SUIT WG
libcsuit Progress

Ken Takayama

Summary: Against WG Documents

- SUIT Manifest (submitted to IESG)
  - 43/45 (96%)
- SUIT Multiple Trust Domains (in WG Last Call)
  - 7/8 (75%) <= ☑Process-Dependency
- SUIT Update Management
  - 10/14 (71%)
- SUIT Encrypted Payloads
  - 1/4 (25%) <= ☑AES-KW, ❌ECDH, ❌HPKE, ❌CEK Verification

What is libcsuit?

What is libcsuit?

- OSS library for SUIT Manifest written in language C
  - available here: https://github.com/kentakayama/libcsuit
  - depends on QCBOR, t_cose, and crypto libraries such as MbedTLS
  - runs even in Intel SGX enclave

- Supported functions

Decode: Read SUIT Manifest and stores it into C struct
  - Verify the suit-authentication-wrapper and suit-digest
  - Print in CBOR Diagnostic notation (can be used with other CBOR tools)

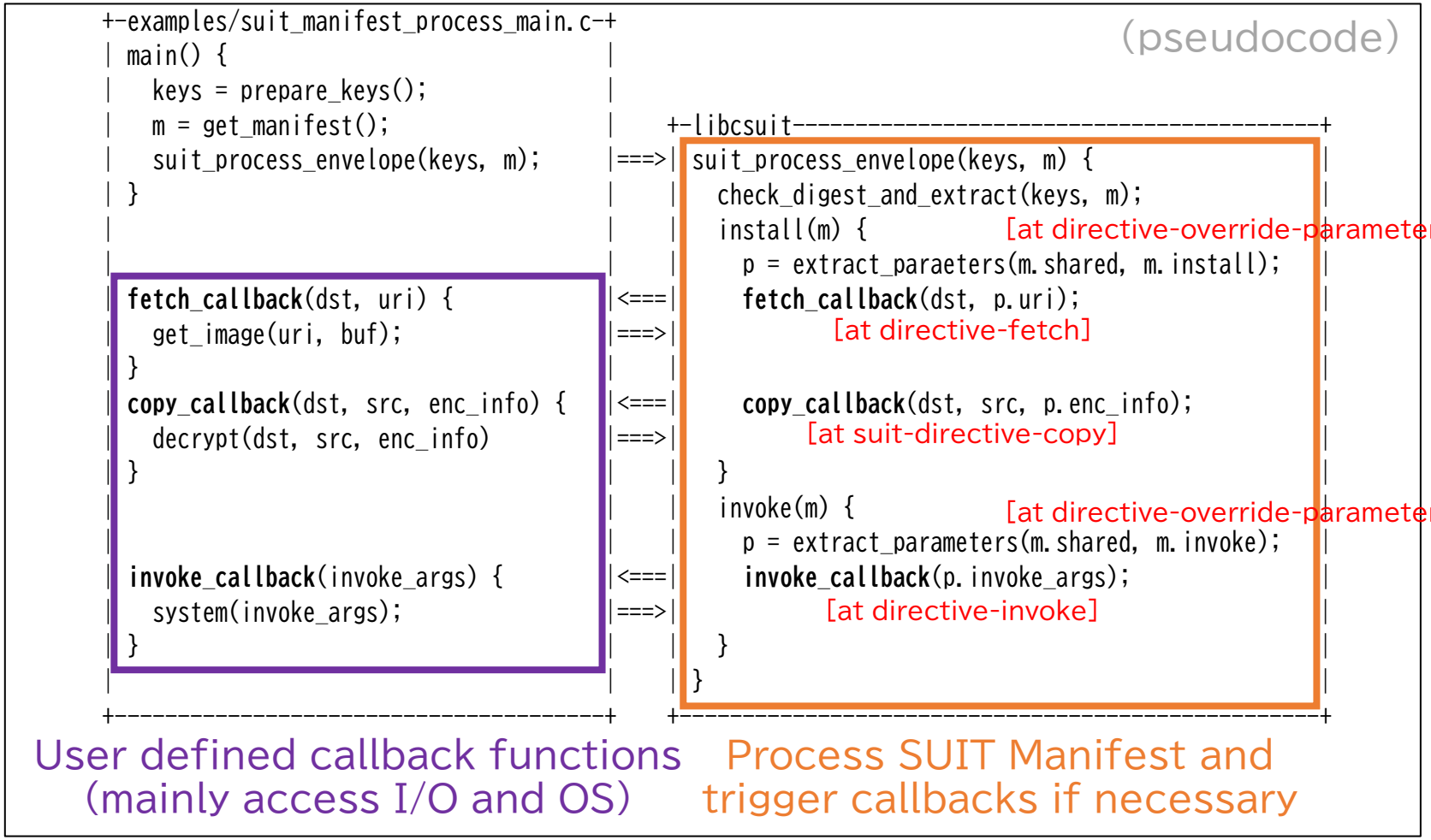
Encode: Write SUIT Manifest from C struct
  - Sign on a SUIT Manifest without suit-authentication-wrapper

Process: Execute SUIT Manifest within less memory

Explain it in detail

Design of libcsuit Process

- Execute the **most part of SUIT Manifest**
- Trigger user defined callbacks on **platform dependent part**
  - such as **fetch, copy, invoke**, etc.
  - You can replace it on your choice in linking phase



