TEEP Architecture
draft-ietf-teep-architecture

Dave Thaler (presenting)

Ming Pei, David Wheeler, Hannes Tschofenig, et al.

Normative Language (1/2)

- Issue:
  - #69 Should arch draft use normative language?
    - Also part of #70 Juergen's feedback
- Draft currently is intended status Informational, but uses 2119 terms
- Should it be Informational or Standards Track?
- If Informational, should it use MUST/SHOULD/MAY terms?
  - If so, is the audience implementers or spec writers?
  - If spec writers, which specs?
- Proposal:
  - Informational
  - Remove RFC 2119 terms

Normative Language (2/2)

- Current example uses directed at spec writers:
  - The Attestation Header SHALL identify the "Attestation Type" and the "Attestation Signature Type" along with an "Attestation Format Version Number."
  - The claims themselves SHALL be defined in an attestation claims dictionary.
  - That registry SHALL be defined in the OTrP protocol …
- Current example uses directed at implementers:
  - Any intermediate CA for TEE device certificates SHOULD be validated by TAM with a Certificate Revocation List (CRL) or Online Certificate Status Protocol (OCSP) method.
  - The TEE SHOULD use validation of the supplied TAM certificates and OCSP stapled data to validate that the TAM is trustworthy.

REE Terminology

- Issues:
  - #53 Abstract, p.1: "regular operating system"
  - #54 Introduction, pp.4,6: regular/normal/typical operating system
  - #55 Introduction, pp.4, 5: "untrusted application" vs "client application“
    - Also part of #70 Juergen’s feedback
- Proposal (pull request #71)
  - “regular/rich/normal/typical operating system” -> “Rich Execution Environment” or “commodity operating system”, depending on context
  - “Client Application” -> “Untrusted Application”
    - Since the app may be a Server for some protocol

TEE Terminology (1/2)

- Issue:
  - #68 TEE Definition
- Previous: “An execution environment that runs alongside of, but is isolated from, an REE.”
- Arguments:
  - “Isolated” implies to the filer that TEE<->REE can’t communicate, vs “separated”
  - Some TEEs have no REE on the same device
- Pull request #71’s definition is:

  An environment that provides hardware enforcement that
  - (1) only authorized code can execute within the TEE, and
  - (2) data used by that code cannot be read or tampered with by code outside the TEE.

TEE Terminology (2/2)

- Other aspects not part of the definition?
  - **REE existence:** some environments (e.g., MCUs) can meet both properties, with no REE. Are they not TEEs?
  - **TEE OS existence:** some TEEs have a trusted OS, some don't
  - **Hardware protection:** there are less secure environments that are just rooted in say a hypervisor, that enforce the two properties in "TAs". Are they TEEs or not?
- TEEP charter:
  - The Trusted Execution Environment (TEE) is **a secure area of a processor**.
  - The TEE provides security features such as **isolated execution and integrity of Trusted Applications, along with provisions for maintaining the confidentiality of their assets**.
  - In general terms, the TEE offers an **execution space that provides a higher level of security than a "rich" operating system and more functionality than a secure element**.
  - For example, implementations of the TEE concept have been developed by ARM and Intel, using the TrustZone and the SGX technology, respectively.

Issues relevant to SUIT

Dependencies on/from TAs (1/2)

- Issues:
  - #13: Is it in scope: TA depends on another TA and related installation?
  - #34: Version dependencies between TA and normal world app
  - #35: Coordinate TA updates with UA
- Had previous WG consensus to use SUIT manifest for dependencies from TA's
  - Draft already has text talking about Untrusted App manifest expressing dependencies on TA's
- Proposal (pull request #75):
  - Add reference to SUIT manifest and explain it expresses dependencies from TAs
  - Add discussion of compatibility issues when updating a dependency

Dependencies on/from TAs (2/2)

Separate from the Client App's manifest, this framework relies on the use of the manifest format in [I-D.ietf-suit-manifest] for expressing how to install the TA as well as dependencies on other TEE components and versions. That is, dependencies from TAs on other TEE components can be expressed in a SUIT manifest, including dependencies on any other TAs, or trusted OS code (if any), or trusted firmware. Installation steps can also be expressed in a SUIT manifest.

