TEEP Protocol

draft-ietf-teep-protocol-05
Hannes Tschofenig, Ming Pei, David Wheeler,
Dave Thaler, Akira Tsukamoto
Changes since IETF 109 (1/2)

• Resolutions from IETF 109 TEEP meeting:
  • Combined Install/Delete into Update message
  • Made token be absent when redundant, e.g., with attestation
  • Both ciphersuites are MTI on TAM and optional on Agent
  • Use SUIT Component Identifier as Trusted Component id

• Resolutions based on IETF 109 SUIT meeting:
  • SUIT manifest is now used to delete components
Changes since IETF 109 (2/2)

1. Added reference to relevant EAT claims - see later slide
2. Added sample EAT - see later slide
3. Updated error codes – see later slide
4. Added max sizes to various fields
5. Added behavior specification
6. Various editorial clarifications
### #49: EAT Claims meeting TEEP Arch req’ts

<table>
<thead>
<tr>
<th>Arch draft requirement</th>
<th>Claim</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device unique identifier</td>
<td>device-identifier</td>
<td>draft-birkholz-rats-suit-claims, § 3.1.3</td>
</tr>
<tr>
<td>Device vendor</td>
<td>vendor-identifier</td>
<td>draft-birkholz-rats-suit-claims, § 3.1.1</td>
</tr>
<tr>
<td>Device class</td>
<td>class-identifier</td>
<td>draft-birkholz-rats-suit-claims, § 3.1.2</td>
</tr>
<tr>
<td>TEE hardware type</td>
<td>chip-version-scheme</td>
<td>draft-ietf-rats-eat, § 3.7</td>
</tr>
<tr>
<td>TEE hardware version</td>
<td>chip-version-scheme</td>
<td>draft-ietf-rats-eat, § 3.7</td>
</tr>
<tr>
<td>TEE firmware type</td>
<td>component-identifier</td>
<td>draft-birkholz-rats-suit-claims, § 3.1.4</td>
</tr>
<tr>
<td>TEE firmware version</td>
<td>version</td>
<td>draft-birkholz-rats-suit-claims, § 3.1.8</td>
</tr>
<tr>
<td>Freshness proof</td>
<td>nonce</td>
<td>draft-ietf-rats-eat, § 3.3</td>
</tr>
</tbody>
</table>

- All requirements appear to be met between the above two drafts
- One is not a RATS WG doc (yet?) but TEEP has a normative dependency
#67: Sample EAT (not yet implementable)

```
/ issuer / 1: "joe",
/ timestamp (iat) / 6: 1(1526542894)
/ nonce / 10: h'948f8860d13a463e8e',
/ secure-boot / 15: true,
/ debug-status / 16: 3, / disabled-permanently /
/ security-level / <TBD>: 3, / secure-restricted /
/ device-identifier / <TBD>: h'e99600dd921649798b013e9752dcf0c5',
/ vendor-identifier / <TBD>: h'2b03879b33434a7ca682b8af84c19fd4',
/ class-identifier / <TBD>: h'9714a5796bd245a3a4ab4f977cb8487f',
/ chip-version-scheme / <TBD35>: "MyTEE v1.0",
/ component-identifier / <TBD>: h'60822887d35e43d5b603d18bcaa3f08d',
/ version / <TBD>: "v0.1"
```

IETF 109 - TEEP WG
OPEN #51: Error codes (1/2)

• Error messages have an integer error code and an optional text message
  • Draft -04 had lots of error codes and an IANA registry

• Draft -05 proposes:
  • New error codes should be added sparingly, not for every implementation error.
  • That is the intent of the err-msg field, which can be used to provide details meaningful to humans.
  • New error codes should only be added if the TAM is expected to do something behaviorally different upon receipt of the error message, rather than just logging the event.
  • Hence, each error code is responsible for saying what the behavioral difference is expected to be.

• Since error code implies protocol behavior difference, and protocol behavior is protocol specification, removed IANA registry

• Also reduced number of error codes, and added example behavioral differences
OPEN #51: Error codes (2/2)

A TAM receiving this error might...

• **ERR_PERMANENT_ERROR**: stop trying for some (long) time
• **ERR_UNSUPPORTED_EXTENSION**: retry without using extensions
• **ERR_UNSUPPORTED_MSG_VERSION**: retry using different version
• **ERR_UNSUPPORTED_CRYPTO_ALG**: retry using different crypto alg
• **ERR_BAD_CERTIFICATE**: attempt to use an alternate certificate
• **ERR_CERTIFICATE_EXPIRED**: renew its certificate before using it again
• **ERR_TEMPORARY_ERROR**: retry after some (short) time
• **ERR_MANIFEST_PROCESSING_FAILED**: attempt to install or update other components that do not depend on the failed manifest
OPEN #129: errors processing QueryRequest

• Spec defines errors for things that could be hit by QueryRequest:
  • ERR_UNSUPPORTED_MSG_VERSION, ERR_UNSUPPORTED_CRYPTO_ALG, ...
• But specifies Error message can only be sent in response to Update

• Options:
  1. Allow Error to be sent in response to QueryRequest
  2. Silently drop QueryRequest (but harder to debug)
  3. Duplicate optional error fields into QueryRequest message

• PROPOSED RESOLUTION: option 1
#132: Ciphersuites (Russ’s review)

| Value | Ciphersuite                          |
|-------+-------------------------------------|
| 1     | AES-CCM-16-64-128, HMAC 256/256, X25519, EdDSA |
| 2     | AES-CCM-16-64-128, HMAC 256/256, P-256, ES256 |

