Abstract

A Trust Anchor Locator (TAL) is used by Relying Parties (RPs) in the Resource Public Key Infrastructure (RPKI) to locate and validate a Trust Anchor (TA) Certification Authority (CA) certificate used in RPKI validation. This document defines an RPKI signed object for a Trust Anchor Key (TAK), that can be used by a TA to signal the location(s) of the accompanying CA certificate for the current key to RPs, as well as the successor key and the location(s) of its CA certificate. This object helps to support planned key rolls without impacting RPKI validation.

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 https://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 11 January 2023.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.
1. Requirements Notation

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Overview

A TAL [RFC8630] is used by an RP in the RPKI to locate and validate TA CA certificates used in RPKI validation. However, until now there has been no in-band way of notifying RPs of updates to a TAL. In-band notification means that TAs can be more confident of RPs being aware of key roll operations.

This document defines a new RPKI signed object that can be used to document the location(s) of the TA CA certificate for the current TA key, as well as the value of the successor key and the location(s) of its TA CA certificate. This allows RPs to be notified automatically of such changes, and enables TAs to stage a successor key so that planned key rolls can be performed without risking the invalidation of the RPKI tree under the TA. We call this object the Trust Anchor Key (TAK) object.

When RPs are first bootstrapped, they use a TAL to discover the key and location(s) of the CA certificate for a TA. The RP can then retrieve and validate the CA certificate, and subsequently validate the manifest [RFC6486] and CRL published by that TA (section 5 of [RFC6487]). However, before processing any other objects it will first validate the TAK object, if present. If the TAK object lists only the current key, then the RP continues processing as per normal. If the TAK object includes a successor key, the RP starts an acceptance timer, and then continues processing as per normal. If, during the following validation runs up until the expiry of the acceptance timer, the RP has not observed any changes to the keys and certificate URLs listed in the TAK object, then the RP will fetch the successor key, update its local state with that key and its associated certification location(s), and continue processing using that key.

The primary motivation for this work is being able to migrate from a Hardware Security Module (HSM) produced by one vendor to one produced by another, where the first vendor does not support exporting keys for use by the second. There may be other scenarios in which key rollover is useful, though.
3. TAK Object Definition

The TAK object makes use of the template for RPKI digitally signed objects [RFC6488], which defines a Cryptographic Message Syntax (CMS) [RFC5652] wrapper for the content as well as a generic validation procedure for RPKI signed objects. Therefore, to complete the specification of the TAK object (see Section 4 of [RFC6488]), this document defines:

* The OID (in Section 3.1) that identifies the signed object as being a TAK. (This OID appears within the eContentType in the encapsContentInfo object, as well as the content-type signed attribute in the signerInfo object.)

* The ASN.1 syntax for the TAK eContent, in Section 3.2.

* The additional steps required to validate a TAK, in Section 3.3.

3.1. The TAK Object Content Type

This document requests an OID for the TAK object as follows:

```
  id-ct-signed-Tal OBJECT IDENTIFIER ::= 
    { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1) pkcs9(9) 
      smime(16) id-smime ct(1) TBD }
```

This OID MUST appear both within the eContentType in the encapsContentInfo object, as well as the content-type signed attribute in the signerInfo object (see [RFC6488]).

3.2. The TAK Object eContent

The content of a TAK object is ASN.1 encoded using the Distinguished Encoding Rules (DER) [X.690], and is defined per the module in Appendix A.

3.2.1. TAKey

This structure defines a TA key, similarly to [RFC8630]. It contains a sequence of one or more URIs and a SubjectPublicKeyInfo.

3.2.1.1. certificateURIs

This field is equivalent to the URI section defined in section 2.2 of [RFC8630]. It MUST contain at least one CertificateURI element. Each CertificateURI element contains the IA5String representation of either an rsync URI [RFC5781], or an HTTPS URI [RFC7230].
3.2.1.2. subjectPublicKeyInfo

This field contains a SubjectPublicKeyInfo (section 4.1.2.7 of [RFC5280]) in DER format [X.690].

3.2.2. TAK

3.2.2.1. version

The version number of the TAK object MUST be 0.

3.2.2.2. current

This field contains the TA key of the repository in which the TAK object is published.

3.2.2.3. predecessor

This field contains the TA key that was in use for this TA immediately prior to the current TA key, if applicable.

3.2.2.4. successor

This field contains the TA key to be used in place of the current key, after expiry of the relevant acceptance timer.