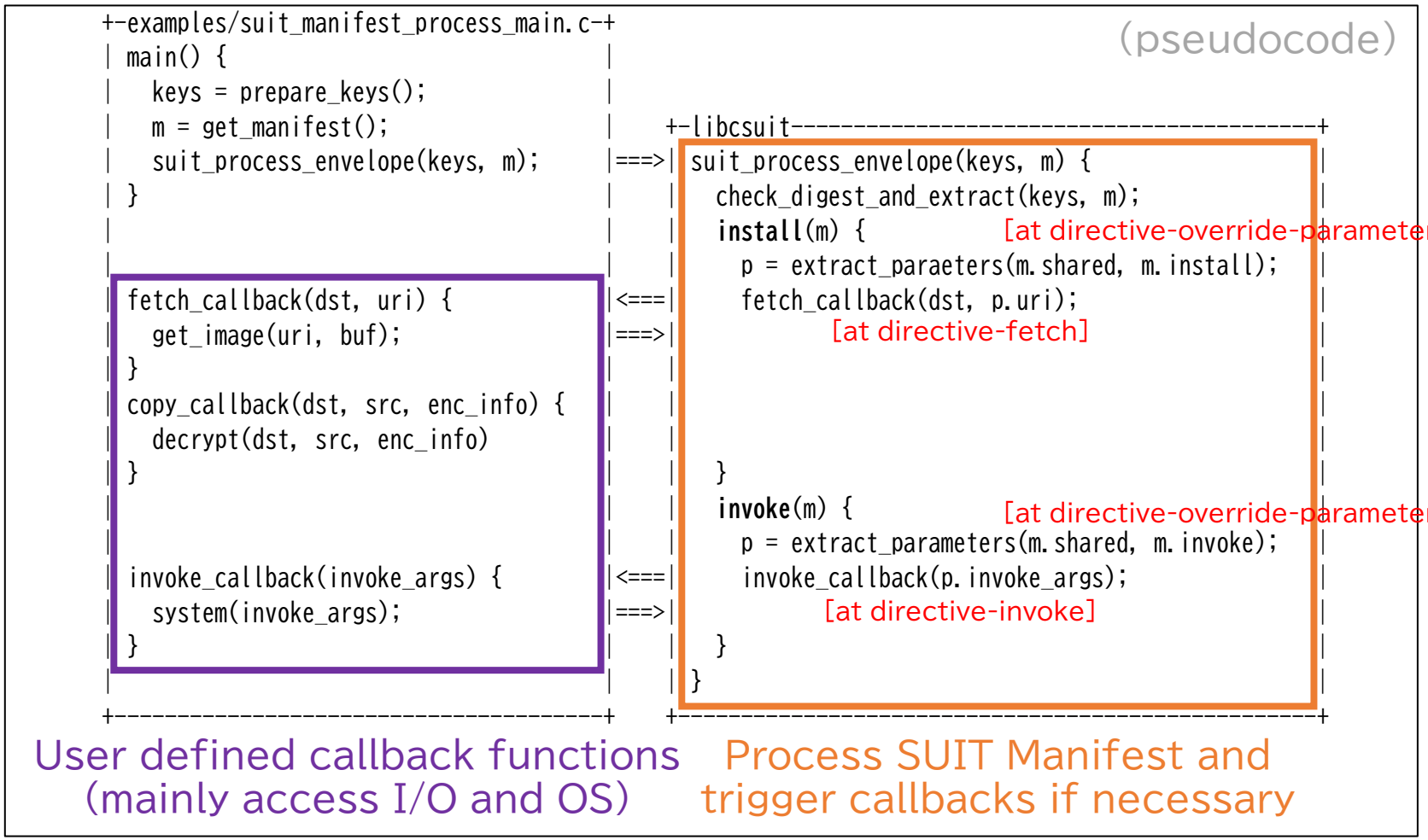
Example SUIT Manifest

```
/ SUIT_Envelope_Tagged / 107({
  / authentication-wrapper / 2: << [
  ...
  ] >>,
  / manifest / 3: << {
    ...,
    / common / 3: << {
      / components / 2: [
        ['a.out']
      ],
    } >>,
    / invoke / 9: << [
      / directive-override-parameters / 20, {
        / parameter-invoke-args / 23: './a.out'
      },
      / directive-invoke / 23, 15
    ] >>,
    / install / 17: << [
      / directive-override-parameters / 20, {
        / parameter-image-digest / 3: << [
          / digest-algorithm-id: / -16 / SHA256 /,
          / digest-bytes: / h'0011223344...'
        ] >>,
        / parameter-image-size / 14: 34768,
        / parameter-uri / 21: "http://example.com/file.bin"
      },
      / directive-fetch / 21, 15,
      / condition-image-match / 3, 15
    ] >>
  } >>
})
```

NOTE: Though SUIT Manifest **MUST** be encoded with the canonical CBOR, libcsuit **DOES NOT** execute from head to foot.

How libcsuit Process It?

- libcsuit process
  - payload-fetch => install => validate => load => invoke,
  - can be skipped by caller's choice



Wanna try?

