

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 22, 2016

T. Bruijnzeels
O. Muravskiy
RIPE NCC
B. Weber
Cobenian
R. Austein
Dragon Research Labs
D. Mandelberg
BBN Technologies
March 21, 2016

RPKI Repository Delta Protocol
draft-ietf-sidr-delta-protocol-02

Abstract

In the Resource Public Key Infrastructure (RPKI), certificate authorities publish certificates, including end entity certificates, Certificate Revocation Lists (CRL), and RPKI signed objects to repositories. Relying Parties (RP) retrieve the published information from those repositories. This document specifies a delta protocol which provides relying parties with a mechanism to query a repository for incremental updates, thus enabling the RP to keep its state in sync with the repository.

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 September 22, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Requirements notation	2
2. Introduction	2
3. RPKI Repository Delta Protocol Implementation	3
3.1. Informal Overview	3
3.2. Certificate Authority Use	4
3.3. Repository Server Use	5
3.3.1. Initialisation	5
3.3.2. Publishing Updates	5
3.4. Relying Party Use	7
3.4.1. Processing the Update Notification File	7
3.4.2. Processing a Snapshot File	8
3.4.3. Processing Delta Files	8
3.4.4. Polling the Update Notification File	9
3.5. File Definitions	9
3.5.1. Update Notification File	9
3.5.2. Snapshot File	11
3.5.3. Delta File	12
3.5.4. XML Schema	14
4. Security Considerations	16
5. IANA Considerations	16
6. Acknowledgements	16
7. Normative References	16
Authors' Addresses	17

1. Requirements notation

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. Introduction

In the Resource Public Key Infrastructure (RPKI), Certificate Authorities (CAs) publish certificates [RFC6487], RPKI signed objects [RFC6488], manifests [RFC6486], and CRLs to repositories. CAs may have an embedded mechanism to publish to these repositories, or they

may use a separate repository server and publication protocol. RPKI repositories are currently accessible using the rsync protocol, allowing Relying Parties (RPs) to synchronise a local copy of the RPKI repository used for validation with the remote repositories [RFC6481].

This document specifies an alternative repository access protocol based on notification, snapshot and delta files that a RP can retrieve over the HTTPS protocol. This allows RPs to perform either a full (re-)synchronisation of their local copy of the repository using snapshot files, or use delta files to keep their local repository updated after initial synchronisation.

This protocol is designed to be consistent (in terms of data structures) with the publication protocol [I-D.ietf-sidr-publication] and treats publication events of one or more repository objects as discrete events that can be communicated to relying parties. This approach helps to minimize the amount of data that traverses the network and thus helps minimize the amount of time until repository convergence occurs. This protocol also provides a standards based way to obtain consistent, point in time views of a single repository, eliminating a number of consistency related issues. Finally, this approach allows these discrete events to be communicated as immutable files, so that caching infrastructure can be used to reduce the load on a repository server when a large number of relying parties are querying it.

3. RPKI Repository Delta Protocol Implementation

3.1. Informal Overview

Certification Authorities (CA) in the RPKI use a repository server to publish their RPKI products, such as manifests, CRLs, signed certificates and RPKI signed objects. This repository server may be remote, or embedded in the CA engine itself. Certificates in the RPKI that use a repository server that supports this delta protocol include a special Subject Information Access (SIA) pointer referring to a notification file.

The notification file includes a globally unique session_id in the form of a version 4 UUID, and serial number that can be used by the Relying Party (RP) to determine if it and the repository are synchronised. Furthermore it includes a link to the most recent complete snapshot of current objects that are published by the repository server, and a list of links to delta files, for each revision starting at a point determined by the repository server, up to the current revision of the repository.

A RP that learns about a notification file location for the first time can download it, and then proceed to download the latest snapshot file, and thus create a local copy of the repository that is in sync with the repository server. The RP should remember the location of this notification file, the session_id and current serial number.

RPs are encouraged to re-fetch this notification file at regular intervals, but not more often than once per minute. After re-fetching the notification file, the RP may find that there are one or more delta files available that allow it to synchronise its local repository with the current state of the repository server. If no contiguous chain of deltas from RP's serial to the latest repository serial is available, or if the session_id has changed, the RP should perform a full resynchronisation instead.

As soon as the RP fetches new content in this way it should start a validation process. An example of a reason why a RP may not do this immediately is because it has learned of more than one notification location and it prefers to complete all its updates before validating.

The repository server may use caching infrastructure to reduce its load. It should be noted that snapshots and deltas for any given session_id and serial number contain an immutable record of the state of the repository server at a certain point in time. For this reason these files can be cached indefinitely. Notification files are polled by RPs to discover if updates exist, and for this reason notification files may not be cached for longer than one minute.

3.2. Certificate Authority Use

Certificate Authorities that use this delta protocol MUST include an instance of an SIA AccessDescription extension in resource certificates they produce, in addition to the ones defined in [RFC6487],

```
AccessDescription ::= SEQUENCE {  
    accessMethod OBJECT IDENTIFIER,  
    accessLocation GeneralName }
```

This extension MUST use an accessMethod of id-ad-rpkiNotify, see: [IANA-AD-NUMBERS],

```
id-ad OBJECT IDENTIFIER ::= { id-pkix 48 }  
id-ad-rpkiNotify OBJECT IDENTIFIER ::= { id-ad 13 }
```

The `accessLocation` MUST be a URI [RFC3986], using the 'https' scheme, that will point to the update notification file for the repository server that publishes the products of this CA certificate.

Relying Parties that do not support this delta protocol MUST NOT reject a CA certificate merely because it has an SIA extension containing this new kind of `AccessDescription`.

3.3. Repository Server Use

3.3.1. Initialisation

When the repository server initialises it must perform the following actions:

The server MUST generate a new random version 4 UUID to be used as the `session_id`

The server MUST then generate a snapshot file for serial number ONE for this new session that includes all currently known published objects that the repository server is responsible for. Note that this snapshot file MAY contain zero publish elements at this point if no objects have been submitted for publication yet.

This snapshot file MUST be made available at a URL that is unique to this `session_id` and serial number, so that it can be cached indefinitely.

The format and caching concerns for snapshot files are explained in more detail in Section 3.5.2.

After the snapshot file has been published the repository server MUST publish a new notification file that contains the new `session_id`, has serial number ONE, has one reference to the snapshot file that was just published, and that contains no delta references.

The format and caching concerns for update notification files are explained in more detail in Section 3.5.1.

3.3.2. Publishing Updates

Whenever the repository server receives updates from a CA it SHOULD generate new snapshot and delta files. However, if a publication server services a large number of CAs it MAY choose to combine updates from multiple CAs. If a publication server combines updates in this way, it MUST NOT postpone publishing for longer than one minute.

Updates must be processed as follows:

- o The new repository serial number MUST be one greater than the current repository serial number.
- o A new delta file MUST be generated for this new serial. This delta file MUST include all new, replaced and withdrawn objects for multiple CAs if applicable, as a single change set.
- o This delta file MUST be made available at a URL that is unique to the current session_id and serial number, so that it can be cached indefinitely.
- o The format and caching concerns for delta files are explained in more detail in Section 3.5.3.
- o The repository server MUST also generate a new snapshot file for this new serial. This file MUST contain all "publish" elements for all current objects.
- o The snapshot file MUST be made available at a URL that is unique to this session and new serial, so that it can be cached indefinitely.
- o The format and caching concerns for snapshot files are explained in more detail in Section 3.5.2.
- o The update notification file SHOULD be kept small, and in order to do so the repository server needs to make a decision about which delta files to support. Any older delta files that, when combined with all more recent delta files, will result in total size of deltas exceeding the size of the snapshot, MUST be excluded.
- o The server MAY also exclude more recent delta files if it finds that their usage by a small number of RPs that would be forced to perform a full synchronisation is outweighed by the performance penalty for all RPs in having a large update notification file. However the repository server SHOULD include all deltas for the last two hours.
- o A new notification file MUST now be created by the repository server. This new notification file MUST include a reference to the new snapshot file, and all delta files selected in the previous steps.
- o The format and caching concerns for update notification files are explained in more detail in Section 3.5.1.

If the repository server is not capable of performing the above for some reason, then it MUST perform a full re-initialisation, as explained above in Section 3.3.1.

3.4. Relying Party Use

3.4.1. Processing the Update Notification File

When a Relying Party (RP) performs RPKI validation and learns about a valid certificate with an SIA entry for the RRDP protocol, it SHOULD prefer to use this protocol as follows.

The RP SHOULD download the update notification file, unless an update notification file was already downloaded and processed from the same location in this validation run.

