REE or device owner to remove any Untrusted Application that depend on the TA). If the TAM does not currently have a connection to the TEEP Agent on a device, such a notification would occur the next time connectivity does exist. **That is, to recover, the TEEP Agent must be able to reach out to the TAM, for example whenever the RequestPolicyCheck API (Section 6.2.1) is invoked by a timer or other event.**
- Section 6.2.1 already had:
  - RequestPolicyCheck: A hint (e.g., based on a timer) that the TEEP Agent may wish to contact the TAM for any changes, without the device itself needing any particular change.

#132: Requirements on Personalization Data (1/3)

- Does *all* personalization data require confidentiality, or can there be device instance-specific data that only needs integrity?
- Text in -06 said:
  - The personalization data **must** be encrypted to preserve the confidentiality of potentially sensitive data contained within it. **Other than this requirement to support confidentiality, TEEP place no limitations or requirements on the personalization data.**
- Hannes proposed (PR #101) replacing with:
  - The personalization data **may need to** be encrypted to preserve the confidentiality of potentially sensitive data contained within it.

#132: Requirements on Personalization Data (2/3)

Ming commented:

- I think personalization data will be largely confidential to a TEE and also to a TA, basing on the original motivation of this. Therefore encryption support is needed.
- On the text change, we can say that personlization data should be encrypted as it is typically confidential data. "may be encrypted" doesn't enforce some recommendation here. As a requirement, the protocol MUST be able to support personalization data encryption.

Discussion at Feb. interim:

- RH: 'implementations must support encryption to allow for loading of sensitive personal data'

#132: Requirements on Personalization Data (3/3)

- OLD (-07):
  - **The personalization data must be encrypted** to preserve the confidentiality of potentially sensitive data contained within it. Other than this requirement to support confidentiality, the TEEP architecture places no limitations or requirements on the personalization data.
- NEW (-08):
  - **Implementations must support encryption of personalization data** to preserve the confidentiality of potentially sensitive data contained within it. Other than this requirement to support confidentiality, the TEEP architecture places no limitations or requirements on the personalization data.

Next steps

- Do a 2nd WGLC