• Russ: I think that future ciphersuites should allow MAC algorithms other than HMAC, such as GMAC. Just change "HMAC" to "MAC"

• In Section 9, please say whether future registrations will allow integrity-without-confidentiality ciphersuites. Let's settle this now instead of dumping on the IANA Expert.
OPEN #41: SUIT manifest example(s)

• Add to TEEP Protocol spec or SUIT manifest spec?
  • SUIT manifest spec has 6 other examples in appendix B,
    • but is close to done and SUIT Reports are in a different spec
  • Propose putting in TEEP protocol spec with both manifest and report

• Possible examples:
  • Manifest: Install a TA that needs personalization data from another TAM
  • Report: partial success (e.g., failure of personalization data)
  • Manifest: Remove a TA and its dependency (e.g., personalization data)
  • Report: full success
Use of Tokens in TEEP: Freshness and State Matching
First some recap of RATS relationship...

Verifier

Attester
(TEE)

Compare evidence
against policy
(reference values)

Evidence
in QueryResponse

Attestation
Result

Relying
Party
(TAM)

Compare attestation
result against policy

TEEP

remediation steps
in SUIT Manifest
Freshness & replay protection in RATS

• Verifier cares about:
  • Was Evidence recently signed by Attester, not an old replay?
  • Are values of claims recent, not obsolete values in recent evidence?

• Relying Party (TAM) cares about:
  • Was Attestation Result is recently signed by Verifier, not an old replay?
  • Are values of any claims recent, not obsolete values in recent results?

• How “recent” is up to the appraisal policy

• Details are up to the protocol, but there are three common ways...
Method 1: Timestamps

• Put timestamps in claims in Evidence and Attestation Results
• Requires roughly synchronized clocks
  • Requires a trusted source of time, internal or external
  • Requires secure time sync protocol (e.g., ntpsec inside TEE)
• Also adds claims about the signer’s time sync mechanism
• No additional messages or state at attestation time
Method 2: Nonces

• Receiver supplies nonce that sender must include in signed Evidence or Attestation Results
• No dependency on time sync or clocks at senders
• Receivers have to keep state to remember each nonce supplied until it’s used
• Receivers need a clock to “expire” nonces, but need not be synced
• Only addresses freshness of Evidence / Attestation Results, not freshness of claim values
Method 3: Epoch handles

- Some “handle distributor” periodically sends out epoch handles to sender(s) and receiver(s)
- Senders use latest epoch handle in all messages in place of nonce
- Receivers check if received handle is in most recent set (e.g., of size 2)
- Receiver state is constant, compared to nonces
- Only handle distributor requires a reliable clock
- “Recency” policy limited to a multiple of handle distribution period
OPEN #131: Challenge when attestation bit set

• QueryResponse has opt. “challenge” field used in creating Evidence:
  • “When a challenge is provided in the QueryRequest and an EAT is returned with the QueryResponse message then the challenge contained in this request MUST be copied into the nonce claim found in the EAT.”

• Q1: does TEEP require nonce approach or is any of the approaches ok?
  • Original intent: any is ok, attestation bit can be set with challenge absent

• Q2: But then how do Verifier & Agent agree on freshness approach?
  • Evidence carries approach specific claim (EAT: nonce, timestamp, handle?)

• Option 1: force use of a single approach (narrows TEEP applicability)
• Option 2: force Agent to support all (hard for constrained Agents)
• Option 3: add field in QueryRequest like we did for supported ciphersuites
  • but TAM is just a passthrough, like for challenge
• Option 4: overload challenge field and make it mandatory when attestation bit set
OPEN #127: Use of token vs challenge in QueryRequest

query-request = [ ...
    ? token => bstr .size (8..64),
    ? challenge => bstr .size (8..64),
... 

The token is not needed when the attestation bit is set in the data-item-requested value. The size of the token is at least 8 bytes (64 bits) and maximum of 64 bytes, which is the same as in an EAT Nonce Claim

• Intent was:
  • token is present iff attestation bit is clear (used in response token)
  • challenge is only allowed if attestation bit is set (used in evidence)

• Currently have separate CBOR label values

• QUESTION: Should we combine them into one label?
OPEN #40: unsolicited QueryResponse?

• If Attestation is used with timestamps or handles, then no need for per-request token or nonce

• Do we still need extra round trip to get QueryRequest?

• Proposed optimization:
  • If a QueryRequest is received that contains no token or challenge, Agent MAY cache the QueryRequest and send unsolicited QueryResponses to that TAM URI when handling a RequestTA or UnrequestTA API call
  • Handling a RequestPolicyCheck API call shall always generate a QueryRequest, to enable detecting changes at TAM

• Also avoids bad Agent incentive to have frequent RequestPolicyCheck just to get QueryRequests you can pend for when you need them
OPEN #83: How long should TAM keep token?

• #43 was discussed at IETF 109 with resolution:
  • Agent’s SUIT processor resolves dependencies, and doesn’t respond to initial Install until all dependencies are resolved and SUIT processing completes
  • This also affects state requirement around tokens (and nonces!)

• Some discussion in PR #110
  • Akira & Ming proposed one-day timeout as strawman
    • This seems too long to me, since freshness and state minimization both motivate as small as possible; prefer O(minutes) but admin configurable
  • Ming mentions “dormant” device case must be handled
    • But device gets token when awake since transport (today) is Agent-initiated
  • Ming proposed allowing TAM to be stateless by carrying TAM’s expiration time in field that is echo’ed back to the TAM
    • Token would need to be TAM-signed “cookie” so Agent can’t simply change it
      • Would need changes to description of token field since not just “random”