…

Updating a TA may cause compatibility issues with any Untrusted Applications or other components that depend on the updated TA, just like updating the OS or a shared library could impact an Untrusted Application. Thus, an implementation needs to take into account such issues.

Security Domains (1/2)

- Issues:
  - #7: Clarify meaning of Security Domain
    - Also part of #70 Juergen’s feedback
  - #62: Editorial update for SD full removal
- Had previous WG consensus to remove formal concept
- Proposal (pull request #72, and part of #75)
  - Remove from API discussion
  - Remove OTrP message name
  - Remove explicit entry from terminology section
  - **Depend on SUIT manifest to express security domain dependency**
  - Include SD informatively in example text (next slide)

Security Domains (2/2)

Separate from the Client App's manifest, this framework relies on the use of the manifest format in [I-D.ietf-suit-manifest] for expressing how to install the TA as well as dependencies on other TEE components and versions. That is, dependencies from TAs on other TEE components can be expressed in a SUIT manifest, including dependencies on any other TAs, or trusted OS code (if any), or trusted firmware. Installation steps can also be expressed in a SUIT manifest.

For example, TEE's compliant with Global Platform may have a notion of a "security domain" (which is a grouping of one or more TAs installed on a device, that can share information within such a group) that must be created and into which one or more TAs can then be installed. It is thus up to the SUIT manifest to express a dependency on having such a security domain existing or being created first, as appropriate.

Keeping secrets from the TAM (1/2)

- Issue:
  - #64 End to end security between a SP and TEE for confidential IP
- Desire to get an encrypted binary that the TAM cannot decrypt
- One proposal was:
  - Encrypted binary can be delivered just like an unencrypted binary
  - Decryption key is delivered separately, but **by a different (SP's) TAM**
- Current text in doc:
  - For any client app, there should be **only a single TAM** for the TEEP Broker to contact.
  - This is also the case when a Client App uses multiple TAs, or when one TA depends on another TA in a software dependency …
  - The reason is that the SP should **provide each TAM** that it places in the Client App's manifest **all the TAs** that the app requires.

Keeping secrets from the TAM (2/2)

- Option 1) Agent uses a single TAM for TA and all dependencies
  - TAM URI of every dependency is assumed to be same as for the depending TA
  - SP must host TAM for all TAs that depend on its secret
- Option 2) Agent can use a separate TAM per dependency
  - Every dependency can have its own TAM URI in the manifest file
  - If a TA depends on another TA, the dependent TA might even have its own SUIT manifest
  - If TAM URI for a dependency is different from the depending TA, can invoke RequestTA recursively

Multiple TAM URIs for a TA

- Issue:
  - #14 Multiple TAMs for a single Client App
- Ming proposes:
  - A TA binary can be carried by multiple TAMs, similar to application stores for client apps.
  - We propose to allow multiple TAM URIs in a manifest file for a TA where it can be downloaded.
  - Only one of the TAMs needs to be contacted and others can be used as failover TAM if the primary fails to respond.

Trust Anchor Storage

- Issue:
  - #51 Should Trust Anchor format be specified in a separate draft?
- Title now a misnomer, now about specifying requirements for protecting trust anchors
- Section 5.8 already covers protecting trust anchors inside the TEE
- SUIT architecture has definitions of ‘trust anchor’ and ‘trust anchor store’ with the requirements from this TEEP issue
- Options:
  1. Copy in definitions from SUIT architecture
  2. Reference definitions in SUIT architecture
  3. …?

“Device Secure Storage” vs TEE

- Issues:
  - #58 Difference between "Device secure storage" and "Device TEE" unclear
  - #30 Cardinality of TEE key pair and certificate
- Current text already permits multiple TEEs per device
- Figure 6 shows keys for different layers/components, including:

Key pair & certificate for:	Location	Cardinality
Trusted firmware	Device **secure storage**	1 per device
TEE	Device TEE	1 per ~~device~~ TEE (#30)

- What is “device secure storage”? (could it be in TEE? ROM? etc.)
- Proposal:
  - Elaborate on requirements for “device secure storage”, similar to requirements on “trust anchor store”

