draft-ietf-suit-manifest-02

Brendan Moran

Hannes Tschofenig

Henk Birkholz

A little bit of history

- draft-moran-suit-manifest-05 approved for adoption at IETF#105
- draft-moran-suit-manifest-04 incorrectly submitted as draft-ietf-suit-manifest-00 (sorry).
- Draft-ietf-suit-manifest-01 matched draft-moran-suit-manifest-05
- Draft-ietf-suit-manifest-02 submitted for this IETF meeting, as announced on the list: https://mailarchive.ietf.org/arch/msg/suit/GIDLicNmqWE9kIllpieIfWfJQ5c

Summary of changes: -01 to -02

- Added interpreter behavior
  - Abstract machine
- Added templates for constructing manifests
- Added text field enums

Interpreter Behavior (1/2)

- Interpreter setup phase
  - What to do before invoking the interpreter
- Required checks
  - Checks that the interpreter must perform
  - Commands that must appear in the manifest
- Lays out how an interpreter should behave
  - Two different interpreters should interpret the same manifest the same way
  - Provides the basis to enable consistent behavior
- Defines the interpreter abstract machine
  - First steps towards a rigorous definition of interpreter behavior
    - Will enable fine-grained test vectors for interpreters
    - Will enable interpreter validation suites

Interpreter Behaviour (2/2)

- Serialized Processing Interpreter
  - Processes the manifest in its entirety once for each listed component
- Parallel Processing Interpreter
  - Processes some commands out-of-order or in parallel
  - How the interpreter handles data dependencies
- How dependencies are handled

Creating Manifests

- Including source material in the text section
- Templates
  - Required template: compatibility check
- Use-case templates
  - Execute-In-Place (XIP) secure boot
  - Firmware download
  - Load from external storage (decompression optional)
  - Dependency handling

Text Fields

- Defined list of text fields
  1. manifest-description
  2. update-description
  3. vendor-name
  4. model-name
  5. vendor-domain
  6. model-info
  7. component-description
  8. json-source
  9. yaml-source
  10. version-dependencies

Next Steps: -03

- Draft is pretty good state already.
- Github repo contains further (mostly editorial) PRs: https://github.com/suit-wg/manifest-spec/
- Could we come up with a better name than “SUIT manifest”?
- Technical proposals for -03:
  - Run-Sequence vs. Try-Each
  - Examples: Should they move?
  - Map-Test-Execute
  - For-Each
- Discussed in subsequent slides.

Run-Sequence vs. Try-Each

- Run-Sequence
  - Ambiguous state of soft-failure
  - No else-clause on soft-failure
- Try-Each
  - Explicit soft-failure
  - Else-Clause provided

- Should we deprecate run-sequence?

Examples

- Example section is very big
- Options:
  - Prune some info (JSON representation)
  - Move to appendix
  - Move to another document with more extensive use-case information

Map-Test-Execute (1/3)

- Common patterns:
  - For each component (for each component, do <commands>)
  - Set component parameters (digest, size)
  - Choose parameter set based on system properties
  - Prioritized Parameter List
- Possible catch-all approach:

```
[
   map      # List of component ID : {parameter set} pairs
   test     # Command sequence that can soft fail
            # (goes to next map pair)
   execute  # Command sequence that does not soft fail
            # (error causes termination)
]
```

Map-Test-Execute (2/3)

- Possible encoding

```
Map_Test_Execute = [
    mte-parameter-list: MTE_Parameter_list,
    mte-test-sequence: bstr .cbor SUIT_Command_Sequence
    ? mte-exec-sequence: bstr .cbor SUIT_Command_Sequence
]

MTE_Parameter_list = [ + (
    mte-component: uint,
    mte-parameters: {+ SUIT_Parameters}
)]
```

Map-Test-Execute (3/3)

- Costs:
  - Need a set of temporary parameters to prevent side-effects
- Benefits:
  - Smaller encoding of repeated patterns with different parameters

- Should this be:
  - An extension?
  - Optional?
  - Required in certain circumstances?

For-Each-Component

- Easier to understand
- Less flexible than Map-Test-Execute
- Implements only:
  - For each component in <list> do <command-sequence>
- Expands “component-id = True”

Roadmap

- Stable document by the hackathon in Feb.2020.
- Code and tools developed as input for that event and refined during the event.
- Working group last call in March 2020 (for IETF#107)