Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 21, 2016

T. Bruijnzeels
O. Muravskiy
RIPE NCC
B. Weber
Cobenian
R. Austein
Dragon Research Labs
D. Mandelberg
BBN Technologies
October 19, 2015

RPKI Repository Delta Protocol
draft-ietf-sidr-delta-protocol-01

Abstract

In the Resource Public Key Infrastructure (RPKI), certificate authorities publish certificates, including end entity certificates, and CRLs to repositories on repository servers. Relying Parties (RP) retrieve the published information from the repository and MAY store it in a cache. This document specifies a delta protocol which provides relying parties with a mechanism to query a repository for changes, thus enabling the RP to keep its state in sync with the repository.

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 rsync, allowing Relying Parties (RPs) to synchronise a local copy of the RPKI repository used for validation with the central repositories using the rsync protocol [RFC6481].

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

This protocol is designed to be consistent 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 a 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 servers, 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 using its local repository. 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. Update Notification File

3.2.1. 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.

3.2.2. 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.

3.2.3. 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://rpki.ripe.net/rrdp/9d--8/3/snapshot.xml"/>
  <delta serial="3" uri="https://rpki.ripe.net/rrdp/9d--8/3/delta.xml"/>
  <delta serial="2" uri="https://rpki.ripe.net/rrdp/9d--8/2/delta.xml"/>
</notification>
```

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

- o A RP MUST NOT process any update notification file that is not well formed, or which does not conform to the RELAX NG schema outlined in Section 5 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

- The serial attribute must be an unbounded, unsigned positive integer 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.

3.2.4. Repository Server Initialisation

When the repository server (re-) initialises it MUST generate a new random version 4 UUID to be used as the session_id. Furthermore it 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. This snapshot file MUST be made available at a URL that is unique to this session and version, so that it can be cached indefinitely. The format and caching concerns for snapshot files are explained in more detail below in Section 3.3. 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.

3.2.5. Publishing Updates

Whenever the repository server receives updates from a CA it SHOULD generate new snapshot and delta files and publish a new notification file as follows:

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

- A new delta file MUST be generated for this new serial. This delta file MUST include all new, replaced and withdrawn objects, as a single change set.

- This delta file MUST be made available at a URL that is unique to this session and version, so that it can be cached indefinitely.

- The repository server MUST also generate a new snaphost 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 version, so that it can be cached indefinitely.

- A new notification file MUST be created by the repository server. This new notification file MUST include a reference to the new snapshot file. The file SHOULD also include available delta files for this and previous updates. However, the server MUST NOT include more delta files than, when combined, exceed the size of the current snapshot.

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 3.2.4.

3.3. Snapshot File

3.3.1. 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.

3.3.2. 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.

Repository servers MAY delete old snapshot files one hour after they are no longer included in the notification file.

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

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

- A RP MUST NOT process any snapshot file that is not well formed, or which does not conform to the RELAX NG schema outlined in Section 5 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 defined in the publication protocol [I-D.ietf-sidr-publication]

3.4. Delta File

3.4.1. 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.

3.4.2. Cache Concerns

Deltas reflect a 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.

Repository servers MAY delete old delta files one hour after they are no longer included in the notification file.

3.4.3. 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:

o A RP MUST NOT process any delta file that is not well formed, or which does not conform to the RELAX NG schema outlined in Section 5 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.

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

- Note that the publish and withdraw elements are defined in the publication protocol [I-D.ietf-sidr-publication]

3.5. SIA for CA certificates

Certificate Authorities that use this delta protocol MUST have an instance of an SIA AccessDescription 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.

4. Relying Party Use

4.1. Full Synchronisation

When a Relying Party first encounters a notification file URI as an SIA of a certificate that it has validated it SHOULD retrieve the notification file and download the latest snapshot to get in sync with the current version of the repository server.

4.2. Processing Deltas

It is RECOMMENDED that the RP notes the URI, session_id and serial number when it first learns about a notification file. The RP MAY then poll the file to discover updates, but SHOULD NOT poll more frequently than once per minute.

If the RP finds that the session_id has changed, or if it cannot find a contiguous chain of links to delta files from its current serial to repository server's current serial, then it MUST perform a full synchronisation instead of continuing to process deltas.

If the RP finds a contiguous chain of links to delta files from its current serial to the repository server's current serial, and the session_id has not changed, it should download all missing delta files. If any delta file cannot be downloaded, or if no such chain of deltas is available, or the session_id has changed, then the RP MUST perform a full synchronisation instead.

New objects found in delta files can be added to the RPs local copy of the repository. However, it is RECOMMENDED that the RP treats object updates and withdraws with some skepticism. A compromised repository server may not have access to the certification authorities' keys, but it can pretend valid objects have been withdrawn. Therefore it may be preferred to use a strategy where local copies of objects are only discarded when the RP is sure that they are no longer relevant, e.g. the CA has explicitly removed the objects in a recent valid manifest and revoked them in a recent valid CRL (unless they have expired).

5. 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:
```

6. Security Considerations

TBD

7. IANA Considerations

This document has no actions for IANA.

8. Acknowledgements

TBD

9. 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>.

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

David Mandelberg
BBN Technologies

Email: david@mandelberg.org