Network Working Group
Internet-Draft
Updates: 6480,6481,7730 (if approved)
Intended status: Standards Track
Expires: July 20, 2017

T. Bruijnzeels
O. Muravskiy
RIPE NCC
B. Weber
Cobenian
R. Austein
Dragon Research Labs
January 16, 2017

RPKI Repository Delta Protocol
draft-ietf-sidr-delta-protocol-05

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 protocol which provides relying parties with a mechanism to query a repository for incremental updates using the HTTP Over TLS (HTTPS) [RFC2818] protocol, thus enabling the RP to keep its state in sync with the repository using a secure transport channel. This document updates [RFC6480], [RFC6481], and [RFC7730].

Status of This Memo

Copyright Notice

Table of Contents

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. We call this the RPKI Repository Delta Protocol, or RRDP in short.

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 ([RFC4122]), 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 records 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 performs a full resynchronisation instead.

As soon as the RP fetches new content in this way it could 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 could use caching infrastructure to reduce its load, particularly because 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: Section 7,

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

The accessLocation MUST be an HTTPS URI as defined in [RFC2818], that will point to the update notification file for the repository server that publishes the products of this CA certificate.

3.3. Repository Server Use

3.3.1. Initialisation

When the repository server initialises it performs 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 MUST generate new snapshot and delta files within one minute. 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 ensure that publication never postponed for longer than one minute for any of the CAs involved.

Updates are processed as follows:

- The new repository serial number MUST be one greater than the current repository serial number.

- 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.

- 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.

- The format and caching concerns for delta files are explained in more detail in Section 3.5.3.

- 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.

- 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.

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

- 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 to avoid that RPs download more data than necessary.

- 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.

- 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 use this protocol as follows.

The RP MUST download the update notification file, unless an update notification file was already downloaded and processed from the same location in this validation run, or because a polling strategy was used (see Section 3.4.4).

It is RECOMMENDED that RP uses a "User-Agent" header explained in section 5.5.3. of [RFC7231] to identify the name and version of the RP software used. It is useful to 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 and RRDP cannot be used. See Section 3.4.5 for considerations.

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 3.4.2, 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 3.4.3.

3.4.2. 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 performs 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. The RP could use additional strategies to determine if an object is still relevant for validation before removing it from its local storage. In particular objects should not be removed if they are included in a current validated manifest.

If any delta file is rejected RPs MUST process the current Snapshot File instead, as described in Section 3.4.3.

3.4.3. Processing a Snapshot File

Snapshot Files MUST only be used if Delta Files are unavailable, or were rejected. As is ensured, if the process described in Section 3.4.1 is followed.

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 performs 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.

If a Snapshot File is rejected that means that RRDP cannot be used. See Section 3.4.5 for considerations.

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 RPKI object validation process. However, a detailed description of the RPKI object validation process itself is out of scope of this document.

3.4.5. Considerations Regarding Operational Failures in RRDP

If an RP experiences any issues with retrieving or processing any of the files used in this protocol, it will be unable to retrieve new RPKI data from the affected publication server.

Relying Parties could attempt to use alternative repository access mechanisms, if they are available, according to the accessMethod element value(s) specified in the SIA of the associated certificate (see Section 4.8.8 of [RFC6487]).

Furthermore Relying Parties may wish to employ re-try strategies in case of network issues. Relying Parties are also advised to keep old objects in their local cache so that validation can be done using old objects.

It is also recommendable that re-validation and retrieval is performed pro-actively before manifests or CRLs go stale, or certificates expire, to ensure that problems on the side of the RP can be identified and resolved before they cause major concerns.

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 SHOULD 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, than 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:

- 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.

- The XML namespace MUST be http://www.ripe.net/rpki/rrdp

- The encoding MUST be US-ASCII

- The version attribute in the notification root element MUST be 1

- The session_id attribute MUST be a random version 4 UUID ([RFC4122]), unique to this session

- The serial attribute MUST be an unbounded, unsigned positive integer in decimal format indicating the current version of the repository.

