Hackathon-122 Bangkok

2025/03/15-16 Sta-Sun

Nobuo AOKI

Motivation

- IETF activities to develop specifications for supply chain security are becoming more active
  - OPSAWG: *RFC 9472*
  - SCITTWG: *draft-ietf-scitt-architecture-11, draft-ietf-scitt-scrapi-04*
- There are still issues that the high-level discussions conducted by SCITT cannot cover
  - Is the architecture and its API all you need?
  - Does the SCITT architecture provide backward compatibility with RFC 9472?
  - Can we claim transparency of computing resources on a per-host basis with only existing statements and the SCITT architecture?
  - Is the versatility of the SCITT architecture and API sufficient?

Challenges to someone’s needs

- Draft’s Treasure Hunting
  - that weaves in and out existing specifications and drafts
- Backward compatibility with RFC 9472
  - is stands for YANG modeling meets the SCITT Architecture
- Feedback from Case Studies
  - allows us to re-examine the relationship between statements and architecture
  - will show you an aspiration of extending statements without changing the architecture
- Support for Dynamic Statement Lifecycles
  - makes SCITT more versatile
  - enables the verification of the transparency of statements and the tracking of changes to statements

i) Draft's Treasure Hunting

- Basic Principles
  - We must not deceive each other
  - It is **also necessary to determine the value of creating specifications**
- Draftable issues in white space
  - A consistent method of representing deliverables in the software supply chain
  - How to deal with strengthening software and hardware
  - The expressive power of software in the software supply chain has not kept pace with the growth of the field, e.g., transparency of service, transparency of hardware

  … But no limited… Save the fun for our next hackathon in Madrid!!)
- After Hunting
  - Create an **extended specification** with care for backward compatibility with the original specification
  - The SCITT specification is highly versatile so that we can **reflect our findings in the SCITT specification or reference implementation**

ii）Backward compatibility w/ RFC 9472 -Summary-

- **Potential problems**
  - Do we have a method for evaluating the transparency of a single computing resource (e.g., Host) at once?
- **Key issues**
  - Not suitable for use cases that consistently check multiple Statements (e.g., X-BOMs) at various layers in one place
    - A layer is a hierarchical relationship between components that differs from a dependency relationship, such as the bootloader, firmware, software, etc.
  - The virtue is to minimize the changes to the SCITT architecture
- **How to solve**
  - For a set of artifacts, define a set space that can express any serializable information
  - The wisdom of the MUD files in RFC 9472 will help us

iii）Feedback from Case Studies -Summary-

- TODO

iv) Support for dynamic statement lifecycles -Summary-

- **Problem statement**
  - With the current SCITT specs, the transparency of statements on the architecture can be verified
  - So, can SCITT audit changes in statements in the supply chain?
- **Key issues**
  - Ability to audit changes in statements as they move through the supply chain
  - Even if changes are made to SCITT, the ability to verify the correspondence between statements and transparent statements needs to be maintained
- **How to solve**
  - Use new parameters that show the context when the statement is issued
  - Or, define the relationship between statements with hysteresis signatures, etc.
  - Support statements about statements that indicate a change in the chain

Security Considerations

- TODO