Setup & Run

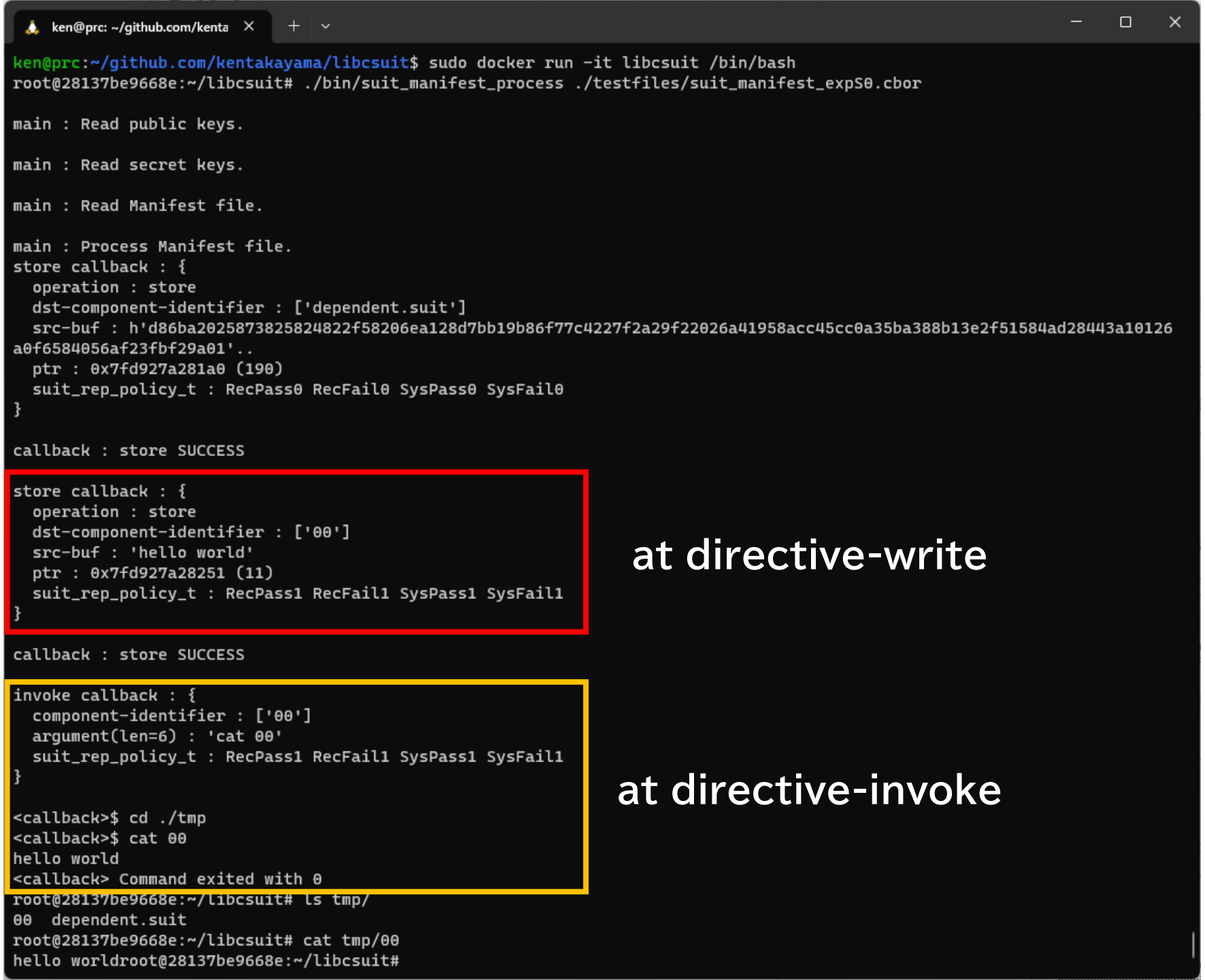
- Install docker
  - see official document
    - e.g. https://docs.docker.com/engine/install/ubuntu/
- Build and Run libcsuit inside container
  - $ cd /path/to/working/dir/
  - $ git clone https://github.com/kentakayama/libcsuit
  - $ cd ./libcsuit
  - $ sudo docker build -t libcsuit -f psa.Dockerfile .
  - $ sudo docker run -it libcsuit /bin/bash
  - # ./bin/suit_manifest_process ./testfiles/suit_manifest_expS0.cbor

- Linux and x86 CPU are RECOMMENDED

S0: Let's Start with Simple Example

```
/ SUIT_Envelope_Tagged / 107({
  / authentication-wrapper / 2: << [
  ...
  ] >>,
  / manifest / 3: << {
    ...,
    / common / 3: << {
      / components / 2: [
        ['00']
      ],
    } >>,
    ...
    / invoke / 9: << [
      / directive-override-parameters / 20, {
        / parameter-invoke-args / 23: 'cat 00'
      },
      / directive-invoke / 23, 15
    ] >>,
    / install / 17: << [
      / directive-override-parameters / 20, {
        / parameter-content / 18: 'hello world'
      },
      / directive-write / 18, 15
    ] >>
  } >>
})
```

You can see full CBOR Diagnostic here
https://github.com/kentakayama/libcsuit/blob/master/testfiles/suit_manifest_expS0.md