The RP MAY use a "User-Agent" header explained in section 5.5.3. of [RFC7231] to identify the name and version of the RP software used. This is not required, but would be useful to help track capabilities of Relying Parties in the event of changes to the RPKI standards.

When the RP downloads an update notification file it MUST verify the file format and validation steps described in section Section 3.5.1.3. If this verification fails, the file MUST be rejected.

The RP MUST verify whether the session_id in this update notification file matches the last known session_id for this update notification file location. If the session_id matches the last known session_id, then an RP MAY download and process missing delta files as described in section Section 3.4.3, provided that all delta files for serial numbers between the last processed serial number and the current serial number in the notification file can be processed this way.

If the session_id was not previously known, or if delta files could not be used, then the RP MUST update its last known session_id to this session_id and download and process snapshot file on the update notification file as described in section Section 3.4.2.

If neither update notification file and one snapshot file or delta files could be processed this way, the RP MUST issue an operator error, and SHOULD use an alternate repository retrieval mechanism if it is available.

3.4.2. Processing a Snapshot File

When the RP downloads a snapshot file it MUST verify the file format and validation steps described in Section 3.5.2.3. If this verification fails, the file MUST be rejected.

Furthermore the RP MUST verify that the hash of the contents of this file matches the hash on the update notification file that referenced it. In case of a mismatch of this hash, the file MUST be rejected.

If an RP retrieved a snapshot file that is valid according to the above criteria, it should perform the following actions:

The RP MUST verify that the `session_id` matches the `session_id` of the notification file. If the `session_id` values do not match the file MUST be rejected.

The RP MUST verify that the serial number of this snapshot file is greater than the last processed serial number for this `session_id`. If this fails the file MUST be rejected.

The RP SHOULD then add all publish elements to a local storage and update its last processed serial number to the serial number of this snapshot file.

3.4.3. Processing Delta Files

If an update notification file contains a contiguous chain of links to delta files from the last processed serial number to the current serial number, then RPs MUST attempt to download and process all delta files in order of serial number as follows.

When the RP downloads a delta file it MUST verify the file format and perform validation steps described in Section 3.5.3.3. If this verification fails, the file MUST be rejected.

Furthermore the RP MUST verify that the hash of the contents of this file matches the hash on the update notification file that referenced it. In case of a mismatch of this hash, the file MUST be rejected.

If an RP retrieved a delta file that is valid according to the above criteria, it should perform the following actions:

The RP MUST verify that the `session_id` matches the `session_id` of the notification file. If the `session_id` values do not match the file MUST be rejected.

The RP MUST verify that the serial number of this delta file is exactly one greater than the last processed serial number for this session_id, and if not this file MUST be rejected.

The RP SHOULD add all publish elements to a local storage and update its last processed serial number to the serial number of this snapshot file.

The RP SHOULD NOT remove objects from its local storage solely because it encounters a "withdraw" element, because this would enable a publication server to withdraw any object without the signing Certificate Authority consent. Instead it is RECOMMENDED that a RP uses additional strategies to determine if an object is still relevant for validation before removing it from its local storage.

3.4.4. Polling the Update Notification File

Once a Relying Party has learned about the location, session_id and last processed serial number of repository that uses the RRDP protocol, the RP MAY start polling the repository server for updates. However the RP MUST NOT poll for updates more often than once every 1 minute, and in order to reduce data usage RPs MUST use the "If-Modified-Since" header explained in section 3.3 of [RFC7232] in requests.

If an RP finds that updates are available it SHOULD download and process the file as described in Section 3.4.1, and initiate a new validation process. A detailed description of the validation process itself is out of scope of this document.

3.5. File Definitions

3.5.1. Update Notification File

3.5.1.1. Purpose

The update notification file is used by RPs to discover whether any changes exist between the state of the repository and the RP's cache. It describes the location of the files containing the snapshot and incremental deltas which can be used by the RP to synchronise with the repository.

3.5.1.2. Cache Concerns

A repository server MAY use caching infrastructure to cache the notification file and reduce the load of HTTPS requests. However, since this file is used by RPs to determine whether any updates are

available the repository server MUST ensure that this file is not cached for longer than 1 minute. An exception to this rule is that it is better to serve a stale notification file, then no notification file.

How this is achieved exactly depends on the caching infrastructure used. In general a repository server may find certain HTTP headers to be useful, such as: Cache-Control: max-age=60. Another approach can be to have the repository server push out new versions of the notification file to the caching infrastructure when appropriate.

Relying Parties SHOULD NOT cache the notification file for longer than 1 minute, regardless of the headers set by the repository server or CDN.

3.5.1.3. File Format and Validation

Example notification file:

```
<notification xmlns="http://www.ripe.net/rpki/rrdp"
  version="1"
  session_id="9df4b597-af9e-4dca-bdda-719cce2c4e28"
  serial="3">
  <snapshot uri="https://host/9d-8/3/snapshot.xml" hash="AB"/>
  <delta serial="3" uri="https://host/9d-8/3/delta.xml" hash="CD"/>
  <delta serial="2" uri="https://host/9d-8/2/delta.xml" hash="EF"/>
</notification>
```

Note: URIs and hash values in this example are shortened because of formatting.

The following validation rules must be observed when creating or parsing notification files:

- o A RP MUST reject any update notification file that is not well-formed, or which does not conform to the RELAX NG schema outlined in Section 3.5.4 of this document.
- o The XML namespace MUST be `http://www.ripe.net/rpki/rrdp`
- o The encoding MUST be US-ASCII
- o The version attribute in the notification root element MUST be 1
- o The session_id attribute MUST be a random version 4 UUID unique to this session

- o The serial attribute must be an unbounded, unsigned positive integer in decimal format indicating the current version of the repository.
- o The notification file MUST contain exactly one 'snapshot' element for the current repository version.
- o If delta elements are included they MUST form a contiguous sequence of serial numbers starting at a revision determined by the repository server, up to the serial number mentioned in the notification element.
- o The hash attribute in snapshot and delta elements must be the hexadecimal encoding of the SHA-256 hash of the referenced file. The RP MUST verify this hash when the file is retrieved and reject the file if the hash does not match.

3.5.2. Snapshot File

3.5.2.1. Purpose

A snapshot is intended to reflect the complete and current contents of the repository for a specific session and version. Therefore it MUST contain all objects from the repository current as of the time of the publication.

3.5.2.2. Cache Concerns

A snapshot reflects the content of the repository at a specific point in time, and for that reason can be considered immutable data. Snapshot files MUST be published at a URL that is unique to the specific session and serial.

Because these files never change, they MAY be cached indefinitely. However, in order to prevent that these files use a lot of space in caching infrastructure it is RECOMMENDED that a limited interval is used in the order of hours or days.

To avoid race conditions where an RP downloads a notification file moments before it's updated, Repository Servers SHOULD retain old snapshot files for at least 5 minutes after a new notification file is published.

3.5.2.3. File Format and Validation

Example snapshot file:

```
<snapshot xmlns="http://www.ripe.net/rpki/rrdp"
  version="1"
  session_id="9df4b597-af9e-4dca-bdda-719cce2c4e28"
  serial="2">
  <publish uri="rsync://rpki.ripe.net/Alice/Bob.cer">
    ZXhbbXBsZTE=
  </publish>
  <publish uri="rsync://rpki.ripe.net/Alice/Alice.mft">
    ZXhbbXBsZTI=
  </publish>
  <publish uri="rsync://rpki.ripe.net/Alice/Alice.crl">
    ZXhbbXBsZTM=
  </publish>
</snapshot>
```

The following rules must be observed when creating or parsing snapshot files:

- o A RP MUST reject any snapshot file that is not well-formed, or which does not conform to the RELAX NG schema outlined in Section 3.5.4 of this document.
- o The XML namespace MUST be `http://www.ripe.net/rpki/rrdp`.
- o The encoding MUST be US-ASCII.
- o The version attribute in the notification root element MUST be 1
- o The session_id attribute MUST match the expected session_id in the reference in the notification file.
- o The serial attribute MUST match the expected serial in the reference in the notification file.
- o Note that the publish element is defined in the publication protocol [I-D.ietf-sidr-publication]

3.5.3. Delta File

3.5.3.1. Purpose

An incremental delta file contains all changes for exactly one serial increment of the repository server. In other words a single delta will typically include all the new objects, updated objects and withdrawn objects that a Certification Authority sent to the repository server. In its simplest form the update could concern only a single object, but it is recommended that CAs send all changes

for one of their key pairs: i.e. updated objects as well as a new manifest and CRL as one atomic update message.

3.5.3.2. Cache Concerns

Deltas reflect the difference between two consecutive versions of a repository for a given session. For that reason deltas can be considered immutable data. Delta files **MUST** be published at a URL that is unique to the specific session and serial.