- The notification file MUST contain exactly one 'snapshot' element for the current repository version.

- 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. Note that the elements may not be ordered.

- 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">
    ZXhhbXBsZTE=
  </publish>
  <publish uri="rsync://rpki.ripe.net/Alice/Alice.mft">
    ZXhhbXBsZTI=
  </publish>
  <publish uri="rsync://rpki.ripe.net/Alice/Alice.crl">
    ZXhhbXBsZTM=
  </publish>
</snapshot>
```

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

- 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.

- The XML namespace MUST be http://www.ripe.net/rpki/rrdp.

- The encoding MUST be US-ASCII.

- The version attribute in the notification root element MUST be 1

- The session_id attribute MUST match the expected session_id in the reference in the notification file.

- The serial attribute MUST match the expected serial in the reference in the notification file.

- Note that the publish element is similar to the publish element defined in the publication protocol [I-D.ietf-sidr-publication]. However, the "tag" attribute is not used here because it is not relevant to relying parties. The "hash" attribute is not used here because this file represents a complete current state of the repository, and therefore it is not relevant to know which existing RPKI object (if any) is updated.

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 (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">
    ZXhhbXBsZTQ=
  </publish>
  <publish uri="rsync://rpki.ripe.net/repo/Alice/Alice.crl"
           hash="5fb1...6a56">
    ZXhhbXBsZTU=
  </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:

- 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.

- The XML namespace MUST be http://www.ripe.net/rpki/rrdp.

- The encoding MUST be US-ASCII.

- The version attribute in the delta root element MUST be 1

- The session_id attribute MUST be a random version 4 UUID unique to this session

- 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 similar to the publish element defined in the publication protocol [I-D.ietf-sidr-publication]. However, the "tag" attribute is not used here because it is not relevant to relying parties.

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. Updates

This section provides updates to several paragraphs in [RFC6480], [RFC6481], and [RFC7730]. For clarity, the original text and the replacement text are shown.

4.1. Updates to RFC6480

4.1.1. Update in Section 4.3, Access Protocols

OLD:

> To ensure all relying parties are able to acquire all RPKI signed objects, all publication points MUST be accessible via rsync (see [RFC5781] and [RSYNC]), although other download protocols MAY also

be supported. A repository publication point may provide update/change/delete functionality via (set of) access protocols that it desires, provided that the supported protocols are clearly communicated to all certification authorities publishing data at a given publication point.

NEW:

To ensure all relying parties are able to acquire all RPKI signed objects, all publication points MUST be accessible using retrieval mechanism(s) consistent with the accessMethod element value(s). Multiple retrieval mechanisms MAY be supported at the repository operator's discretion. A repository publication point may provide update/change/delete functionality via (set of) access protocols that it desires, provided that the supported protocols are clearly communicated to all certification authorities publishing data at a given publication point.

4.1.2. Update in Section 11.1, Normative References

Remove the reference to RFC5781, "The rsync URI Scheme".

4.1.3. Update in Section 11.2, Informative References

Remove the reference to rsync, "rsync web pages".

4.2. Updates to RFC6481

4.2.1. Update in Section 3, Resource Certificate Publication Repository Considerations

OLD:

The publication repository MUST be available using rsync [RFC5781] [RSYNC]. Support of additional retrieval mechanisms is the choice of the repository operator. The supported retrieval mechanisms MUST be consistent with the accessMethod element value(s) specified in the SIA of the associated CA or EE certificate.

NEW:

The publication repository MUST be available using retrieval mechanism(s) consistent with the accessMethod element value(s) specified in the SIA of the associated CA or EE certificate. Support of multiple retrieval mechanisms is the choice of the repository operator.

4.2.2. Update in Section 9.1, Normative References

Remove the reference to RFC5781, "The rsync URI Scheme".

4.2.3. Update in Section 9.2, Informative References

Remove the reference to rsync, "rsync web pages".

4.3. Updates to RFC7730

4.3.1. Update in Section 2.1, Trust Anchor Locator Format

OLD:

> where the URI section is comprised of one of more of the ordered sequence of:
>
> 1.1) an rsync URI [RFC5781],
>
> 1.2) a <CRLF> or <LF> line break.

NEW:

> where the URI section is comprised of one of more of the ordered sequence of:
>
> 1.1) a URI [RFC3986],
>
> 1.2) a <CRLF> or <LF> line break.

4.3.2. Update in Section 2.2, TAL and Trust Anchor Certificate Considerations

OLD:

> Each rsync URI in the TAL MUST reference a single object. It MUST NOT reference a directory or any other form of collection of objects.
>
> ...
>
> Where the TAL contains two or more rsync URIs, then the same self-signed CA certificate MUST be found at each referenced location.

NEW:

> Each URI in the TAL MUST reference a single object. It MUST NOT reference a directory or any other form of collection of objects.