at directive-write

at directive-invoke

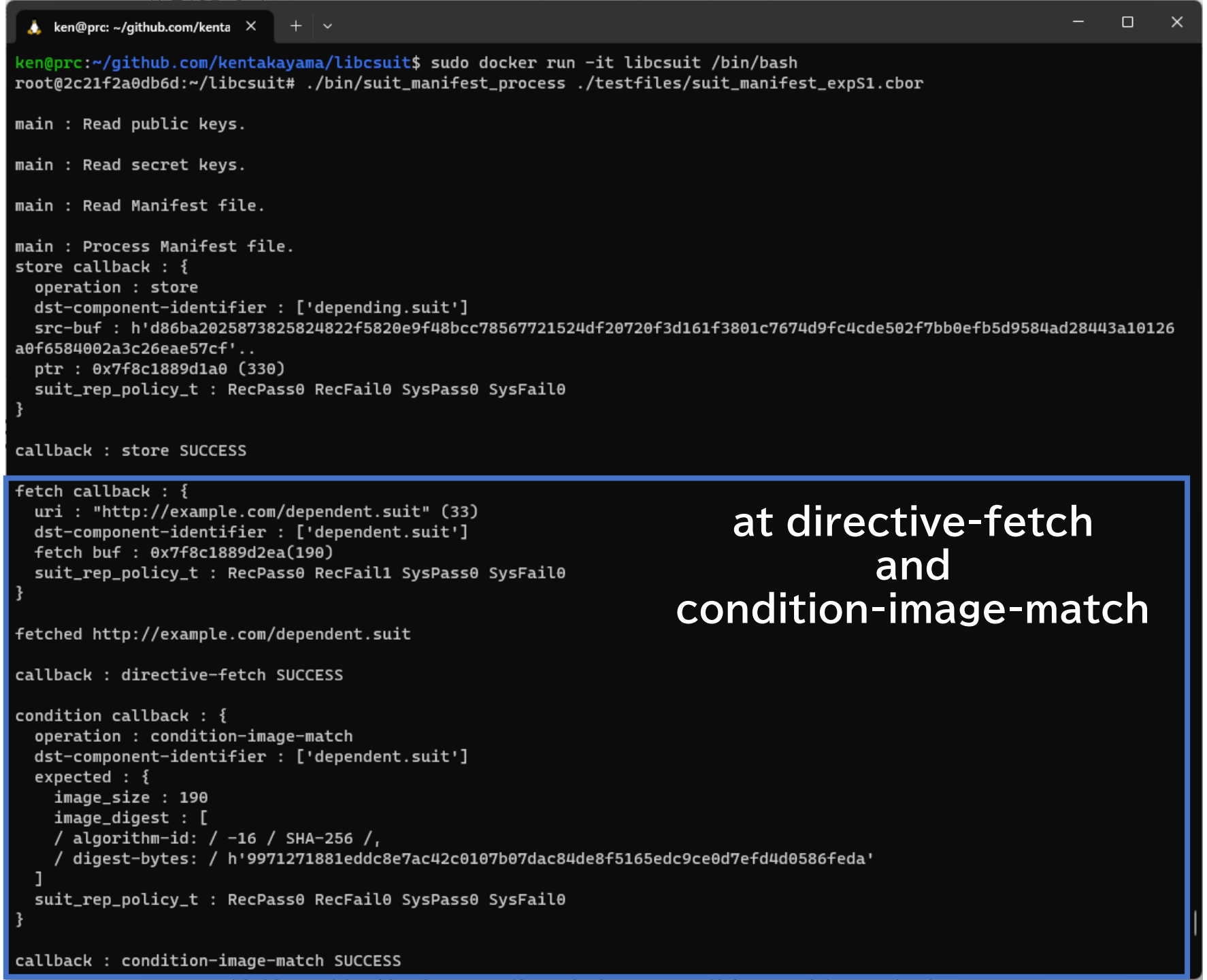
S1: depends S0 Example

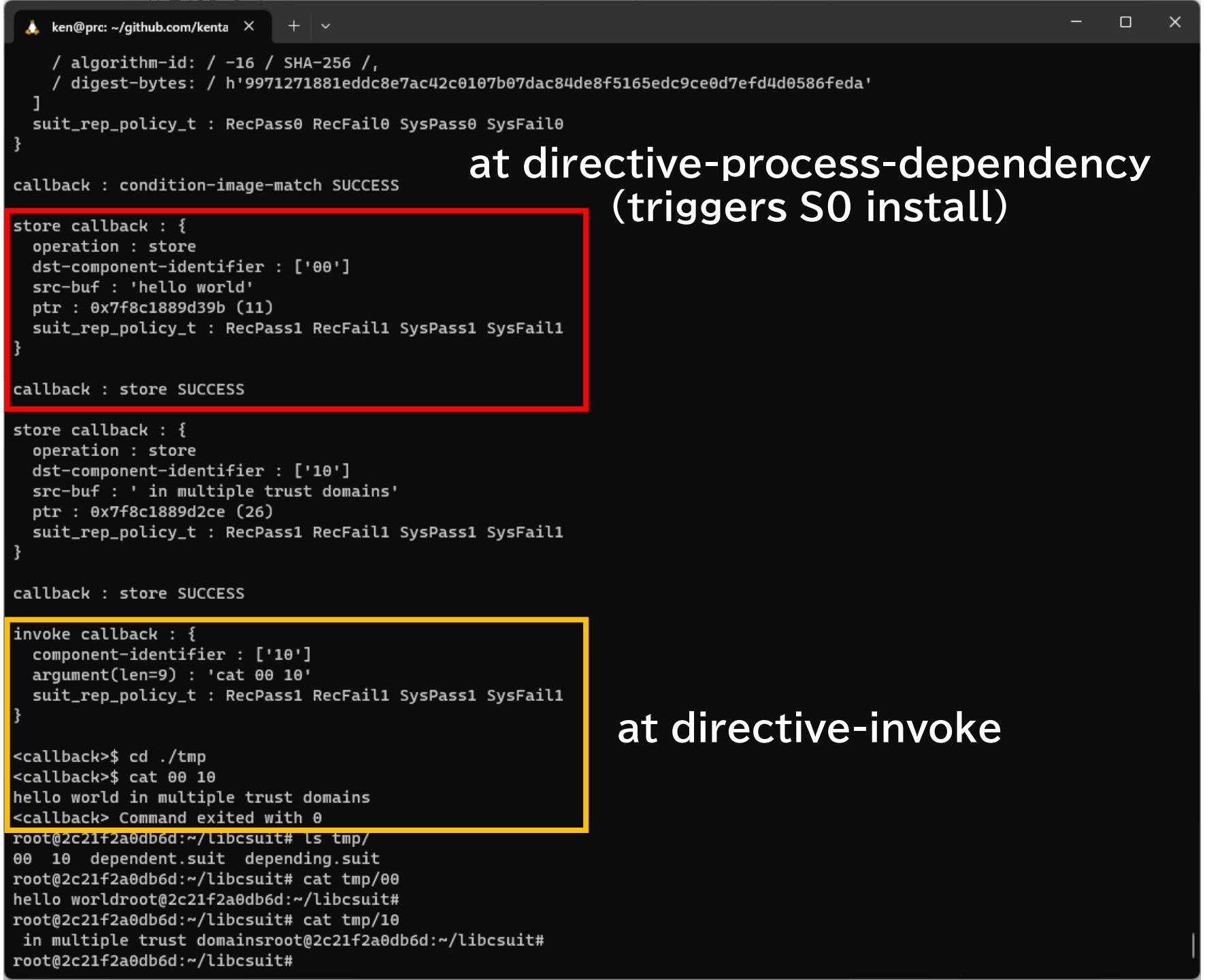
```
/ SUIT_Envelope_Tagged / 107({
  / authentication-wrapper / 2: << [
  ...
  ] >>,
  / manifest / 3: << {
    ...,
    / common / 3: << {
      / dependencies / 1: {
        / component-index / 1: {
          / dependency-prefix / 1: [
            'dependent.suit'
          ]
        }
      },
      / components / 2: [
        ['10']
      ],
    } >>,
    ...
    / invoke / 9: << [
      / directive-override-parameters / 20, {
        / parameter-invoke-args / 23: 'cat 00 10'
      },
      / directive-invoke / 23, 15
    ] >>,
```

```
    / dependency-resolution / 15: << [
      / directive-set-component-index / 12, 1,
      / directive-override-parameters / 20, {
        / parameter-image-digest / 3: << [
          / digest-algorithm-id: / -16 / SHA256 /,
          / digest-bytes: / h'997127...'
        ] >>,
        / parameter-image-size / 14: 190,
        / parameter-uri / 21: "http://example.com/dependent.suit"
      },
      / directive-fetch / 21, 2,
      / condition-image-match / 3, 15
    ] >>,
    / install / 17: << [
      / directive-set-component-index / 12, 1,
      / directive-process-dependency / 11, 0,

      / directive-set-component-index / 12, 0,
      / directive-override-parameters / 20, {
        / parameter-content / 18: ' in multiple trust domains'
      },
      / directive-write / 18, 15
    ] >>
```