Because these files never change, they **MAY** be cached indefinitely. However, in order to prevent these files from using a lot of space in caching infrastructure it is **RECOMMENDED** that a limited interval is used in the order of hours or days.

To avoid race conditions where an RP downloads a notification file moments before it's updated, Repository Servers **SHOULD** retain old delta files for at least 5 minutes after they are no longer included in the latest notification file.

3.5.3.3. File Format and Validation

Example delta file:

```
<delta xmlns="http://www.ripe.net/rpki/rrdp"
  version="1"
  session_id="9df4b597-af9e-4dca-bdda-719cce2c4e28"
  serial="3">
  <publish uri="rsync://rpki.ripe.net/repo/Alice/Alice.mft"
    hash="50d8...545c">
    ZXhbbXBsZTQ=
  </publish>
  <publish uri="rsync://rpki.ripe.net/repo/Alice/Alice.crl"
    hash="5fb1...6a56">
    ZXhbbXBsZTU=
  </publish>
  <withdraw uri="rsync://rpki.ripe.net/repo/Alice/Bob.cer"
    hash="caeb...15c1"/>
</delta>
```

Note that a formal RELAX NG specification of this file format is included later in this document. A RP **MUST NOT** process any delta file that is incomplete or not well-formed.

The following validation rules must be observed when creating or parsing delta files:

- o A RP MUST reject any delta file that is not well-formed, or which does not conform to the RELAX NG schema outlined in Section 3.5.4 of this document.
- o The XML namespace MUST be `http://www.ripe.net/rpki/rrdp`.
- o The encoding MUST be US-ASCII.
- o The version attribute in the delta root element MUST be 1
- o The session_id attribute MUST be a random version 4 UUID unique to this session
- o The session_id attribute MUST match the expected session_id in the reference in the notification file.
- o The serial attribute MUST match the expected serial in the reference in the notification file.
- o Note that the publish and withdraw elements are defined in the publication protocol [I-D.ietf-sidr-publication]

3.5.4. XML Schema

The following is a RELAX NG compact form schema describing version 1 of this protocol.

```
#
# RelaxNG schema for RPKI Repository Delta Protocol (RRDP).
#

default namespace = "http://www.ripe.net/rpki/rrdp"

version = xsd:positiveInteger    { maxInclusive="1" }
serial  = xsd:nonNegativeInteger
uri     = xsd:anyURI
uuid    = xsd:string             { pattern = "[\0-9a-fA-F]+" }
hash    = xsd:string             { pattern = "[0-9a-fA-F]+" }
base64  = xsd:base64Binary

# Notification file: lists current snapshots and deltas

start |= element notification {
  attribute version    { version },
  attribute session_id { uuid },
  attribute serial     { serial },
  element snapshot {
    attribute uri { uri },
```

```
    attribute hash { hash }
  },
  element delta {
    attribute serial { serial },
    attribute uri    { uri },
    attribute hash   { hash }
  }*
}

# Snapshot segment: think DNS AXFR.

start |= element snapshot {
  attribute version    { version },
  attribute session_id { uuid },
  attribute serial     { serial },
  element publish     {
    attribute uri { uri },
    base64
  }*
}

# Delta segment: think DNS IXFR.

start |= element delta {
  attribute version    { version },
  attribute session_id { uuid },
  attribute serial     { serial },
  delta_element+
}

delta_element |= element publish {
  attribute uri { uri },
  attribute hash { hash }?,
  base64
}

delta_element |= element withdraw {
  attribute uri { uri },
  attribute hash { hash }
}

# Local Variables:
# indent-tabs-mode: nil
# comment-start: "# "
# comment-start-skip: "#[ \t]*"
# End:
```

4. Security Considerations

TBD

5. IANA Considerations

This document has no actions for IANA.

6. Acknowledgements

TBD

7. Normative References

[I-D.ietf-sidr-publication]

Weiler, S., Sonalker, A., and R. Austein, "A Publication Protocol for the Resource Public Key Infrastructure (RPKI)", draft-ietf-sidr-publication-07 (work in progress), September 2015.

[IANA-AD-NUMBERS]

"SMI Security for PKIX Access Descriptor",
<<http://www.iana.org/assignments/smi-numbers/smi-numbers.xhtml#smi-numbers-1.3.6.1.5.5.7.48>>.

[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>>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <<http://www.rfc-editor.org/info/rfc3986>>.

[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>>.

[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>>.
- [RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <<http://www.rfc-editor.org/info/rfc7231>>.
- [RFC7232] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Conditional Requests", RFC 7232, DOI 10.17487/RFC7232, June 2014, <<http://www.rfc-editor.org/info/rfc7232>>.

Authors' Addresses

Tim Bruijnzeels
RIPE NCC

Email: tim@ripe.net

Oleg Muravskiy
RIPE NCC

Email: oleg@ripe.net

Bryan Weber
Cobenian

Email: bryan@cobenian.com

Rob Austein
Dragon Research Labs

Email: sra@hacitrn.net

David Mandelberg
BBN Technologies

Email: david@mandelberg.org

SIDR
Internet-Draft
Intended status: Informational
Expires: September 22, 2016

O. Muravskiy
T. Bruijnzeels
RIPE NCC
March 21, 2016

RPKI Certificate Tree Validation by a Relying Party Tool
draft-ietf-sidr-rpki-tree-validation-00

Abstract

This document currently describes the approach to validate the content of the RPKI certificate tree, as used by the RIPE NCC RPKI Validator. This approach is independent of a particular object retrieval mechanism. This allows it to be used with repositories available over the rsync protocol, the RPKI Repository Delta Protocol, and repositories that use a mix of both.

This algorithm does not rely on content of repository directories, but uses the Authority Key Identifier (AKI) field of a manifest and a certificate revocation list (CRL) objects to discover manifest and CRL objects issued by a particular Certificate Authority (CA). It further uses the hashes of manifest entries to discover other objects issued by the CA.

If the working group finds that algorithm outlined here is useful for other implementations, we may either update future revisions of this document to be less specific to the RIPE NCC RPKI Validator implementation, or we may use this document as a starting point of a generic validation document and keep this as a detailed description of the actual RIPE NCC RPKI Validator implementation.

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 September 22, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Top-down Validation of a Single Trust Anchor Certificate Tree	3
2.1. Fetching the Trust Anchor Certificate Using the Trust Anchor Locator	4
2.2. Resource Certificate Validation	4
2.2.1. Finding most recent valid manifest and CRL	5
2.2.2. Manifest entries validation	6
2.3. Object Store Cleanup	6
3. Remote Objects Fetcher	6
3.1. Fetcher Operations	7
3.1.1. Fetch repository objects	7
3.1.2. Fetch single repository object	7
4. Local Object Store	8
4.1. Store Operations	8
4.1.1. Store Repository Object	8
4.1.2. Update object's last fetch time	8
4.1.3. Get objects by hash	8
4.1.4. Get certificate objects by URI	8
4.1.5. Get manifest objects by AKI	8
4.1.6. Delete objects for URI	8
4.1.7. Delete outdated objects	8
4.1.8. Update object's validation time	9
5. Acknowledgements	9
6. IANA Considerations	9
7. Security Considerations	9
8. References	10
8.1. Normative References	10
8.2. Informative References	11
Authors' Addresses	11

1. Introduction

In order to use information published in RPKI repositories, Relying Parties (RP) need to retrieve and validate the content of certificates, CRLs, and other RPKI signed objects. To validate a particular object, one must ensure that all certificates in the certificate chain up to the Trust Anchor (TA) are valid. Therefore the validation of a certificate tree is usually performed top-down, starting from the TA certificate and descending down the certificate chain, validating every encountered certificate and its products. The result of this process is a list of all encountered RPKI objects with a validity status attached to each of them. These results may later be used by a Relying Party in taking routing decisions, etc.

Traditionally RPKI data is made available to RPs through the repositories [RFC6481] accessible over rsync protocol. Relying parties are advised to keep a local copy of repository data, and perform regular updates of this copy from the repository (Section 5 of [RFC6481]). The RPKI Repository Delta Protocol [I-D.ietf-sidr-delta-protocol] introduces another method to fetch repository data and keep the local copy up to date with the repository.

This document describes how a Relying Party tool could discover RPKI objects to download, build certificate path, and validate RPKI objects, independently from what repository access protocol is used. To achieve this, it puts downloaded RPKI objects in an object store, where objects could be found by their URI, hash of their content, value of the object's AKI field, or combination of these. It also keeps track of download and validation time for every object, to perform cleanups of the local copy.

