draft-harrison-sidrops-manifest-numbers-01

●●●

Tom Harrison (tomh@apnic.net)
George Michaelson (ggm@apnic.net)
Job Snijders (job@fastly.com)
IETF 119 SIDROPS Working Group

What is this about? (1)

- Manifests include a field called manifestNumber
- RFC 6486: Manifests for the RPKI (February 2012)
  - “This field is an integer that is incremented each time a new manifest is issued for a given publication point”
  - “Manifest verifiers MUST be able to handle number values up to 20 octets”
- RFC 9286: Manifests for the RPKI (June 2022)
  - “Each RP MUST verify that a purported "new" manifest contains a higher manifestNumber than previously validated manifests”
  - “If the purported "new" manifest contains a manifestNumber value equal to or lower than manifestNumber values of previously validated manifests, the RP SHOULD use locally cached versions of objects”

What is this about? (2)

- A strict reading of the text requires that relying parties reject new manifests once the largest signed 20-byte value (i.e. `1 << ((20 * 8) – 1) – 1`, called `MN_MAX` from here) is reached
  - The CA can't be used from that point onwards
- But `MN_MAX` is a very large number?
  - Yes: if manifestNumber is incremented by one on reissuance, `MN_MAX` would not be reached in billions of years, even if issuing billions of manifests per second
  - But the manifest number can be set to an arbitrary value by the issuer
    - E.g. ARIN's TA's manifestNumber is currently ~$1x10^{45}$ (~1/128 of `MN_MAX`)
  - And bugs or similar could cause manifestNumber to be set to a large value, or to increment by large values
  - In particular, if a TA becomes unusable, requires new root key issuance
    - Time-consuming, plus long tail of users still using old TAL

How should RPs handle this?

- Draft includes strawman proposal: if the manifest filename changes, reset the manifest number check
  - This is the current behaviour in rpki-client
    - See appendix A for a detailed description of the rpki-client implementation
    - Based on reading 9286 as referring to manifest numbers on a per-manifest-filename basis, rather than a per-CA basis
  - Pros:
    - Simple to implement
    - If there is consensus that this is required by 9286 as-is, then no further draft work required
  - Cons:
    - 9286 can be interpreted as though this is not required, or even permitted
      - See also RFC 8488 (RIPE NCC's Implementation of Resource Public Key Infrastructure (RPKI) Certificate Tree Validation)

What are the other options?

- Remove the manifestNumber check
- Make the largest manifestNumber a function of the current time
- Use serial number arithmetic to facilitate rollover
- Leave as-is on the RP side
  - Up to CAs/TAs to ensure manifestNumber makes sense on the server side before publication

RP support for 9286

	validador FORT			
Manifest number reuse	Accepted	Accepted	Accepted	Rejected
Manifest number regression	Accepted	Accepted	Accepted	Rejected
Manifest number of `MN_MAX`	Rejected	Accepted	Accepted	Accepted
Manifest number > `MN_MAX`	Rejected	Accepted	Rejected	Accepted
Manifest number of `MN_MAX_2`*	Rejected	Accepted	Rejected	Accepted
Manifest number > `MN_MAX_2`*	Rejected	Accepted	Rejected	Rejected

* `(1 << (20 * 8)) - 1` (i.e. the largest unsigned 160-bit value, rather than the largest signed 160-bit value)

https://github.com/APNIC-net/rpki-mft-number-demo

Next steps

- Get feedback on current draft approach and potential alternatives