You can see full CBOR Diagnostic here
https://github.com/kentakayama/libcsuit/blob/master/testfiles/suit_manifest_expS1.md


ken@prc: ~/github.com/kenta
ken@prc:~/github.com/kentakayama/libcsuit$ sudo docker run -it libcsuit /bin/bash
root@2c21f2a0db6d:~/libcsuit# ./bin/suit_manifest_process ./testfiles/suit_manifest_expS1.cbor
main : Read public keys.
main : Read secret keys.
main : Read Manifest file.
main : Process Manifest file.
store callback : {
operation : store
dst-component-identifier : ['depending.suit']
src-buf : h'd86ba20258738258248 22f5820e9f48bcc78567721524df20720f3d161f3801c7674d9fc4cde502f7bb0efb5d9584ad28443a10126
a0f6584002a3c26eae57cf'..
ptr : 0x7f8c1889d1a0 (330)
suit_rep_policy_t : RecPass0 RecFail0 SysPass0 SysFail0
}
callback : store SUCCESS
fetch callback : {
uri : "http://example.com/dependent.suit" (33)
dst-component-identifier : ['dependent.suit']
fetch buf : 0x7f8c1889d2ea(190)
suit_rep_policy_t : RecPass0 RecFail1 SysPass0 SysFail0
}
fetched http://example.com/dependent.suit
callback : directive-fetch SUCCESS
condition callback : {
operation : condition-image-match
dst-component-identifier : ['dependent.suit']
expected : {
image_size : 190
image_digest : [
/ algorithm-id: / -16 / SHA-256 /,
/ digest-bytes: / h'9971271881eddc8e7ac42c0107b07dac84de8f5165edc9ce0d7efd4d0586feda'
]
suit_rep_policy_t : RecPass0 RecFail0 SysPass0 SysFail0
}
callback : condition-image-match SUCCESS
at directive-fetch
and
condition-image-match



at directive-process-dependency
(triggers S0 install)

