IETF 112 TEEP Hackathon

November 14, 2021

Akira Tsukamoto (AIST)

IETF 112 SUIT TEEP Hackathon

- Date November 2 Tuesday – November 5 Friday
  Jointly with SUIT
- Participants:

Akira Tsukamoto, AIST

Kuniyasu Suzaki, TRASO/AIST

Kohei Isobe, SECOM

Ken Takayama, SECOM

Masashi Kikuchi, TRASIO

Takahiko Nagata, TRASIO

Brendan Moran, ARM

Hannes Tschofenig, ARM

Dave Thaler, Microsoft

Background and objective

Background

- TEEP requires SUIT, RATS and COSE
- Started implementing to support SUIT manifests in TEEP Update messages in TEEP protocol around IETF 111

- Would like to complete how to handle SUIT manifests on TEEP at IETF112
- After IETF112, start implement to support COSE and EAT in RATS on TEEP

Objective

- Finalizing formats of SUIT manifests in the draft after trying the implementation
- Start looking in to COSE on TEEP messages, SUIT manifests already handling COSE, so use it as much as possible in the implementation

What got done

Implementations

- TAM: tamproto: Distribute the TC binary integrated in the SUIT manifest in TEEP Update
- libcsuit: Supporting three examples of suit manifests
- TEEP-device: Update TEEP message formats from draft-05 to 06
  Adding handling COSE on SUIT manifests
  Adding handling ‘integrated-payload’ (not complete)

Feedback on drafts

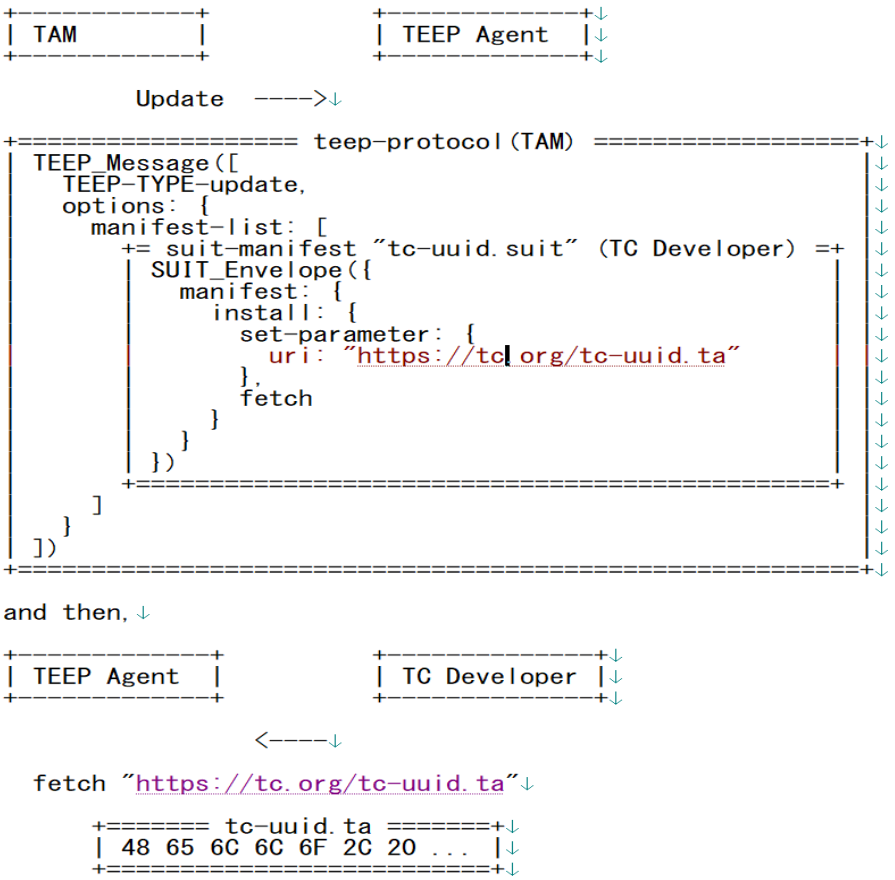
- TEEP: When to include token or not in Update message
  - https://github.com/ietf-teep/teep-protocol/issues/166
- TEEP: Proposing cipher-suites to use cddl description in SUIT for TEEP
  - https://github.com/ietf-teep/teep-protocol/issues/167
- TEEP: Revising the suit manifest for deleting TC by using ‘unlink’
  - https://github.com/ietf-teep/teep-protocol/issues/168
  - https://github.com/ietf-teep/teep-protocol/pull/169
- TEEP: Adding three examples of SUIT manifest formats <- description in following pages
  - https://github.com/ietf-teep/teep-protocol/pull/161
- SUIT: One line patch for defining ES256 used in SUIT examples
  - https://github.com/suit-wg/manifest-spec/pull/50