2. Top-down Validation of a Single Trust Anchor Certificate Tree

The validation of a Trust Anchor (TA) certificate tree starts from its TA certificate. To retrieve the TA certificate, a 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 7 of [RFC6487] and Section 2.2 of [RFC7730].

Then the TA certificate and all its subordinate objects are validated as described in Section 2.2.

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

Outdated objects are removed from the store as described in Section 2.3. This completes the validation of the TA certificate tree.

2.1. Fetching the Trust Anchor Certificate Using the Trust Anchor Locator

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

- o (Optional) If the Trust Anchor Locator contains a "prefetch.uris" field, pass the URIs contained there to the fetcher (see Section 3.1.1). (This field is a non-standard extension to the TAL format supported by the RIPE NCC Validator. It helps fetching non-hierarchical rsync repositories more efficiently.)
- o Extract the TA certificate URI from the TAL's URI section (see Section 2.1 of[RFC7730]) and pass to the object fetcher (Section 3.1.2).
- o Retrieve from the object 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 (see Section 2.1 of [RFC7730]).
- o If no, or more than one such objects are found, issue an error and stop validation process. Otherwise, use that object as the Trust Anchor certificate.

2.2. Resource Certificate Validation

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

- o If both the caRepository (Section 4.8.8.1 of [RFC6487]), and the id-ad-rpkiNotify (Section 3.5 of [I-D.ietf-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 object fetcher (see Section 3.1.1).
- o For a given resource certificate, find its 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.

- o Compare the URI found in the given resource certificate's id-ad-rpkiManifest field (Section 4.8.8.1 of [RFC6487]) with the URI of the manifest found in the previous step. If they are different, issue a warning.
- o Perform manifest entries validation as described in Section 2.2.2.
- o Validate all resource certificate objects found on the manifest, using the CRL object found on the manifest, according to Section 7 of [RFC6487].
- o Validate all ROA objects found on the manifest, using the CRL object found on the manifest, according to the Section 4 of [RFC6482].
- o Validate all Ghostbusters Record objects found on the manifest, using the CRL object found on the manifest, according to the Section 7 of [RFC6493].
- o 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 its 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 manifestNumber field (Section 4.2.1 of [RFC6486]), for which all following conditions are met:

- o 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.
- o The manifest's certificate AKI equals the above CRL's AKI
- o The above CRL is a valid object according to Section 6.3 of [RFC5280]
- o 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:

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

2.3. Object Store Cleanup

At the end of the TA tree validation the store cleanup is performed:

- o Given all objects that were validated during the current validation run, remove from the store (Section 4.1.7) all objects whose URI attribute matches the URI of one of the validated objects, but the content's hash is different.
- o Remove from the store all objects that were last validated more than 7 days ago.
- o Remove from the store all objects that were downloaded more than 2 hours ago and have never been used in a validation process.

The time intervals used in the steps above are a matter of local policy.

3. Remote Objects Fetcher

The fetcher is responsible for downloading objects from remote repositories (described in Section 3 of [RFC6481]) using rsync protocol ([rsync]), or RPKI Repository Delta Protocol (RRDP) ([I-D.ietf-sidr-delta-protocol]).

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 repository. For RRDP repository it points to the repository's notification file.

The fetcher performs following steps:

- o If the given URI has been downloaded recently (as specified by the local policy), skip all following steps.
- o Download the remote objects using the URI provided (for an rsync repository use a recursive mode).
- o 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].
- o For every downloaded valid object, record it in the object store (Section 4.1.1), and set its 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:

- o If the given URI has been downloaded recently (as specified by the local policy), skip all following steps.
- o Download the remote object using the URI provided.
- o Try to parse the 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].
- o If the downloaded object is not valid, issue an error and skip further steps.

- o Delete objects from the object store (Section 4.1.6) whose URI matches the URI given.
- o Put validated object in the object store (Section 4.1.1), and set its 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 its 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.

4.1.1.8. 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

This document describes the algorithm as it is implemented by the software development team at the RIPE NCC. The original idea behind it was outlined by Tim Bruijnzeels. The authors would also like to acknowledge contributions by Carlos Martinez, Andy Newton, and Rob Austein.

6. IANA Considerations

This document has no actions for IANA.

7. Security Considerations

This algorithm uses the content of a manifest object to discover other objects issued by a particular CA. It verifies that the manifest is located in the publication point designated in the CA Certificate. However, if there are other (not enlisted in the manifest) objects located in that publication point directory, they will be ignored, even if their content is correct and they are issued by the same CA as the manifest.

In contrast, objects whose content hash matches the hash listed in the manifest, but that are not located in the publication directory listed in their CA certificate, will be used in the validation process (although a warning will be issued in that case).

The store cleanup procedure described in Section 2.3 tries to minimise removal and subsequent re-fetch of objects that are published in some repository but not used in the validation. Once such objects are removed from the remote repository, they will be discarded from the local object store after a period of time specified by a local policy. By generating an excessive amount of syntactically valid RPKI objects, a man-in-the-middle attack rendered between a validating tool and a repository could force an implementation to fetch and store those objects in the object store before they are being validated and discarded, leading to an out-of-memory or out-of-disk-space conditions, and, subsequently, a denial of service.

8. References

8.1. Normative References

- [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>>.
- [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>>.
- [RFC7730] Huston, G., Weiler, S., Michaelson, G., and S. Kent, "Resource Public Key Infrastructure (RPKI) Trust Anchor Locator", RFC 7730, DOI 10.17487/RFC7730, January 2016, <<http://www.rfc-editor.org/info/rfc7730>>.

8.2. Informative References

- [I-D.ietf-sidr-delta-protocol]
Bruijnzeels, T., Muravskiy, O., Weber, B., Austein, R.,
and D. Mandelberg, "RPKI Repository Delta Protocol",
draft-ietf-sidr-delta-protocol-02 (work in progress),
March 2016.
- [rsync] "Rsync home page", <<https://rsync.samba.org>>.

Authors' Addresses

Oleg Muravskiy
RIPE NCC

Email: oleg@ripe.net

Tim Bruijnzeels
RIPE NCC

Email: tim@ripe.net

Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 22, 2016

G. Huston
G. Michaelson
APNIC
C. Martinez
LACNIC
T. Bruijnzeels
RIPE NCC
A. Newton
ARIN
A. Aina
AFRINIC
March 21, 2016

RPKI Validation Reconsidered
draft-ietf-sidr-rpki-validation-reconsidered-03

Abstract

This document proposes an alternative to the certificate validation procedure specified in RFC6487 that reduces aspects of operational fragility in the management of certificates in the RPKI.

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 September 22, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Certificate Validation in the RPKI	2
3. Operational Considerations	3
4. An Amended RPKI Certification Validation Process	4
4.1. Changes to existing standards	4
4.2. An example	5
5. Security Considerations	7
6. IANA Considerations	7
7. Acknowledgements	7
8. Normative References	7
Authors' Addresses	8

1. Introduction

This document proposes an alternative to the certificate validation procedure specified in RFC6487 that reduces aspects of operational fragility in the management of certificates in the RPKI.

2. Certificate Validation in the RPKI

As currently defined in section 7.2 of [RFC6487], validation of PKIX certificates that conform to the RPKI profile relies on the use of a path validation process where each certificate in the validation path is required to meet the certificate validation criteria.

These criteria require in particular that the resources on each certificate in the validation path are "encompassed" by the resources on the issuing certificate. The first certificate in the path is required to be a trust anchor, and its resources are considered valid by definition.

For example, in the following sequence:

Certificate 1 (trust anchor):

Issuer TA, Subject TA, Resources 192.0.2.0/24, AS64496-AS64500

Certificate 2:

Issuer TA, Subject CA1, Resources 192.0.2.0/24, AS64496-AS64500

Certificate 3:

Issuer CA1, Subject CA2, Resources 192.0.2.0/24, AS64496-AS64500

ROA 1:

Embedded Certificate 4 (EE certificate):

Issuer CA2, Subject R1, Resources 192.0.2.0/24

Prefix 192.0.2.0/24, Max Length 24, ASN 64496

All certificates in this scenario are considered valid in that the resources on each certificate are encompassed by the issuing certificate. The roa "ROA1" is also considered valid here in this regard - the prefix is encompassed by the embedded EE certificate.

3. Operational Considerations

Resource allocations can change in the RPKI. And this can lead to situations where an "over-claiming" certificate is introduced.