at directive-invoke

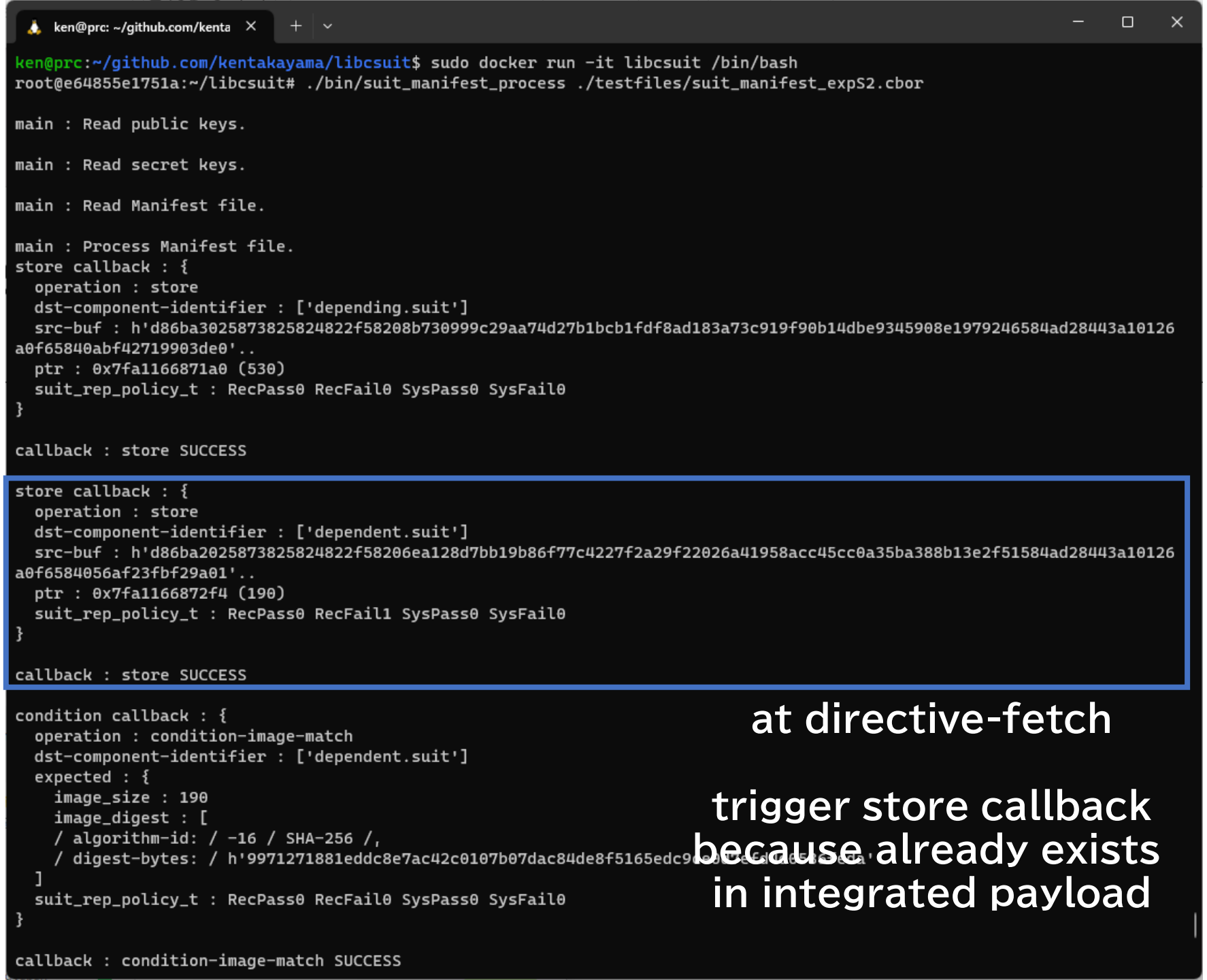
S2: depends S0 in Integrated Dependency

You can see full CBOR Diagnostic here
https://github.com/kentakayama/libcsuit/blob/master/testfiles/suit_manifest_expS2.md

```
...,
/ common / 3: << {
  / dependencies / 1: {
    / component-index / 1: {
      / dependency-prefix / 1: [
        'dependent.suit'
      ]
    }
  },
  / components / 2: [
    ['10']
  ],
} >>,
...
/ invoke / 9: << [
  / directive-override-parameters / 20, {
    / parameter-invoke-args / 23: 'cat 00 10'
  },
  / directive-invoke / 23, 15
] >>,
```

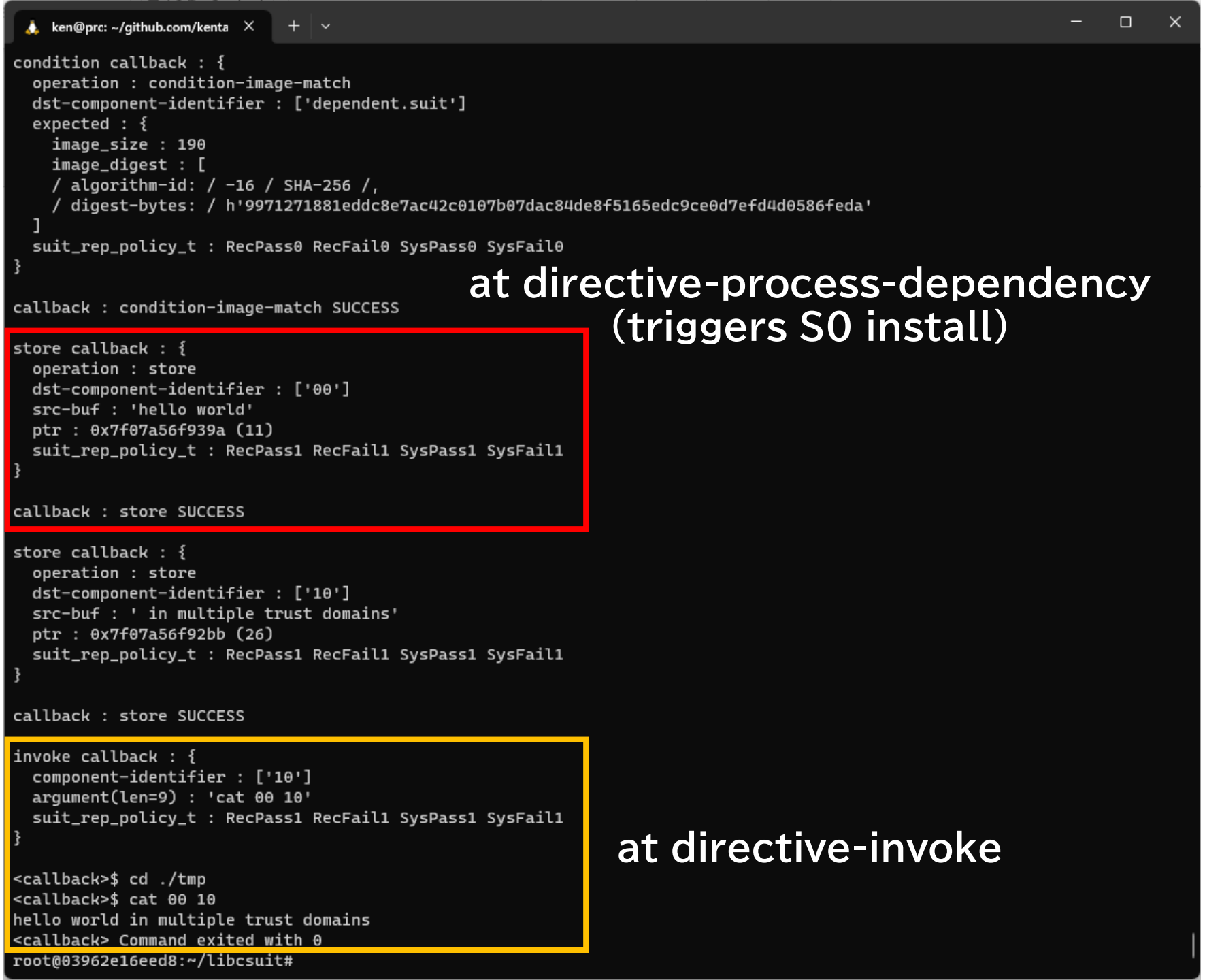
```
dependency-resolution / 15: << [
  / directive-set-component-index / 12, 1,
  / directive-override-parameters / 20, {
    / parameter-image-digest / 3: << [
      / digest-algorithm-id: / -16 / SHA256 /,
      / digest-bytes: / h'997127...'
    ] >>,
    / parameter-image-size / 14: 190,
    / parameter-uri / 21: "#dependent.suit"
  },
  / directive-fetch / 21, 2,
  / condition-image-match / 3, 15
] >>,
/ install / 17: << [
  / directive-set-component-index / 12, 1,
  / directive-process-dependency / 11, 0,

  / directive-set-component-index / 12, 0,
  / directive-override-parameters / 20, {
    / parameter-content / 18: ' in multiple trust domains'
  },
  / directive-write / 18, 15
] >>
} >>,
"#dependent.suit": h'd86ba2…'
})
```