```
      ...
```

Where the TAL contains two or more URIs, then the same self-signed CA certificate MUST be found at each referenced location.

4.3.3. Update in Section 5.1, Normative References

Remove the reference to RFC5781, "The rsync URI Scheme".

Add a reference to RFC3986, "Uniform Resource Identifier (URI): Generic Syntax".

5. Operational Considerations

5.1. Compatibility with previous standards

This protocol has been designed to replace rsync as a distribution mechanism of an RPKI repository. However, it is also designed to co-exist with existing implementations based on rsync, to enable smooth transition from one distribution mechanism to another.

For every repository object listed in the snapshot and delta files both the hash of the object's content and the rsync URI [RFC5781] of its location in the repository are listed. This makes it possible to distribute the same RPKI repository, represented by a set of files on a filesystem, using both rsync and RRDP. It also enables Relying Parties tools to query, combine, and consequently validate objects from repositories of different types. (For an example of such implementation see [I-D.ietf-sidrops-rpki-tree-validation].)

5.2. Distribution considerations

One of the design goals of RRDP was to minimise load on a repository server while serving clients. To achieve this, neither the content, nor the URLs of the snapshot and delta files are modified after they have been published in the notification file. This allows their effective distribution, either by a single HTTP server, or using a Content Distribution Network (CDN).

The RECOMMENDED way for RPs to keep up with the repository updates is to poll the Update Notification File for changes. The content of that file is updated with every new serial version of a repository (while its URL remains stable). To effectively implement distribution of the notification file, an "If-Modified-Since" HTTP request header is required to be present in all requests for notification file (see Section 3.4.4.) Therefore it is RECOMMENDED that RP tools implement a mechanism to keep track of a previous successful fetch of a notification file.

Implementations of RRDP should also take care of not producing new versions of the repository (and subsequently, new Notification, Snapshot and Delta files) too often. Usually the maintenance of the RPKI repository includes regular updates of manifest and CRL objects, performed on a schedule. This often results in bursts of repository updates during a short period of time. Since the RPs are required to poll for the Update Notification File not more often than once per minute (Section 3.4.4), it is not practical to generate new serial versions of the repository much more often than 1 per minute. It is allowed to combine multiple updates, possibly from different CAs, into a new serial repository version (Section 3.3.2). This will significantly shorten the size of the Update Notification File and total amount of data distributed to all RPs.

5.3. HTTPS considerations

It is RECOMMENDED that Relying Parties and Publication Servers follow the Best Current Practices outlined in [RFC7525] on the use of HTTP over TLS (HTTPS) [RFC2818].

Note that a Man-in-the-Middle (MITM) cannot produce validly signed RPKI data, but they can perform withhold or replay attacks targeting an RP, and keep the RP from learning about changes in the RPKI. Because of this RPs SHOULD do TLS certificate and host name validation when they fetch from an RRDP Publication Server.

RP tools SHOULD log any TLS certificate or host name validation issues they find, so that an operator can investigate the cause. However, such validation issues are often due to configuration errors, or a lack of a common TLS trust anchor. In these cases it is better if the RP retrieves the signed RPKI data regardless, and performs validation on it. Therefore RP MUST continue to retrieve the data in case of errors. The RP MAY choose to log encountered issues only when fetching the notification update file, but not when it subsequently fetches snapshot or delta files from the same host. Furthermore the RP MAY provide a way for operators to accept untrusted connections for a given host, after the cause has been identified.

6. Security Considerations