Consider the following sequence:

Certificate 1 (trust anchor):

Issuer TA, Subject TA, Resources 192.168.2.0/24, AS64496-AS64500

Certificate 2:

Issuer TA, Subject CA1, Resources 192.168.2.0/24

Certificate 3:

Issuer CA1, Subject CA2, Resources 192.168.2.0/24, AS64496-AS64500

ROA 1:

Embedded Certificate 4 (EE certificate):

Issuer CA2, Subject R1, Resources 192.168.2.0/24

Prefix 192.168.2.0/24, Max Length 24, ASN 64496

Here Certificate 2 from the previous example was re-issued by TA to CA1 and certain AS resources were removed. However, CA1 failed to re-issue a new Certificate 3 to CA2. As a result Certificate 3 is now over-claiming and considered invalid, and by recursion ROA1 issued by CA2 is also invalid.

It should be noted that CA2 is not claiming any resources on ROA1 that it cannot receive on a new Certificate 3. If CA1 would only re-issue a Certificate 3 without the AS resources to CA2, then ROA1 would be considered valid without the need for any further action by CA2.

[RFC6492] describes the protocol for provisioning resource certificates. In this protocol new resource certificates are always issued by request of a child. If that protocol were strictly followed then CA1 would have known that its resource set was about to shrink, and it would have known that it issued some of those resources to its child CA2.

The protocol currently lacks normative wording on how CAs should deal with this situation, but one can imagine amending the protocol with normative instructions that would require CA1 to refuse to request a certificate with a shrunk resource set until all of its children would have requested new shrunk certificates where applicable. And that would forbid any parent CA to pro-actively re-issue a certificate with shrunk resource set before receiving a certificate re-issuance request from its child CA.

In practice such a model is unworkable for the CA higher in the path, because it has no control over if and when it can shrink a certificate for its children. Therefore higher level CAs will pro-actively re-issue shrunk resource certificates when resources are no longer validly held by a child.

The question here is whether the impact of such a re-issuance should be limited to just the resources that seem to be under dispute between TA and CA1, or all resources issued to CA2.

4. An Amended RPKI Certification Validation Process

4.1. Changes to existing standards

The following is a amended specification of certificate validation as described in section 7.2 item number 6 of certificate validation in [RFC6487] that describes the validation of resources in the RPKI path:

The Relying Party MUST keep a set of verified resources for the certificate independent of the RFC3779 extension itself, that is built up using the following approach:

For any of the resource extensions that use the "inherit" element as described in sections 2.2.3.5 and 3.2.3.3 of [RFC3779], the corresponding resources of this type should be

taken from the parent certificate, where this issuer is the subject.

For any other resources the intersection of the quoted resources on this certificate and the parent certificate is kept. If any resources were found on this certificate that were not present on the parent certificate a warning SHOULD be issued to help operators rectify this situation.

If the the set of verified resources obtained this way is empty, then the certificate MUST be considered invalid.

Note that if this approach would be used in the example we cite in section 3 of this document, Certificate 3 would have a verified resource set that contains only "192.0.2.0/24", and a warning would be issued with regards to resources "AS64496-AS64500". ROA1 would be considered valid because the quoted prefix was also part of the verified resource set of the embedded Certificate 4.

4.2. An example

Consider the following example under the amended approach:

Certificate 1 (trust anchor):

Issuer TA, Subject TA, Resources 192.168.2.0/24, AS64496-AS64500

Verified resources: 192.168.2.0/24, AS64496-AS64500

Warnings: none

Certificate 2:

Issuer TA, Subject CA1, Resources 192.168.2.0/24

Verified resources: 192.168.2.0/24

Warnings: none

Certificate 3:

Issuer CA1, Subject CA2, Resources 192.168.2.0/24, AS64496-AS64500

Verified resources: 192.168.2.0/24

Warnings: overclaim for AS64496-AS64500

ROA 1:

Embedded Certificate 4 (EE certificate):

Issuer CA2, Subject R1, Resources 192.168.2.0/24

Verified resources: 192.168.2.0/24

Warnings: none

Prefix 192.168.2.0/24, Max Length 24, ASN 64496

ROA1 is considered valid because the prefix matches the verified resources on the embedded EE certificate.

ROA 2:

Embedded Certificate 5 (EE certificate):

Issuer CA2, Subject R2, Resources 192.168.3.0/24

Verified resources: none

Warnings: overclaim for 192.168.3.0/24

Prefix 192.168.3.0/24, Max Length 24, ASN 64496

ROA2 is considered invalid because the prefix does not match the verified resources on the embedded EE certificate. The amended approach cannot lead to ROAs showing up as valid for resources that are not verified on the full path from the Trust Anchor down to the ROA.

5. Security Considerations

The problem described in section 3 of this document has not occurred to date. So one could consider this a low probability problem today. However the potential impact on routing security would be high if the inconsistency occurred near the apex of the RPKI hierarchy and would invalidate the entirety of the sub-tree located below the point of this inconsistency.

The proposed process does not change the probability of this problem, but it limits the impact to just the resources that are under dispute. As far as the authors can see there are no real new problems introduced by this approach.

It should be noted that although this is a problem with a low probability today this is largely due to the fact that most current RPKI systems use their own Trust Anchor and do not support any large number of delegated CAs. If this changes and the issuance and publication of a certificate, by the parent, and its use, by a child, are handled by different organisations more commonly, then the probability of this problem will increase.

6. IANA Considerations

No updates to the registries are suggested by this document.

7. Acknowledgements

TBA.

8. Normative References

- [RFC3779] Lynn, C., Kent, S., and K. Seo, "X.509 Extensions for IP Addresses and AS Identifiers", RFC 3779, DOI 10.17487/RFC3779, June 2004, <<http://www.rfc-editor.org/info/rfc3779>>.
- [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>>.
- [RFC6492] Huston, G., Loomans, R., Ellacott, B., and R. Austein, "A Protocol for Provisioning Resource Certificates", RFC 6492, DOI 10.17487/RFC6492, February 2012, <<http://www.rfc-editor.org/info/rfc6492>>.

Authors' Addresses

Geoff Huston
Asia Pacific Network Information Centre
6 Cordelia St
South Brisbane, QLD 4101
Australia

Phone: +61 7 3858 3100
Email: gih@apnic.net

George Michaelson
Asia Pacific Network Information Centre
6 Cordelia St
South Brisbane, QLD 4101
Australia

Phone: +61 7 3858 3100
Email: ggm@apnic.net

Carlos M. Martinez
Latin American and Caribbean IP Address Regional Registry
Rambla Mexico 6125
Montevideo 11400
Uruguay

Phone: +598 2604 2222
Email: carlos@lacnic.net

Tim Bruijnzeels
RIPE Network Coordination Centre
Singel 258
Amsterdam 1016 AB
The Netherlands

Email: tim@ripe.net

Andrew Lee Newton
American Registry for Internet Numbers
3635 Concorde Parkway
Chantilly, VA 20151
USA

Email: andy@arin.net

Alain Aina
African Network Information Centre (AFRINIC)
11th Floor, Raffles Tower
Cybercity, Ebene
Mauritius

Phone: +230 403 51 00
Email: aalain@afarinic.net

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: June 16, 2016

T. King
D. Kopp
DE-CIX
A. Lambrianidis
AMS-IX
A. Fenoux
France-IX
December 14, 2015

Signaling RPKI Validation Results from a Route-Server to Peers
draft-kklf-sidr-route-server-rpki-light-00

Abstract

This document defines the usage of the BGP Prefix Origin Validation State Extended Community [I-D.ietf-sidr-origin-validation-signaling] to signal RPKI validation results from a route-server to its peers. Upon reception of RPKI validation results peers can use this information in their local routing decision process.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" are to be interpreted as described in [RFC2119] only when they appear in all upper case. They may also appear in lower or mixed case as English words, without normative meaning.

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 June 16, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Signaling RPKI Validation Results from a Route-Server to Peers	3
3. Operational Recommendations	3
3.1. Local Routing Decision Process	3
3.2. Route-Server Receiving the BGP Prefix Origin Validation State Extended Community	3
3.3. Error Handling at Peers	4
4. IANA Considerations	4
5. Security Considerations	4
6. References	4
6.1. Normative References	4
6.2. Informative References	5
Authors' Addresses	5

1. Introduction

RPKI-based route origin validation [RFC6480] can be a significant operational burden for BGP peers to implement and adopt. In order to boost acceptance and usage of RPKI and ultimately increase the security of the Internet routing system, IXPs may provide RPKI-based route origin validation at the route-server [I-D.ietf-idr-ix-bgp-route-server]. The result of this route origin validation is signaled to peers by using the BGP Prefix Origin Validation State Extended Community as introduced in [I-D.ietf-sidr-origin-validation-signaling].

Peers receiving the route origin validation result from the route-server(s) can use this information in their local routing decision process for acceptance, rejection, preference, or other traffic engineering purposes of a particular route.

2. Signaling RPKI Validation Results from a Route-Server to Peers

The BGP Prefix Origin Validation State Extended Community (as defined in [I-D.ietf-sidr-origin-validation-signaling]) is utilized for signaling RPKI validation result from a route-server to peers.