at directive-fetch

trigger store callback
because already exists
in integrated payload



at directive-process-dependency
(triggers S0 install)

at directive-invoke

You can test libcsuit with a lot of Examples

- 6 examples from SUIT Manifest document
  - suit_manifest_exp[0-5]
- 2 examples for SUIT Payload Encryption document
  - suit_manifest_expEW and suit_manifest_expEF
- 4 examples for SUIT Multiple Trust Domain document
  - suit_manifest_expS[0-3]
- 4 examples for TEEP Protocol document
  - suit_manifest_exp(U|I|D|R)

libcsuit Next Plan

1. Support ALL

- SUIT Manifest (submitted to IESG)
  - 45/45 (100%)
- SUIT Multiple Trust Domains (in WG Last Call)
  - 8/8 (100%)
- SUIT Update Management
  - 14/14 (100%)
- SUIT Encrypted Payloads
  - 4/4 (100%) <= ☑AES-KW, ☑ECDH, ☑HPKE, ☑CEK Verification
- SUIT Report

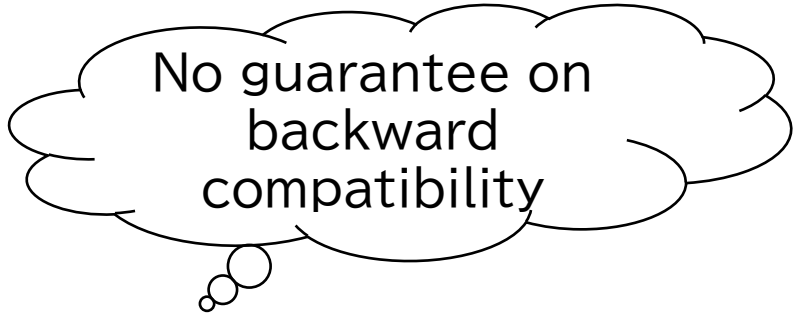
(Platform dependent part are out of work)

2. Provide SUIT Manifest Examples

- SUIT Multiple Trust Domains (as soon as possible)
  - Simple Dependency (S1 is enough?)
  - Integrated Dependency (S2 is enough?)
- SUIT Update Management
  - Covers All Condition Check
- SUIT Reports
  - Produce Capability Report
    - e.g. disabling dependency feature of libcsuit and create a SUIT Report when parsing a SUIT Manifest using the feature
- TEEP Protocol (TEEP WG)
  - Encrypted Personalization Data

Any Feedbacks are Welcome!

LICENSE



Special Thanks

- Yuichi Takita
  - original implementor of libcsuit
- Brendan Moran
  - for precise documents
- Hannes Tschofenig
  - for direct feedback on libcsuit
- Laurence Lundblade
  - for his useful and powerful libraries QCBOR and t_cose

Appendix
libcsuit with Encrypted Payload

libcsuit supports AES-KW

- Current Example in draft-suit-firmware-encryption
  - uses 2 components: [h'00'] and [h'01']
  - [h'01'] is encrypted payload
  - parameter-encryption-info has COSE_Encrypt encryption-info
  - 👍 the SUIT Manifest parser can save working memory
  - 👎 consumes 2 components, SUIT Manifest is complicated a bit

