SIDR
Internet-Draft
Intended status: Informational
Expires: April 21, 2016

T. Bruijnzeels
O. Muravskiy
RIPE NCC

October 19, 2015

RPKI Repository Validation Using Local Cache
draft-tbruijnzeels-sidr-validation-local-cache-02

Abstract

This document describes the approach to validate the content of the RPKI repository, which is independent of a particular object retrieval mechanism. This allows it to be used with repositories available over rsync protocol (see Section 3 of[RFC6481]), and delta protocol ([I-D.tbruijnzeels-sidr-delta-protocol]), as well as repositories that use a mix of both.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 21, 2016.

Copyright Notice

Table of Contents

1. Introduction

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

2. Top-down Validation of a Single Repository

The validation of one repository is independent from any other repository, and thus, multiple repositories could be validated concurrently.

The validation of a repository starts from it's Trust Anchor (TA) certificate. To retrieve the TA, the Trust Anchor Locator (TAL) object is used, as described in Section 2.1.

If the TA certificate is retrieved, it is validated according to the Section 2.2 of [RFC6490].

Then the TA certificate is validated as a resource certificate, as described in Section 2.2.

For all repository objects that were validated during this validation run, their validation timestamp is updated in the local store (see Section 4.1.8).

Outdated objects are removed from the store as described in Section 2.3. This completes the validation of a repository.

2.1. Fetching Trust Anchor Certificate Using Trust Anchor Locator

The following steps are performed in order to fetch the Trust Anchor Certificate:

- If the Trust Anchor Locator contains "prefetch.uris" field, pass the URIs contained there to the fetcher (see Section 3.1.1).
- Pass to the fetcher (Section 3.1.2) the URI from the TAL (see Section 2.1 of [RFC6490]).
- Retrieve from the local store (see Section 4.1.4) all certificate objects, for which the URI matches the URI extracted from the TAL in the previous step, and the public key matches the subjectPublicKeyInfo field of the TAL (Section 2.1 of [RFC6490]).
- If no, or more than one such objects are found, issue an error and stop validation process. Otherwise, use that object as a Trust Anchor certificate.

2.2. Resource Certificate Validation

The following steps describe the validation of a single resource certificate:

- If both the caRepository (Section 4.8.8.1 of [RFC6487]), and the id-ad-rpkiNotify (Section 3.5 of [I-D.tbruijnzeels-sidr-delta-protocol]) SIA pointers are present in the given resource certificate, use a local policy to determine which pointer to use. Extract the URI from the selected pointer and pass it to the fetcher (see Section 3.1.1).

- For a given resource certificate, find it's manifest and certificate revocation list (CRL), using the procedure described in Section 2.2.1. If no such manifest and CRL could be found, issue an error and stop processing current certificate.

- Compare given resource certificate's manifest URI with the URI of the manifest found in the previous step. If they are different, issue a warning.

- Get from the local store and validate repository objects that correspond to the manifest entries, using the procedure described in the Section 2.2.2.

- Validate all resource certificate objects found on the manifest, using the CRL object found on the manifest, according to Section 7 of [RFC6487].

- Validate all ROA objects found on the manifest, using the CRL object found on the manifest, according to the Section 4 of [RFC6482].

- Validate all Ghostbusters Record objects found on the manifest, using the CRL object found on the manifest, according to the Section 7 of [RFC6493].

- For every valid resource certificate object found on the manifest, apply the procedure described in this section (Section 2.2), recursively, provided that this resource certificate (identified by it's SKI) has not yet been validated during current repository

validation run.

2.2.1. Finding most recent valid manifest and CRL

Fetch from the store (see Section 4.1.5) all objects of type manifest, whose certificate's AKI field matches the SKI of the current CA certificate.

Find the manifest object with the highest manifest number, for which all following conditions are met:

- There is only one entry in the manifest for which the store contains exactly one object of type CRL, whose hash matches the hash of the entry.
- The manifest's certificate AKI equals the above CRL's AKI
- The above CRL is a valid object according to Section 6.3 of [RFC5280]

- The manifest is a valid object according to Section 4.4 of [RFC6486], using the CRL found above

Report an error for every invalid manifest with the number higher than the number of the valid manifest.

2.2.2. Manifest entries validation

For every entry in the manifest object:

- Construct an entry's URI by appending the entry name to the current CA's publication point URI.
- Get all objects from the store whose hash attribute equals entry's hash (see Section 4.1.3).
- If no such objects found, issue an error.
- For every found object, compare it's URI with the URI of the manifest entry. If they do not match, issue a warning.
- If no objects with matching URI found, issue a warning.

- If some objects with non-matching URI found, issue a warning.

2.3. Store Cleanup

At the end of repository validation, the store cleanup is performed. Given all objects that were validated during current validation run, it removes from the store (Section 4.1.7) all objects whose URI attribute matches URI of validated object(s), but the hash attribute is different.

3. Remote Objects Fetcher

The fetcher is responsible for downloading objects from remote repositories. Currently rsync and RRDP repositories are supported.