Three SUIT manifest formats, Example 1 (1/2)

Having one SUIT Manifest pointing to a URI of a Trusted Component Binary

- Pros:
  - The Trusted Component Developer can ensure that the intact Trusted Component Binary is downloaded by TEEP Devices
  - The TAM does not have to deliver Update message containing Trusted Component Binary which may have a large size
- Cons:
  - The Trusted Component Developer must host the Trusted Component Binary server
  - The TEEP Device must fetch the Trusted Component Binary in another connection after receiving an Update message

Three SUIT manifest formats, Example 1 (2/2)



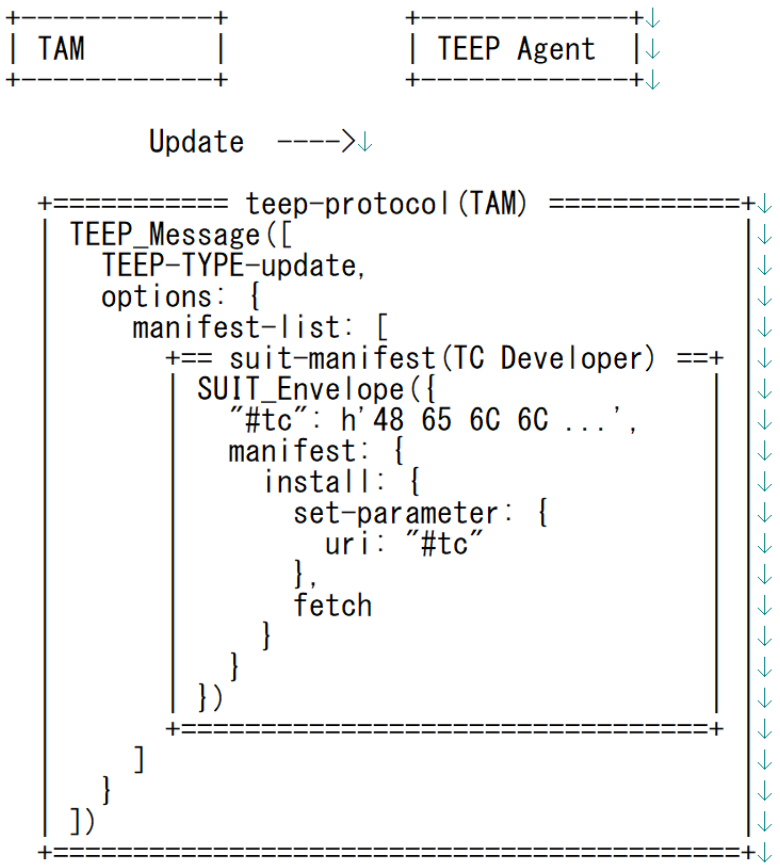
Three SUIT manifest formats, Example 2 (1/2)

Having a SUIT Manifest include the Trusted Component Binary

- Pros:
  - The TEEP Device can obtain the Trusted Component Binary and its SUIT manifest together in one Update message
  - The Trusted Component Developer does not have to host a server to deliver the Trusted Component Binary directly to TEEP Devices