- New Example use parameter-content + directive-write
  - uses only 1 component: [h'00']
  - parameter-component has encrypted payload
  - parameter-encryption-info has COSE_Encrypt encryption-info
  - 👍 consumes only 1 component, directives are quite simple
  - 👎 encrypted payload cannot be detached from the SUIT Manifest

Parameter-URI + Directive-Fetch

```
/ common / 3: << {
  / components / 2: [
    [h'00'] / to be decrypted firmware /,
    [h'01'] / encrypted firmware /
  ]
} >>,
/ install / 17: << [
  / fetch encrypted firmware /
  / directive-set-component-index / 12, 1 / [h'01'] /,
  / directive-override-parameters / 20, {
    / parameter-uri / 21: "https://author.example.com/encrypted-firmware.bin",
    / parameter-image-size / 14: 47
  },
  / directive-fetch / 21, 15,
```

Encrypted Payload

AES128KW Recipient in Encryption Info

```
  / decrypt encrypted firmware /
  / directive-set-component-index / 12, 0 / [h'00'] /,
  / directive-override-parameters / 20, {
    / parameter-source-component / 22: 1 / [h'01'] /,
    / parameter-encryption-info / 19: << 96([
      / protected: / << {
        / alg / 1: 1 / AES-GCM-128 /
      } >>,
      / unprotected: / {
        / IV / 5: h'1de460e8b5b68d7222c0d6f20484d8ab'
      },
      / payload: / null / detached ciphertext /,
      / recipients: / [
        [
          / protected: / << {
          } >>,
          / unprotected: / {
            / alg / 1: -3 / A128KW /,
            / kid / 4: 'kid-1'
          },
          / payload: / h'a86200e4754733e4c00fc08c6a72cc1996e129922eab504f'
          / CEK encrypted with KEK /
        ]
      ]
    ]) >>
  },
  / directive-copy / 22, 15 / consumes the SUIT_Encryption_Info above /,
```

Parameter-URI + Directive-Fetch

```
$ ./bin/suit_manifest_process ./testfiles/suit_manifest_expEF.cbor
main : Process Manifest file.
fetch callback : {
  uri : "https://author.example.com/encrypted-firmware.bin" (49)
  dst-component-identifier : [h'01']
  fetch buf : 0x7f73994512e9(47)
  suit_rep_policy_t : RecPass1 RecFail1 SysPass1 SysFail1
}

fetched https://author.example.com/encrypted-firmware.bin

callback : directive-fetch SUCCESS

store callback : {
  operation : copy
  dst-component-identifier : [h'00']
  src-component-identifier : [h'01']
  src-buf :
  encryption-info : h'd8608443a10101a105501de460e8b5b68d7222c0d6f20484d8abf6818341a0a2012204456b69642d315818a86200e4754733e4c00fc08c6a72cc1996e129922eab504f'
  ptr : (nil) (0)
  suit_rep_policy_t : RecPass1 RecFail1 SysPass1 SysFail1
}

callback : copy SUCCESS
...
callback : condition-image-match SUCCESS
```

Encrypted Payload

AES128KW Recipient in Encryption Info

Plaintext Payload is extracted ("This is a real firmware image.")

Parameter-Content + Directive-Write

```
/ install / 17: << [
  / fetch encrypted firmware /
  / directive-override-parameters / 20, {
    / parameter-content / 18: h'2b3765ff457dd98a4ba7130a40462b663c08198146d23f3a32094392ca5040c3121c8e5f4c04d5a3d1d6171bcf362b',
    / parameter-encryption-info / 19: << 96([
      / protected: / << {
        / alg / 1: 1 / AES-GCM-128 /
      } >>,
      / unprotected: / {
        / IV / 5: h'1de460e8b5b68d7222c0d6f20484d8ab'
      },
      / payload: / null / detached ciphertext /,
      / recipients: / [
        [
          / protected: / << {
          } >>,
          / unprotected: / {
            / alg / 1: -3 / A128KW /,
            / kid / 4: 'kid-1'
          },
          / payload: / h'a86200e4754733e4c00fc08c6a72cc1996e129922eab504f' / CEK encrypted with KEK /
        ]
      ]
    ]) >>
  }
  / decrypt encrypted firmware /
  / directive-write / 18, 15 / consumes the SUIT_Encryption_Info above /,
```

Encrypted Payload

AES128KW Recipient in Encryption Info

Parameter-Content + Directive-Write

```
$ ./bin/suit_manifest_process ./testfiles/suit_manifest_expEW.cbor
main : Process Manifest file.
store callback : {
  operation : store
  dst-component-identifier : [h'00']
  src-buf : h'2b3765ff457dd98a4ba7130a40462b663c08198146d23f3a32094392ca5040c3121c8e5f4c04d5a3d1d6171bcf362b'
  encryption-info : h'd8608443a10101a105501de460e8b5b68d7222c0d6f20484d8abf6818341a0a2012204456b69642d315818a86200e4754733e4c00fc08c6a72cc1996e129922eab504f'
  ptr : 0x7fd47712d232 (47)
  suit_rep_policy_t : RecPass1 RecFail1 SysPass1 SysFail1
}

callback : store SUCCESS

condition callback : {
  operation : condition-image-match
  dst-component-identifier : [h'00']
  expected :
    image_size : 31
    image_digest : [
    / algorithm-id: / -16 / SHA-256 /,
    / digest-bytes: / h'efe16b6a486ff25e9fb5fabf515e2bfc3f38b405de377477b23275b53049b46b'
  ]
  suit_rep_policy_t : RecPass0 RecFail0 SysPass0 SysFail0
}

callback : condition-image-match SUCCESS
```

Encrypted Payload

AES128KW Recipient in Encryption Info

Plaintext Payload (”This is a real firmware image.”)