3.1. Fetcher Operations

3.1.1. Fetch repository objects

This operation receives one parameter - a URI. For rsync protocol this URI points to a directory in a remote rsync repository. For RRDP repository it points to the repository's notification file.

The fetcher performs following steps:

- If the given URI has been downloaded recently (as specified by the local policy), do nothing.
- Download remote objects using the URI provided (for rsync repository use recursive mode).
- For every new object that is downloaded, try to parse it as an object of specific RPKI type (certificate, manifest, CRL, ROA, Ghostbusters record), based on the object's filename extension (.cer, .mft, .crl, .roa, and .gbr, respectively), and perform basic RPKI object validation, as specified in [RFC6487] and [RFC6488].
- For every downloaded valid object, record it in the local store (Section 4.1.1), and set it's last fetch time to the time it was

downloaded (Section 4.1.2).

3.1.2. Fetch single repository object

This operation receives one parameter - a URI that points to an object in a remote repository.

The fetcher performs following operations:

- If the given URI has been downloaded recently (as specified by the local policy), do nothing.
- Download the remote object using the URI provided.
- Try to parse downloaded object as an object of a specific RPKI type (certificate, manifest, CRL, ROA, Ghostbusters record), based on the object's filename extension (.cer, .mft, .crl, .roa, and .gbr, respectively), and perform basic RPKI object validation, as specified in [RFC6487] and [RFC6488].
- If the downloaded object is not valid, issue an error and skip further steps.
- Delete objects from the local store (Section 4.1.6) using given URI.
- Put validated object in the local store (Section 4.1.1), and set it's last fetch time to the time it was downloaded (Section 4.1.2).

4. Local Object Store

4.1. Store Operations

4.1.1. Store Repository Object

Put given object in the store, along with it's type, URI, hash, and AKI, if there is no record with the same hash and URI fields.

4.1.2. Update object's last fetch time

For all objects in the store whose URI matches the given URI, set the last fetch time attribute to the given timestamp.

4.1.3. Get objects by hash

Retrieve all objects from the store whose hash attribute matches the given hash.

4.1.4. Get certificate objects by URI

Retrieve from the store all objects of type certificate, whose URI attribute matches the given URI.

4.1.5. Get manifest objects by AKI

Retrieve from the store all objects of type manifest, whose AKI attribute matches the given AKI.

4.1.6. Delete objects for URI

For a given URI, delete all objects in the store with matching URI attribute.

4.1.7. Delete outdated objects

For a given URI and a list of hashes, delete all objects in the store with matching URI, whose hash attribute is not in the given list of hashes.

4.1.8. Update object's validation time

For all objects in the store whose hash attribute matches the given hash, set the last validation time attribute to the given timestamp.

5. Acknowledgements

6. IANA Considerations

7. Security Considerations

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <http://www.rfc-editor.org/info/rfc5280>.

[RFC6481] Huston, G., Loomans, R., and G. Michaelson, "A Profile for Resource Certificate Repository Structure", RFC 6481, DOI 10.17487/RFC6481, February 2012, <http://www.rfc-editor.org/info/rfc6481>.

[RFC6482] Lepinski, M., Kent, S., and D. Kong, "A Profile for Route Origin Authorizations (ROAs)", RFC 6482, DOI 10.17487/RFC6482, February 2012, <http://www.rfc-editor.org/info/rfc6482>.

[RFC6486] Austein, R., Huston, G., Kent, S., and M. Lepinski, "Manifests for the Resource Public Key Infrastructure (RPKI)", RFC 6486, DOI 10.17487/RFC6486, February 2012, <http://www.rfc-editor.org/info/rfc6486>.

[RFC6487] Huston, G., Michaelson, G., and R. Loomans, "A Profile for X.509 PKIX Resource Certificates", RFC 6487, DOI 10.17487/RFC6487, February 2012, <http://www.rfc-editor.org/info/rfc6487>.

[RFC6488] Lepinski, M., Chi, A., and S. Kent, "Signed Object Template for the Resource Public Key Infrastructure (RPKI)", RFC 6488, DOI 10.17487/RFC6488, February 2012, <http://www.rfc-editor.org/info/rfc6488>.

[RFC6490] Huston, G., Weiler, S., Michaelson, G., and S. Kent, "Resource Public Key Infrastructure (RPKI) Trust Anchor Locator", RFC 6490, DOI 10.17487/RFC6490, February 2012, <http://www.rfc-editor.org/info/rfc6490>.

[RFC6493] Bush, R., "The Resource Public Key Infrastructure (RPKI) Ghostbusters Record", RFC 6493, DOI 10.17487/RFC6493, February 2012, <http://www.rfc-editor.org/info/rfc6493>.

8.2. Informative References

[I-D.tbruijnzeels-sidr-delta-protocol]
Bruijnzeels, T., Muravskiy, O., Weber, B., Austein, R., and D. Mandelberg, "RPKI Repository Delta Protocol", draft-tbruijnzeels-sidr-delta-protocol-03 (work in progress), December 2014.

Authors' Addresses

Tim Bruijnzeels
RIPE NCC

Email: tim@ripe.net

Oleg Muravskiy
RIPE NCC

Email: oleg@ripe.net