An Information Model for Manifests

draft-ietf-suit-information-model-01

Status

- Draft –v1 was published in time for the deadline; v0 was published ahead of the Berlin Hackathon/virtual interim meeting.
- Lots of changes in both versions
- This talk focuses only on a subset.

THREATS

Unqualified Firmware

Multiple Network Operators with a Single Device Operator

- Assumptions:
  - Device Operators expect the rights to create firmware
  - Network Operators expect the rights to qualify firmware as fit-for-purpose on their networks
  - Device Operators manage devices that can be deployed on any network
- Attack Scenario:
  - An attacker may obtain a manifest for a device on Network A.
  - Then, this attacker sends that manifest to a device on Network B.
  - Because Network A and Network B are under control of different Operators, and the firmware for a device on Network A has not been qualified to be deployed on Network B, the target device on Network B is now in violation of the Operator B's policy and may get disabled by this unqualified, but signed firmware.

Unqualified Firmware

Single Network Operator with Multiple Device Operators

- Multiple devices that interoperate are used on the same network and communicate with each other.
- Some devices are manufactured and managed by Device Operator A and other devices by Device Operator B.
- A new firmware is released by Device Operator A that breaks compatibility with devices from Device Operator B.
- An attacker sends the new firmware to the devices managed by Device Operator A without approval of the Network Operator.
- This breaks the behaviour of the larger system causing denial of service and possibly other threats.

USE CASE/USABILITY REQUIREMENT

Terminology

- **Heterogeneous Storage Architecture (HeSA)**:
  A device that stores at least one firmware component differently from the rest.
  - Example: A device with internal and SPI-connected external flash memory.
- **Homogeneous Storage Architecture (HoSA):**

  A device that stores all firmware components in the same way.
  - Example: Storage in on-chip flash memory.

Multiple Payloads/Firmware Images

- **Example 1: Multiple Microcontrollers**

  An IoT device with multiple microcontrollers in the same physical device (HeSA) will likely require multiple payloads with different component identifiers.

Multiple Payloads/Firmware Images

- **Example 2: Code and Configuration**

  A firmware image can be divided into two payloads: code and configuration. These payloads may require authorizations from different actors in order to install. This structure means that multiple manifests may be required, with a dependency structure between them.

Multiple Payloads/Firmware Images

- **Example 3: Multiple Chunks**

  A firmware image can be divided into multiple functional blocks for separate testing and distribution. This means that code would need to be distributed in multiple payloads. For example, this might be desirable in order to ensure that common code between devices is identical in order to reduce distribution bandwidth.

Rollback

- Use case introduced by David Brown.
- Imagine a battery powered device connected to a bandwidth constrained network.
- Device keeps old firmware in case of rollback.
- Rollback could require distribution of old firmware + manifest (with increased sequence number).
- Draft also allows just to distribute a new manifest pointing to the old firmware (still with increased sequence number).
- Covered implicitly by the spec but not described in the document.

MANIFEST ELEMENTS

Processing Steps

- A list of all payload processors necessary to process a nested format and any parameters needed by those payload processors.
- Each Processing Step SHOULD indicate the expected digest of the payload after the processing is complete.
- Processing steps are distinct from Directives in that Directives apply to the manifest as a whole, whereas Processing Steps apply to an individual payload and provide instructions on how to unpack it.

Storage Locations

- This element tells the device which component is being updated. The device can use this to establish which permissions are necessary and the physical location to use.

- **Two Storage Locations:**

  A device supports two components: an OS and an application. These components can be updated independently, expressing dependencies to ensure compatibility between the components.

- **File System**

  A device supports a full filesystem. The firmware authority chooses to make the storage identifier the path at which to install the payload.

- **Flash Memory**

  A device supports flash memory. The firmware authority chooses to make the storage identifier the offset where the image should be written.

Component Identifier

- In a heterogeneous storage architecture, a storage identifier is insufficient to identify where and how to store a payload.
- To resolve this, a component identifier indicates which part of the storage architecture is targeted by the payload.

Conditions

- Some conditions need to be checked before installation
  - Identifiers, time, precursors
- Some conditions need to be checked after installation.
- Conditions now split into pre-conditions and post-conditions. Change made to manifest format but not to information model.

Directives, Aliases, Dependencies, etc.

- Most recent changes in manifest serialization draft:
  - Dependencies, Aliases, and payload info are now resources
  - Payloadinfo is divided into resources, processors, and targets
- Text regarding directives, aliases and dependencies out of sync with manifest serialization document.
- More detailed explanation about status in email: https://www.ietf.org/mail-archive/web/suit/current/msg00570.html

Next Steps

- Separating information model from serialization format sounds useful but is difficult to accomplish in practice.
  - Changes to manifest serialization format lead to changes in information model and vice versa.
  - Reviewers like to focus on solution rather than abstract information model.
  - Lots of changes recently.
- Recommended approach in the future:
  - Start with one document and then when discussions are settled split the functionality.