[I-D.ietf-sidr-origin-validation-signaling] proposes an encoding of the RPKI validation result [RFC6811] as follows:

Value	Meaning
0	Valid
1	Not found
2	Invalid

Table 1

This encoding is re-used. Route-servers providing RPKI-based route origin validation set the validation state according to the RPKI validation result (see [I-D.ietf-sidr-rpki-validation-reconsidered]).

3. Operational Recommendations

3.1. Local Routing Decision Process

A peer receiving an RPKI validation result from the route server MAY use the information in its own local routing decision process. The local routing decision process SHOULD apply to the rules as described in section 5 [RFC6811].

A peer receiving an RPKI validation result from the route server MAY redistribute this information within its own AS.

3.2. Route-Server Receiving the BGP Prefix Origin Validation State Extended Community

An IXP route-server receiving routes from its peers containing the BGP Prefix Origin Validation State Extended Community MUST remove the extended community before the route is re-distributed to its peers. This is required regardless of whether the route-server is executing RPKI origin validation or not.

Failure to do so would allow opportunistic peers to advertise routes tagged with arbitrary RPKI validation results via a route-server, influencing maliciously the decision process of other route-server peers.

3.3. Error Handling at Peers

A route sent by a route-server SHOULD only contain none or one BGP Prefix Origin Validation State Extended Community.

A peer receiving a route from a route-server containing more than one BGP Prefix Origin Validation State Extended Community SHOULD only consider the largest value (as described in Table 1) in the validation result field and disregard the other values. Values larger than two in the validation result field MUST be disregarded.

4. IANA Considerations

None.

5. Security Considerations

A route-server could be misused to spread malicious RPKI validation results. However, peers have to trust the route-server anyway as it collects and redistributes BGP routing information to other peers.

The introduction of a mechanisms described in this document does not pose a new class of attack vectors to the relationship between route-servers and peers.

6. References

6.1. Normative References

- [I-D.ietf-sidr-rpki-validation-reconsidered]
Huston, G., Michaelson, G., Martinez, C., Bruijnzeels, T., Newton, A., and A. Aina, "RPKI Validation Reconsidered", draft-ietf-sidr-rpki-validation-reconsidered-02 (work in progress), October 2015.
- [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>>.
- [RFC4360] Sangli, S., Tappan, D., and Y. Rekhter, "BGP Extended Communities Attribute", RFC 4360, DOI 10.17487/RFC4360, February 2006, <<http://www.rfc-editor.org/info/rfc4360>>.
- [RFC6811] Mohapatra, P., Scudder, J., Ward, D., Bush, R., and R. Austein, "BGP Prefix Origin Validation", RFC 6811, DOI 10.17487/RFC6811, January 2013, <<http://www.rfc-editor.org/info/rfc6811>>.

6.2. Informative References

- [I-D.ietf-idr-ix-bgp-route-server]
Jasinska, E., Hilliard, N., Raszuk, R., and N. Bakker,
"Internet Exchange BGP Route Server", draft-ietf-idr-ix-
bgp-route-server-09 (work in progress), October 2015.
- [I-D.ietf-sidr-origin-validation-signaling]
Mohapatra, P., Patel, K., Scudder, J., Ward, D., and R.
Bush, "BGP Prefix Origin Validation State Extended
Community", draft-ietf-sidr-origin-validation-signaling-07
(work in progress), November 2015.
- [RFC6480] Lepinski, M. and S. Kent, "An Infrastructure to Support
Secure Internet Routing", RFC 6480, DOI 10.17487/RFC6480,
February 2012, <<http://www.rfc-editor.org/info/rfc6480>>.

Authors' Addresses

Thomas King
DE-CIX Management GmbH
Lichtstrasse 43i
Cologne 50825
DE

Email: thomas.king@de-cix.net

Daniel Kopp
DE-CIX Management GmbH
Lichtstrasse 43i
Cologne 50825
DE

Email: daniel.kopp@de-cix.net

Aristidis Lambrianidis
Amsterdam Internet Exchange
Frederiksplein 42
Amsterdam 1017 XN
NL

Email: aristidis.lambrianidis@ams-ix.net

Arnaud Fenioux
France-IX
88 Avenue Des Ternes
Paris 75017
FR

Email: afenieux@franceix.net

Secure Inter-Domain Routing
Internet-Draft
Intended status: Informational
Expires: July 30, 2016

X. Lee
X. Liu
Z. Yan
G. Geng
Y. Fu
CNNIC

January 27, 2016

RPKI Deployment Considerations: Problem Analysis and Alternative
Solutions
draft-lee-sidr-rpki-deployment-01

Abstract

With the global deployment of RPKI, a lot of concerns about technical problems have been and will be raised. In this draft, we collect and analyze the problems that have appeared or that seem likely to appear during the process of RPKI deployment, and suggest some solutions to address or mitigate these problems.

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 July 30, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. RPKI Architecture	2
1.2. Status of RPKI Deployment	3
2. Terminology	4
3. Considerations of RPKI Deployment	4
3.1. More than One TA	4
3.2. Problems of CAs	5
3.2.1. Misoperations	5
3.2.2. Unilateral Resource Revocation	5
3.3. Mirror World Attacks	5
3.4. Data Synchronization	6
3.5. Problems of Staged and Incomplete Deployment	6
3.6. Low Validation Accuracy	7
4. Alternative Solutions to RPKI Deployment Problems	8
4.1. Solutions to Multiple TAs	8
4.2. Solutions to Misbehaved CAs	8
4.3. Solutions to Data Synchronization	9
4.4. Solutions to Incomplete Deployment and Low Validation Accuracy	9
5. Security Considerations	9
6. IANA Considerations	9
7. Acknowledgements	10
8. References	10
8.1. Normative References	10
8.2. Informative References	10
Authors' Addresses	11

1. Introduction

1.1. RPKI Architecture

In RPKI, CAs (Certification Authorities) are organized in a hierarchical structure which is aligned to the existing INR (Internet Number Resources) allocation hierarchy (including IP prefixes and AS numbers). Each INR allocation requires corresponding resource certificates to attest to it, for security. In RPKI, two types of resource certificates [RFC6480] are generated as adjuncts to this allocation process: CA certificates and EE (End-entity) certificates. CA certificates attest to the INR holdings; EE certificates are primarily used for the validation of ROAs (Route Origin Authorizations) [RFC6482]. ROAs are used to bind IP prefixes to the

AS that is permitted to originate routes for these IP prefixes. Manifests [RFC6486] are also validated by using EE certificates. Manifests are used to ensure the integrity of the RPKI repository system.

The process of using the RPKI to verify the origin of a route is as follows.

1. CAs, including IANA (Internet Assigned Numbers Authority), five RIRs (Regional Internet Registries), NIRs (National Internet Registries) and ISPs (Internet Service Providers), publish authoritative objects (including resource certificates, ROAs, Manifest and so on) into their repositories.
2. RPs (Relying Parties) all over the world collect (using rsync or RRDP protocol) and verify (using rcynic or RPSTIR) the RPKI objects from these repositories, and provide the results of verification to BGP border routers.
3. Finally, BGP border routers can make use of these results to verify the route origin information in the BGP update messages they receive.

1.2. Status of RPKI Deployment

Each of the five RIRs has initiated the deployment of RPKI, and each now offers RPKI services to its members. A number of countries (Ecuador, Japan, Bangladesh, etc.) have also started to test and deploy RPKI internally. In order to promote the deployment of RPKI, ICANN (Internet Corporation for Assigned Names and Numbers), the five RIRs, many NIRs and companies have making continuous efforts to solve the existing problems and improve the corresponding policies and technical standards.

However, RPKI is still in its early stages of global deployment. According to the data provided by RPKI Dashboard as of January 2016, the current routing table holds about 628,858 IP prefixes in total, and the RPKI validation state has been determined for 39584 IP prefixes, which means that only 6.29% of the prefixes in the routing table can be validated using the RPKI. Table 1 details of the RPKI "adoption rate" (the percentage of members deployed RPKI) in each of the RIRs.

RIR	AFRINIC	APNIC	ARIN	LACNIC	RIPE NCC
Adoption Rate	1.65%	3.1%	1.02%	18.37%	11.35%

Table 1. Adoption rate of RPKI in 5 RIRs

As we can see from Table 1, LACNIC has the highest adoption rate, which is about 18.37%. While the adoption rates in ARIN and AFRINIC are much lower, which are only 1.02% and 1.65% respectively.