RRDP deals exclusively with transfer of RPKI objects from a repository server to a relying party. The trust relation between a CA and its repository server is out of scope for this document. However, it should be noted that from a relying party point of view all RPKI objects (certificates, CRLs, and CMS-wrapped objects) are already covered by object security mechanisms including signed manifests. This allows validation of these objects even though the

repository server itself is not trusted. This document makes no change to RPKI validation procedures per se.

The original RPKI transport protocol is rsync, which offers no channel security mechanism. RRDP replaces the use of rsync by HTTPS; while the channel security mechanism underlying RRDP (HTTPS) is not a cure-all, it does make some forms of denial of service attack more difficult for the attacker. HTTPS issues are discussed in more detail in Section 5.3.

Supporting both RRDP and rsync necessarily increases the number of opportunities for a malicious RPKI CA to perform denial of service attacks on relying parties, by expanding the number of URIs which the RP may need to contact in order to complete a validation run. However, other than the relative cost of HTTPS versus rsync, adding RRDP to the mix does not change this picture significantly: with either RRDP or rsync a malicious CA can supply an effectively infinite series of URIs for the RP to follow. The only real solution to this is for the RP to apply some kind of bound to the amount of work it is willing to do. Note also that the attacker in this scenario must be an RPKI CA, since otherwise the normal RPKI object security checks would reject the malicious URIs.

Processing costs for objects retrieved using RRDP may be somewhat different from the same objects retrieved using rsync: because RRDP treats an entire set of changes as a unit (one "delta"), it may not be practical to start processing any of the objects in the delta until the entire delta has been received. With rsync, by contrast, incremental processing may be easy, but the overall cost of transfer may be higher, as may be the number of corner cases in which the RP retrieves some but not all of the updated objects. Overall, RRDP's behavior is closer to a proper transactional system, which (probably) leads to an overall reliability increase.

RRDP is designed to scale much better than rsync. In particular, RRDP is designed to allow use of HTTPS caching infrastructure to reduce load on primary publication servers and increase resilience against denial of service attacks on the RPKI publication service.

7. IANA Considerations

IANA is requested to update the reference for id-ad-rpkiNotify to this document in the PKIX Access Descriptor registry [IANA-AD-NUMBERS].

8. Acknowledgements

The authors would like to thank David Mandelberg for reviewing this document.

9. References

9.1. 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-10 (work in progress), January 2017.

[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>.

[RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <http://www.rfc-editor.org/info/rfc2818>.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <http://www.rfc-editor.org/info/rfc4122>.

[RFC5781] Weiler, S., Ward, D., and R. Housley, "The rsync URI Scheme", RFC 5781, DOI 10.17487/RFC5781, February 2010, <http://www.rfc-editor.org/info/rfc5781>.

[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>.

[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>.

[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>.

[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>.

[RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <http://www.rfc-editor.org/info/rfc7525>.

[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>.

9.2. Informative References

[I-D.ietf-sidrops-rpki-tree-validation] Muravskiy, O. and T. Bruijnzeels, "RPKI Certificate Tree Validation by the RIPE NCC RPKI Validator", draft-ietf-sidrops-rpki-tree-validation-00 (work in progress), January 2017.

[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>.

[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>.

[rsync] "Rsync home page", <https://rsync.samba.org>.

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@hactrn.net