Trust Anchor Update

- Issue:
  - #32 Trust Anchor lifecycle management
    - Also part of #70 Juergen’s feedback
- Current text:
  - It is **out of the scope** in this document to specify how the trust anchors should be updated when a new root certificate should be added or existing one should be updated or removed.
  - **A device manufacturer is expected to provide its TEE trust anchors live update or out-of-band update** to Device Administrators.
- Can’t/shouldn’t you use TEEP to update trust anchors in the TEE?
- Proposal:
  - A manufacturer may have a Trust Anchor Manager TA, with trust anchors in the configuration data for that TA

Summary of Potential SUIT Manifest Requirements

1. Ability to list one or more TAM URIs
   - This is different from a URI to download the binary
2. Install steps/dependencies for Security Domain on GP devices
3. …

Issues relevant to RATS

Attestation (1/2)

- Issue:
  - #17 Capabilities of the Attestation Mechanism
  - #31 Seed for TAM protocol
  - #70 Comments from Jürgen Schönwälder
- (#17) TEEP Agent may not have access to an attestation key for signing OTrP messages (e.g., on SGX)
- (#31) “The attestation hierarchy and seed required for TAM protocol operation must be built into the device at manufacture.” is details about attestation
- (#70) Doc specifies specific mandatory digital signature formats for attestation format, which list can get out of date over time

Attestation (2/2)

- WG agreement to replace OTrP with TEEP protocol
- WG agreement that TEEP protocol should just depend on RATS
- Proposal:
  - Arch doc should:
    - Explain the relationship between TEEP and attestation
    - Reference RATS architecture
    - Leave protocol details to the TEEP protocol spec
  - Arch doc should NOT discuss attestation details:
    - Signing anything with any attestation key?
    - Seeding of attestation keys?
    - Specific crypto algorithms for attestation?

TAM validation of device cert

- Issue:
  - #70 Comments from Jürgen Schönwälder
- Doc says:
  - If the root CA of some TEE device certificates is compromised, these devices might be rejected by a TAM, which is a decision of the TAM implementation and policy choice.
  - Any intermediate CA for TEE device certificates SHOULD be validated by TAM with a Certificate Revocation List (CRL) or Online Certificate Status Protocol (OCSP) method.
- Why SHOULD and not MUST?
- Options:
  1. Remove, leave it to RATS
  2. SHOULD, but elaborate on why
  3. MUST

RATS Claims

- Issue: (none filed yet)
- To use RATS work, the TEEP WG needs a list of what attestation claims that a TAM needs to work with
- Currently no official list exists, but one can be derived from the OTrP spec
- Options:
  1. Put a list in the arch doc
  2. Put a list in the TEEP protocol spec
  3. Put a list in some new TEEP doc

Other TEEP issues

TEEP Conceptual APIs

- Issue:
  - #11: TEEP Broker and APIs
- Concern: common names for events to more easily associate transport spec actions with OTrP/TEEP protocol spec actions
- Proposal (pull request #74):
  - Add TEEP Agent event names:
    - RequestTA, ProcessTeepMessage, RequestPolicyCheck, ProcessError
  - Add TAM event names:
    - ProcessConnect, ProcessTeepMessage
  - Transport spec already uses these event names
  - TEEP protocol spec needs to be updated to use them (separate github issue)
    - OTrP spec previously used ProcessOTrPMessage event name

Rich Execution Environment security

- Issues:
  - #65 IoT DDoS draft issues
  - #66 IoT DDoS issues
- "Is TEEP the right security protocol to prevent DDoS attacks from ioT devices. Types of attacks could be random network traffic generated, reflection of network packets or amplification of legitimate network packets."
- Proposal (no change):
  - TEEP manages the **TEE** in a device, not the **REE** in a device.
  - Propose closing issues as out of scope
  - This may be relevant to RATS though

Editorial-only

- Issues:
  - #56 Terminology: p.6: Device User: a human being
    - Pull request 71 updates text to not imply there needs to be a human being
  - #63 Clarification of the location of keys, certs, and CA certs for prototyping
    - Suggests diagram of locations of keys instead of table
  - #70 Comments from Jürgen Schönwälder (parts not covered earlier)
    - Various typos and grammatical confusion