RIPE NCC provides some statistics regarding the number of resource certificates and ROAs in each RIR. From these statistics we find a good sign that the global deployment status of RPKI rises gradually, and with its further evolution, the global adoption rate of RPKI should achieve a faster growth.

2. Terminology

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].

3. Considerations of RPKI Deployment

During the process of incremental deployment of RPKI, several technical problems have appeared and may appear. In this section, we attempt to collect and analyze the problems which seem most critical.

3.1. More than One TA

A TA (Trust Anchor) is an authoritative entity represented by a public key and its associated data [RFC5914]. The public key is used to verify digital signatures and the associated data describes the types of information and actions for which the TA is authoritative. There are more than one TA in the RPKI architecture today, for example, IANA and five RIRs are candidates to be default TAs.

With more than one TA, there is no technical mechanism to prevent two or more TAs from asserting control over the same set of INRs accidentally or maliciously, which means that certificates might be issued for allocations of the overlapping INRs. This, in turn, may lead to inconsistent and conflicting assertions about to whom the specific INRs have been allocated. This kind of problem obviously may cause resource conflicts on the Internet.

3.2. Problems of CAs

3.2.1. Misoperations

Considering misconfiguration is inevitable and the significant impact it may cause, misconfiguration of CAs in RPKI is a potential risk in actual deployment.

The misconfiguration of CAs in RPKI may lead to serious consequences similar to those caused by malicious attacks (black-hole routes, traffic interception, and denial-of-service attacks). For example, misconfigurations of an ROA (such as adding a new ROA, whacking an existing ROA) may cause all routes covered by this ROA to become invalid.

3.2.2. Unilateral Resource Revocation

In the RPKI architecture, there is a risk that CAs have the power to unilaterally revoke the INRs which have been allocated to their descendants, just by revoking corresponding CA certificates [RFC6480].

This is a natural aspect of PKIs and it is a necessary capability for CAs as they manage re-allocation of resources within their domains. However, if revocation occurs accidentally, or because the CA has been compelled by authorities, the results can be significant. Specifically, all RPs will view the origin assertions by the CA (and its descendants) to be invalid. This may cause ISPs to deprefer routes to the affected prefixes.

3.3. Mirror World Attacks

In mirror world attacks, a malicious CA presents one view of the RPKI repository (that it manages) to some RPs, and a different view to others.

Since a CA in the RPKI can control everything in its own repository, there are possibilities that a malicious CA may perform these attacks easily. For example, a malicious CA presents the correct view of its repository to some RPs, but a forged view (e.g. the CA adds a specific ROA) to the others. When these deceived RPs offer their validation results to BGP routers, the routers may abandon the legitimate routes which are considered to be invalid according to the validation results they have received.

3.4. Data Synchronization

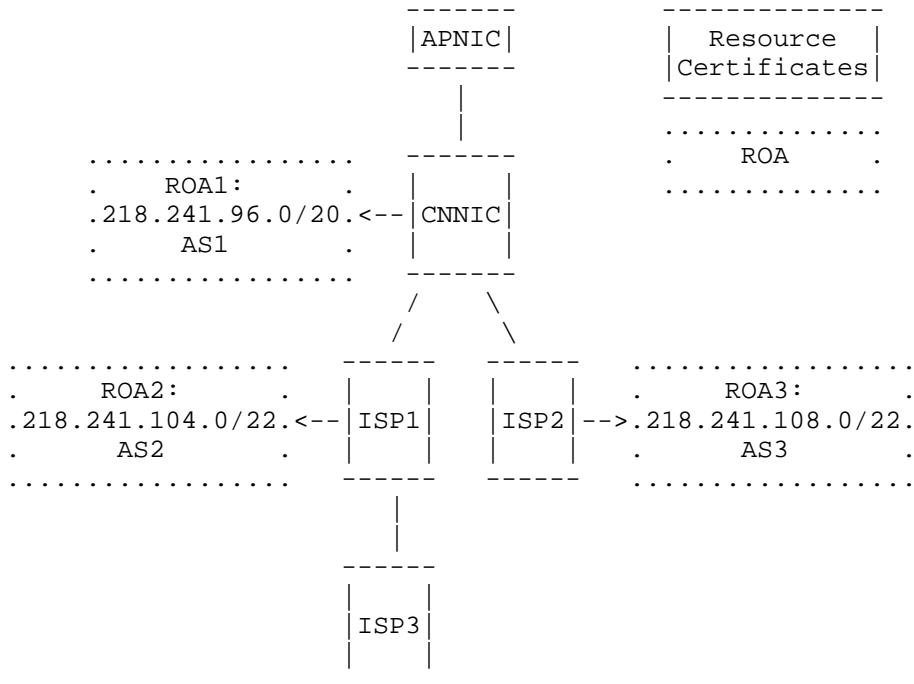
It is required in [RFC6480] that all repositories must be accessible via rsync protocol which is used by RPs to get the RPKI objects in the global distributed repositories. However, the rsync protocol is considered to be controversial with its following disadvantages:

1. Lack of standards and non-modular implementation: Although rsync is widely adopted in backup, restore, and file transfer, it has not been standardized by IETF. And the rsync implementation is non-modular, making it difficult to use its source code.
2. Not good enough in efficiency, scalability and security: Rsync is efficient when it is used between one client and one server. However, in RPKI a large number of clients may regularly connect to the repository server at the same time. Rsync is not efficient and scalable enough to deal with this concurrent case. Moreover, rsync itself provides little security (no content encryption and weak authentication especially in old versions) while transferring data.
3. Underlying overhead caused by repository updates during active data transmissions: During data transmissions between RPs and the repository, a new update to the repository may cause data inconsistency between them. And in order to rectify this inconsistency, extra overhead costs (such as performing the synchronization once more) are required.

3.5. Problems of Staged and Incomplete Deployment

Since the global deployment of RPKI is an incremental and staged process, unexpected problems may appear during this process. Let's take an example to explain why the incomplete deployment of RPKI may cause legitimate routes to be misclassified into invalid. In Fig. 1, we make the following assumptions:

1. CNNIC, ISP1 and ISP2 have deployed the RPKI, but ISP3 has not yet. ISP1 and ISP2 received allocations from CNNIC, and ISP3 received its allocation from ISP1.
2. CNNIC allocated IP prefix 218.241.104.0/22 to ISP1 and 218.241.108.0/22 to ISP2.
3. Three ROAs (ROA1, ROA2, ROA3) are issued respectively by CNNIC, ISP1 and ISP2.



An example of incomplete deployment

Now ISP3 announces to be the origination of 218.241.106.0/23. When other entities receive this announcement, they can validate it with ROAs information. Since prefix 218.241.104.0/22 described in ROA2 encompasses prefix 218.241.106.0/23 and no matching ROA describes 218.241.106.0/23 could be found [RFC6483], the announcement for prefix 218.241.106.0/23 will be considered to be invalid.

3.6. Low Validation Accuracy

The route origin validation accuracy refers to the percentage of valid routes. i.e., $\text{Accuracy} = \frac{\text{number_of_valid_routes}}{\text{number_of_valid_routes} + \text{number_of_invalid_routes}}$.

As we can see from Table 2, the accuracy of route origin validation in the five RIRs differs a lot. LACNIC and RIPE NCC have the highest validation accuracy and both of them are over 90%, while the accuracy in APNIC is less than 70%. Many reasons may account for the low validation accuracy, such as misconfigurations, low RPKI adoption rates, etc.

RIR	Total	Valid	Invalid	Unknown	Accuracy	Adoption Rate
AFRI-NIC	14948	242	5	14701	97.98%	1.65%
APNIC	158020	3332	1564	153124	68.06%	3.1%
ARIN	219779	1911	337	217531	85.01%	1.02%
LACNIC	76841	13379	736	62726	94.79%	18.37%
RIPE	15925	16771	1307	141178	92.77%	11.35%
NCC	6					

Table 2. Route Origin Validation Accuracy in 5 RIRs

4. Alternative Solutions to RPKI Deployment Problems

In this section, we propose and analyze the alternative solutions and strategies to solve or mitigate the problems mentioned in Section 3.

4.1. Solutions to Multiple TAs

The RIRs are trying to continually evolve RPKI, including the migration to a single GTA (Global Trust Anchor) as the root of the RPKI hierarchical structure. ICANN and RIRs have developed a technical testbed with an RPKI GTA. It's assumed that there must be a single root trust anchor eventually. With this single root trust anchor deployed, the risks of resource conflicts (at the level of RIR certificates) could be significantly reduced.

However, this solution cedes more power to ICANN and thus might exacerbate the risk of "Unilateral Resource Revocation" (power imbalance) mentioned in Section 3.2.2.