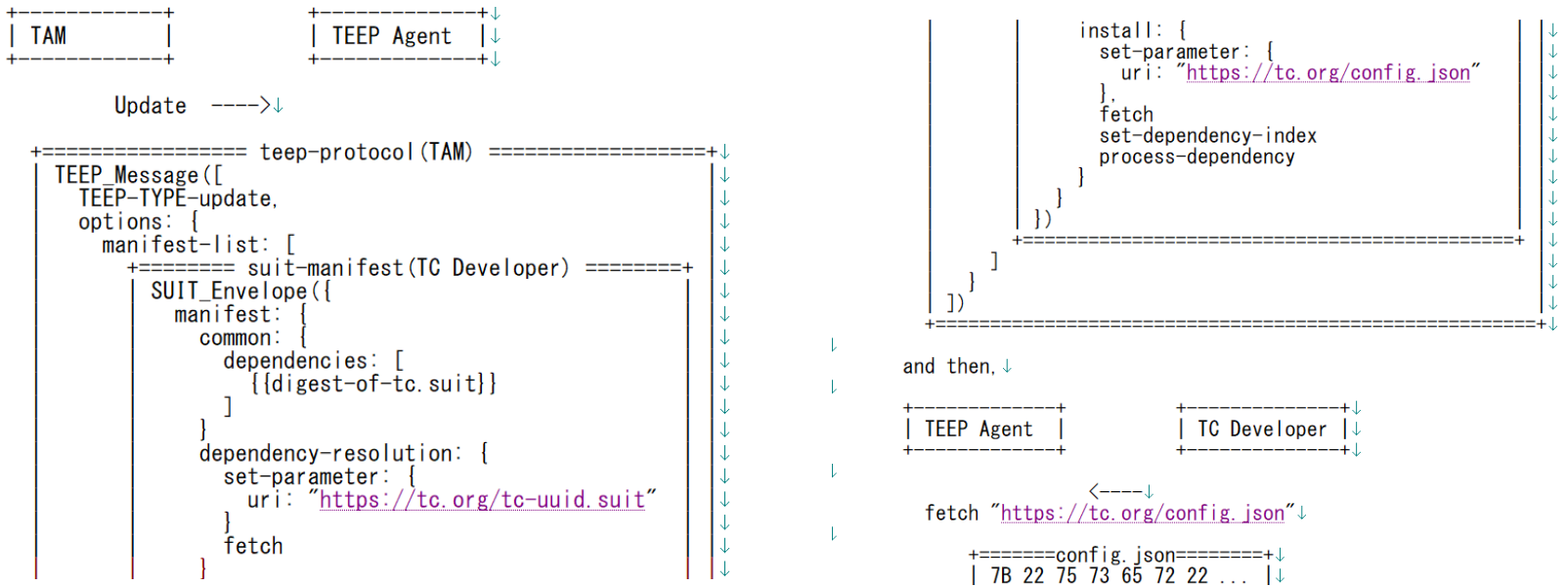
- Cons:
  - The TAM must host the Trusted Component Binary itself, rather than delegating such storage to the Trusted Component Developer
  - The TAM must deliver Trusted Component Binaries with integrated SUIT manifest in Update messages, which may result increasing the Update message size

Three SUIT manifest formats, Example 2 (2/2)



Three SUIT manifest formats, Example 3

Supplying Personalization Data for the installed Trusted Component Binary



Format examples expanding in the draft

- Indispensable to keep compatibilities of implementation among different vendors
- Require breakdowns of binary representations when debugging between different TAM and TEEP devices
- It is good CBOR tutorial

