TEEP Protocol
draft-ietf-teep-protocol-12

Dave Thaler <dthaler@microsoft.com>
297: Error return for QueryResponse

• TEEP Agent can return error via TEEP messages
• TAM has no way to return an error, e.g. processing a Query Response
• Examples: TAM rejects QueryResponse based on AttestionResult, or lack of reachability to a Verifier, etc.
• Sometimes fix is to push a SUIT update to the Agent, but not always:
  • Multiple TAMs, transient errors, etc.
• Added to Update message, for symmetry with QueryResponse:
  • ? err-code => uint .size 1,
  • ? err-msg => text .size (1..128),
Use of EAT
310: Attestation Results impersonation

- **Threat:** A Compromised Agent tries to use Attestation Results from a healthy Agent
- **Mitigation:** require “cnf” claim as defined in S3.1 of RFC 8747 (Proof-of-Possession Key Semantics for CBOR Web Tokens)

7.1.1.1. Handling an Attestation Result

The Attestation Result must first be validated as follows:

1. Verify that the Attestation Result was signed by a Verifier that the TAM trusts.
2. Verify that the Attestation Result contains a "cnf" claim (as defined in Section 3.1 of [RFC8747]) where the key ID is the hash of the TEEP Agent public key used to verify the signature on the TEEP message, and the hash is computed using the Digest Algorithm specified by one of the SUIT profiles supported by the TAM (SHA-256 for the ones mandated in this document).

See Sections 3.4 and 6 of [RFC8747] for more discussion.
Security Considerations

Attestation

A TAM relies on signed Attestation Results provided by a Verifier, either obtained directly using a mechanism outside the TEEP protocol (by using some mechanism to pass Evidence obtained in the attestation payload of a QueryResponse, and getting back the Attestation Results), or indirectly via the TEEP Agent forwarding the Attestation Results in the attestation payload of a QueryResponse. See the security considerations of the specific mechanism in use (e.g., EAT) for more discussion.

An impersonation attack, where one TEEP Agent attempts to use the attestation payload of another TEEP Agent, can be prevented using a proof-of-possession approach. The "cnf" claim is mandatory in the EAT profile for EAT for this purpose. See Section 6 of [RFC8747] and Section 7.1.1.1 of this document for more discussion.
281: EAT profile mandatory claims

- Thomas Fossati: Currently all claims are listed as optional. Shouldn't they be mandatory since the TAM needs them to determine what to do? nonce is mandatory only when the nonce freshness mechanism is used, so maybe that one is optional? Or should you use a separate profile per freshness mechanism?

- NEW:
  - Required Claims: ueid, oemid, hwmodel, hwversion, manifests, cnf.
  - Additional Claims: eat_nonce. See {{freshness-mechanisms}} for discussion.
289: Relationship to AR4SI

- Addressed per discussion at IETF 115 with informative reference and explanation
- TEEP profile: needed by TAM as Relying Party
- AR4SI: might be used for other Relying Parties
- TEEP Agent might need to get both
Relationship to AR4SI cont.

Three cases for TAM<->Verifier protocol (outside of TEEP):

• Case 1: allows getting multiple AR formats from same Evidence
  • Request both profiles for same Evidence

• Case 2: only 1 format per Evidence, but freshness doesn’t use nonce
  • Send same Evidence in two queries, one per format

• Case 3: only 1 format per Evidence, with nonce
  • Don’t use Figure 1, but have TEEP Agent talk directly to Verifier to get AR4SI
Use of EAT + SUIT
285: SUIT in EAT manifests claim

- Added into TEEP EAT profile section:
  - The sw-name claim for a Trusted Component holds the URI of the SUIT manifest for that component.
  - The manifests claim uses a SUIT manifest, where the manifest body contains a SUIT_Reference as defined in Section 4 of [I-D.ietf-suit-report], and the content type is as defined in [I-D.ietf-suit-report].
286: Summary of crypto algorithm cases

1. signing TEEP messages exchanged between the TEEP Agent and the TAM.
2. signing EAT-based Evidence sent by the Attester via the TEEP Agent and the TAM to the Verifier. (If evidence is not encrypted by the TEEP Agent then it will be opaque to the TEEP Agent and to the TAM.)
3. encrypting EAT-based Evidence sent by the TEEP Agent to the TAM. (The TAM will decrypt the encrypted Evidence and will forward it to the Verifier.)
4. signing and optionally encrypting SUIT reports sent by the TEEP Agent to the TAM.
5. signing and optionally encrypting SUIT manifests sent by the Trusted Component Signer to the TEEP Agent.
286: Sensitive info in EATs and SUIT Reports

• We need to be able to encrypt EATs and SUIT Reports
• We have ways to negotiate cipher suite for TEEP messages, which don’t use encryption
• Could encrypt TEEP messages, but EATs may go TEEP Agent<->Verifier
  • Hence previous agreement to use COSE inside TEEP messages, as is done for personalization data (draft-ietf-suit-firmware-encryption)
• What encryption algorithms are mandatory to implement?
  • IETF 115 meeting suggested AEAD with e.g., CCM
  • draft-moran-suit-mti had AEAD with GCM (not CCM)
• How do we negotiate cipher suites?
EAT/SUIT cipher suite negotiation

• Add separate “supported-eat-suit-ciphersuites” to QueryRequest
  • Similar to existing supported-teep-cipher-suite
  • Currently assumes same ciphersuites for EAT vs SUIT encryption
  • Question: is this ok or should we split?
Mandatory algorithms for EAT/SUIT

TEEP uses COSE for confidentiality of EATs and SUIT Reports sent by a TEEP Agent. The TEEP Agent obtains a signed EAT and then SHOULD encrypt it using the TAM as the recipient. A SUIT Report is created by a SUIT processor, which is part of the TEEP Agent itself. The TEEP Agent is therefore in control of signing the SUIT Report and SHOULD encrypt it. Again, the TAM is the recipient of the encrypted content. For content-key distribution Hybrid Public Key Encryption (HPKE) is used in this specification. See COSE-HPKE [I-D.ietf-cose-hpke] for more details. This specification uses the COSE-HPKE variant for a single recipient, i.e., the TAM, which uses COSE_Encrypt0. This variant is described in Section 3.1.1 of [I-D.ietf-cose-hpke].

To perform encryption with HPKE the TEEP Agent needs to be in possession of the public key of the recipient, i.e., the TAM. See Section 5 of [I-D.ietf-teep-architecture] for more discussion of TAM keys used by the TEEP Agent.

This specification defines cipher suites for confidentiality protection of EATs and SUIT Reports. The TAM MUST support each cipher suite defined below, based on definitions in [I-D.moran-suit-mti]. A TEEP Agent MUST support at least one of the cipher suites below but can choose which one. For example, a TEEP Agent might choose a given cipher suite if it has hardware support for it. A TAM or TEEP Agent MAY also support other algorithms in the COSE Algorithms registry. It MAY also support use with COSE_Encrypt or other COSE types in additional cipher suites.

```
$suit-cose-profile /= suit-sha256-es256-hpke-a128gcm
$suit-cose-profile /= suit-sha256-eddsa-hpke-a128gcm
```

Propose: switch hpke to ecdh, to match draft-ietf-suit-mti
Next steps

• WGLC on next version?