4.2. Solutions to Misbehaved CAs

S. Kent et al. put forward a collection of mechanisms named "Suspenders". "Suspenders" is designed to address the adverse effects on INR holders which were caused by CAs' accidental or deliberate misbehavior or attacks on CAs and repositories. This mechanism imports two new objects: an INRD (Internet Number Resource Declaration) file and a LOCK object. The INRD file is external to the RPKI repository, and it contains the most recent changes that were made by the INR holder. The LOCK object is published in the INR holder's repository. It contains a URL which points to the INRD file, and a public key used to verify the signature of INRD file.

Whenever the RPs detect the inconsistencies between the actual changes and the INRD file, they can determine individually whether to accept these changes or not.

4.3. Solutions to Data Synchronization

A number of alternative protocols have been presented to take the place of "rsync" protocol due to its shortcomings mentioned above.

1) RRDP

T. Bruijnzeels et al. have proposed an alternative protocol (RRDP, RPKI Repository Delta Protocol) for RPs to keep their local caches in sync with the repository system [I-D.ietf-sidr-delta-protocol]. This new protocol is based on notification, snapshot and delta files. When RPs query a repository for updates, they will use delta files (and snapshot files as needed) to keep their local caches updated. Moreover, RRDP protocol can work with the existing rsync URIs.

Compared with rsync protocol, RRDP is considered to be effective to eliminate a number of consistency related issues, help to reduce the load on publication servers, and have higher scalability.

2) Improved Rsync Protocol

CNNIC also proposed an improved rsync mechanism which transfers the work of checksums calculation to RPs in order to reduce the computation load on the rsync server side. The mechanism also offered a NOTIFY method that send NOTIFY message to make some important RPs to actively fetch the updated RPKI objects in time.

4.4. Solutions to Incomplete Deployment and Low Validation Accuracy

Both of the two problems (incomplete deployment and low validation accuracy) are caused by the partial deployment of RPKI. With the widely deployment of RPKI in the near future, these two problems ought to be mitigated.

5. Security Considerations

TBD

6. IANA Considerations

This draft does not request any IANA action.

7. Acknowledgements

The authors would like to thanks the valuable comments made by Stephen Kent and other members of sidr WG.

This document was produced using the xml2rfc tool [RFC2629].

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>>.
- [RFC6480] Lepinski, M. and S. Kent, "An Infrastructure to Support Secure Internet Routing", RFC 6480, DOI 10.17487/RFC6480, February 2012, <<http://www.rfc-editor.org/info/rfc6480>>.
- [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>>.
- [RFC6483] Huston, G. and G. Michaelson, "Validation of Route Origination Using the Resource Certificate Public Key Infrastructure (PKI) and Route Origin Authorizations (ROAs)", RFC 6483, DOI 10.17487/RFC6483, February 2012, <<http://www.rfc-editor.org/info/rfc6483>>.
- [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>>.

8.2. Informative References

- [I-D.ietf-sidr-delta-protocol] Bruijnzeels, T., Muravskiy, O., Weber, B., Austein, R., and D. Mandelberg, "RPKI Repository Delta Protocol", draft-ietf-sidr-delta-protocol-01 (work in progress), October 2015.
- [RFC2629] Rose, M., "Writing I-Ds and RFCs using XML", RFC 2629, DOI 10.17487/RFC2629, June 1999, <<http://www.rfc-editor.org/info/rfc2629>>.

[RFC5914] Housley, R., Ashmore, S., and C. Wallace, "Trust Anchor Format", RFC 5914, DOI 10.17487/RFC5914, June 2010, <<http://www.rfc-editor.org/info/rfc5914>>.

Authors' Addresses

Xiaodong Lee
CNNIC
No.4 South 4th Street, Zhongguancun
Beijing, 100190
P.R. China

Email: xl@cnnic.cn

Xiaowei Liu
CNNIC
No.4 South 4th Street, Zhongguancun
Beijing, 100190
P.R. China

Email: liuxiaowei@cnnic.cn

Zhiwei Yan
CNNIC
No.4 South 4th Street, Zhongguancun
Beijing, 100190
P.R. China

Email: yanzhiwei@cnnic.cn

Guanggang Geng
CNNIC
No.4 South 4th Street, Zhongguancun
Beijing, 100190
P.R. China

Email: gengguanggang@cnnic.cn

Yu Fu
CNNIC
No.4 South 4th Street, Zhongguancun
Beijing, 100190
P.R. China

Email: fuyu@cnnic.cn

Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: September 22, 2016

A. Newton, Ed.
ARIN
C. Martinez-Cagnazzo, Ed.
LACNIC
D. Shaw
AfrinIC
T. Bruijnzeels
RIPE NCC
B. Ellacott
APNIC
March 21, 2016

RPKI Multiple "All Resources" Trust Anchors Applicability Statement
draft-rir-rpki-allres-ta-app-statement-00

Abstract

This document provides an applicability statement for the use of multiple, overclaiming 'all resources' (0/0) RPKI root certificates operated by the Regional Internet Registry community to help mitigate the risk of massive downstream invalidation in the case of transient registry inconsistencies.

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 September 22, 2016.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

(<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Requirements Language	2
2. Introduction	2
3. Applicability to reduce overclaiming possibilities	3
4. Normative References	4
Authors' Addresses	4

1. Requirements Language

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 RFC 2119 [RFC2119].

2. Introduction

RPKI is a hierarchical cryptologic system that uses certificates to match and validate holdship of IP addresses as it moves down the allocation chain from IANA to a RIR to ISPs and ending up at end users who use the address block. Since these allocations can be cryptographically validated, one could then tie assertions made by the holder of that IP address space. As an improvement of this system, RPKI was improved to add origin routing announcements via ROAs to this system. These ROAs then can be independently validated via cryptologic means by third parties to assure themselves that the origin of the announcement can be checked via what is in the actual routing system.

Since this system is envisioned to be used by ISPs to determine their routing decisions, there is a goal to be 100% correct 100% of the time. This goal could be achieved if it was to be contained in a static environment where there is little to no movement of holdship changes from one organization to another for IP addresses. Unfortunately, this state can not be achieved as there is now movement of IP addresses from organization to organization based on IPv4 runout.

Unfortunately, this state is infeasible in a model where separate entities are operating independently, yet rely critically on each others' perfect synchronisation at all times.

Because the current validation mechanism is all-or-nothing, any inconsistency at all at a high apex CA invalidates a large number of additional INRs. The higher the apex, and the larger the total set of INRs maintained by the CA, the greater the impact of even a small inconsistency.

As resources change at high apex CAs for a variety of reasons, the likelihood of a small inconsistency is non-zero, and the likelihood of a transitional inconsistency is moderate. Due to the distributed nature of the RPKI repository mechanism, even if all CAs were able to operate in perfect synchronicity, there is a reasonable likelihood that a given client may witness a temporarily inconsistent state of the system as a whole. A risk of wide-spread invalidity therefore exists as a very high impact and moderate likelihood event.

This brittleness in the RPKI validation rules has been identified and presented by the current RPKI TA operators to the IETF. A solution has also been proposed (insert ref to validation reconsidered), a solution that would allow for accidental over-claiming only to invalidate the resource that is incorrectly listed and allow the remaining to continue to be valid. This draft has had little forward progress in the IETF to date.

3. Applicability to reduce overclaiming possibilities

The consequences to a RIR over-claiming are grave given that every ISP within their certificate would be invalidated. If routing was to be reliant on RPKI at this point, all routes by those ISPs by the affected RIR certificate would no longer work.

To mitigate risk and alleviate this threat, each RIR will move from a Trust Anchor that reflects their current holdings to one that reflects all holdings (e.g. 0/0) so that over-claiming can not occur at a RIR level when dealing with transfers from one RIR to another. RPKI validators will not see the five Trust anchors from the RIRs as over-claiming and validation can proceed normally.

For those who want to audit the RIRs to ensure that RIRs are not allocating the same IP addresses in separate regions, they can audit the RIRs by matching the inventories of each RIR (**extended stats reference here) that is provided on a daily basis to the certificates issued by the RIRs within RPKI. Note that there will be a minor change from time to time to account for movements from IP address holdings that is in flight from one RIR to another.

4. 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>>.

Authors' Addresses

Andrew Newton (editor)
ARIN
Chantilly VA
United States

Email: andy@arin.net

Carlos Martinez-Cagnazzo (editor)
LACNIC
Montevideo
Uruguay

Email: carlos@lacnic.net

Daniel Shaw
AfrinIC
Cybercity Ebene
Republic of Mauritius

Email: daniel@afrrinic.net

Tim Bruijnzeels
RIPE NCC
Amsterdam
Netherlands

Email: tim@ripe.net

Byron Ellacott
APNIC
Brisbane
Australia

Email: bje@apnic.net