TEEP message format examples

```
D.1.1.  D.1.1.  CBOR Diagnostic Notation
/ query-request = /
[
  1,  / type : TEEP-TYPE-query-request = 1 (uint (0..23)) /
  / options : /
  {
    20 : 0xa0a1a2a3a4a5a6a7a8a9aaabacadaeaf,
           / token = 20 (mapkey) :
             h'a0a1a2a3a4a5a6a7a8a9aaabacadaeaf' (bstr .size (8..64)),
             generated by TAM /
    1 : [ 1 ], / supported-cipher-suites = 1 (mapkey) :
                 TEEP-AES-CCM-16-64-128-HMAC256--256-X25519-EdDSA =
                 [ 1 ] (array of .within uint .size 4) /
    3 : [ 0 ] / version = 3 (mapkey) :
                 [ 0 ] (array of .within uint .size 4) /
  },
  3   / data-item-requested :
        attestation | trusted-components = 3 (.within uint .size 8) /
]
```

```
D.1.2.  D.1.2.  CBOR Binary Representation

   83                    # array(3)
     01                  # unsigned(1) uint (0..23)
     A4                  # map(4)
       14                # unsigned(20) uint (0..23)
       4F                # bytes(16) (8..64)
         A0A1A2A3A4A5A6A7A8A9AAABACADAEAF
       01                # unsigned(1) uint (0..23)
       81                # array(1)
         01              # unsigned(1) within uint .size 4
       03                # unsigned(3) uint (0..23)
       81                # array(1)
         00              # unsigned(0) within uint .size 4
       04                # unsigned(4) uint (0..23)
       43                # bytes(3)
         010203          # "¥x01¥x02¥x03"
     03                  # unsigned(3) .within uint .size 8
```

SUIT message format examples

```
E.2.  Example 2: SUIT Manifest including the Trusted Component Binary

   ### CBOR Diagnostic Notation of SUIT Manifest

/ SUIT_Envelope_Tagged / 107 ( {
  / suit-authentication-wrapper / 2: << [
    << [
      / suit-digest-algorithm-id: / -16 / cose-alg-sha256 /,
      / suit-digest-bytes: / h'C8363BDF3DCF68F0234A9DD320C2FEA72DE68F46AAE7CE700AFF
    ] >>,
    << / COSE_Sign1_Tagged / 18 ( [
      / protected: / << {
        / algorithm-id / 1: -7 / ES256 /
      } >>,
      / unprotected: / {},
      / payload: / null,
      / signature: / h'E0D2973A7B7185BBDA108458FB68EFAF65CDC
    ] ) >>
  ] >>,
  / suit-integrated-payload / "#tc": h'48656C6C6F2C205365637
  / suit-manifest / 3: << {
    / suit-manifest-version / 1: 1,
    / suit-manifest-sequence-number / 2: 3,
    / suit-common / 3: << {
      / suit-components / 2: [
        [
          h'544545502D446576696365',          / "TEEP-Devic
          h'5365637572654653',                / "SecureFS"
          h'8D82573A926D4754935332DC29997F74', / tc-uuid /
          h'7461'                             / "ta" /
```

```
E.2.1.  CBOR Binary Representation

D8 6B                                    # tag(107) / SUIT_Envelope_Tagged /
   A3                                    # map(3)
      02                                 # unsigned(2) / suit-authentication-wrapper /
      58 73                              # bytes(115)
         82                              # array(2)
            58 24                        # bytes(36)
               82                        # array(2)
                  2F                     # negative(15) / -16 = cose-alg-sha256 /
                  58 20                  # bytes(32)
                     C8363BDF3DCF68F0234A9DD320C2FEA72DE68F46AAE7CE700AFF87085516A335
            58 4A                        # bytes(74)
               D2                        # tag(18) / COSE_Sign1_Tagged /
                  84                     # array(4)
                     43                  # bytes(3)
                        A1               # map(1)
                           01            # unsigned(1) / algorithm-id /
                           26            # negative(6) / -7 = ES256 /
                     A0                  # map(0)
                     F6                  # primitive(22) / null /
                     58 40               # bytes(64)
                        E0D2973A7B7185BBDA108458FB68EFAF65CD031F2283E784129A95D4229F0EB11F8947D3E
```

Summary

- Moving from purifying TEEP message itself, to defining and handling SUIT and COSE in TEEP message formats
- Implementation progressed for supporting SUIT manifests
- Draft updated related on SUIT and COSE from the hackathon
- Explaining three formats of SUIT manifests
- How do the examples looks of CBOR in TEEP
- After 112, many remining on COSE and EAT on TEEP protocol

A part of this hackathon presentation is based on results obtained from a project, JPNP16007,commissioned by the New Energy and Industrial Technology Development Organization (NEDO).