3.3. TAK Object Validation

To determine whether a TAK object is valid, the RP MUST perform the following checks in addition to those specified in [RFC6488]:

* The eContentType OID matches the OID described in Section 3.1.

* The TAK object appears as the product of a TA CA certificate (i.e. the TA CA certificate is itself the issuer of the EE certificate of the TAK object).

* The TA CA has published only one TAK object in its repository for this key, and this object appears on the manifest as the only entry using the ".tak" extension (see [RFC6481]).

* The EE certificate of this TAK object describes its Internet Number Resources (INRs) using the "inherit" attribute.

* The decoded TAK content conforms to the format defined in Section 3.2.
* The SubjectPublicKeyInfo value of the current TA key in the TAK object matches that of the TA CA certificate used to issue the EE certificate of the TAK object.

If any of these checks does not succeed, the RP MUST ignore the TAK object, and proceed as though it were not listed on the manifest.

The RP is not required to compare its current set of certificateURIs for the current key with those listed in the TAK object. The RP MAY alert the user that these sets of certificateURIs do not match, with a view to the user manually updating the set of certificateURIs in their configuration. The RP MUST NOT automatically update its configuration to use these certificateURIs in the event of inconsistency, though, because migration of users to new certificateURIs should happen by way of the successor key process.

4. TAK Object Generation and Publication

A TA MAY choose to use TAK objects to communicate its current, predecessor, and successor keys. If a TA chooses to use TAK objects, then it SHOULD generate and publish TAK objects under each of its keys.

A non-normative guideline for naming this object is that the filename chosen for the TAK object in the publication repository be a value derived from the public key part of the entity’s key pair, using the algorithm described for CRLs in section 2.2 of [RFC6481] for generation of filenames. The filename extension of ".tak" MUST be used to denote the object as a TAK.

In order to generate a TAK object, the TA MUST perform the following actions:

* The TA MUST generate a key pair for a "one-time-use" EE certificate to use for the TAK.

* The TA MUST generate a one-time-use EE certificate for the TAK.

* This EE certificate MUST have an SIA extension access description field with an accessMethod OID value of id-ad-signedObject, where the associated accessLocation references the publication point of the TAK as an object URL.
As described in [RFC6487], an [RFC3779] extension is required in the EE certificate used for this object. However, because the resource set is irrelevant to this object type, this certificate MUST describe its Internet Number Resources (INRs) using the "inherit" attribute, rather than explicit description of a resource set.

This EE certificate MUST have a "notBefore" time that matches or predates the moment that the TAK will be published.

This EE certificate MUST have a "notAfter" time that reflects the intended duration for which this TAK will be published. If the EE certificate for a TAK object is expired, it MUST no longer be published, but it MAY be replaced by a newly generated TAK object with equivalent content and an updated "notAfter" time.

The current TA key for the TAK MUST match that of the TA CA certificate under which the TAK was issued.

5. Relying Party Use

Relying Parties MUST keep a record of the current key for each configured TA, as well as the URI(s) where the CA certificate for this key may be retrieved. This record is typically bootstrapped by the use of a pre-configured (and unsigned) TAL file [RFC8630].

When performing top-down validation, RPs MUST first validate and process the TAK object for its current known key, by performing the following steps:

A CA certificate is retrieved and validated from the known URIs as described in sections 3 and 4 of [RFC8630].

The manifest and CRL for this certificate are then validated as described in [RFC6487] and [RFC6486].

The TAK object, if present, is validated as described in Section 3.3.

If the TAK object includes a successor key, then the RP must verify the successor key by doing the following:

performing top-down validation using the successor key, in order to validate the TAK object for the successor TA;

ensuring that a valid TAK object exists for the successor TA;
enabling that the successor TAK object’s current key matches the initial TAK object’s successor key; and

* ensuring that the successor TAK object’s predecessor key matches the initial TAK object’s current key.

If any of these steps fails, then the successor key has failed verification.

If the successor key passes verification, and the RP has not seen that successor key on the previous successful validation run for this TA, then the RP:

* sets an acceptance timer of 30 days for this successor key for this TA;

* cancels the existing acceptance timer for this TA (if applicable); and

* continues standard top-down validation as described in [RFC6487] using the current key.

If the successor key passes verification, and the RP has seen that successor key on the previous successful validation run for this TA:

* if the relevant acceptance timer has not expired, the RP continues standard top-down validation using the current key;

* otherwise, the RP updates its current known key details for this TA to be those of the successor key, and then begins top-down validation again using the successor key.

If the successor key does not pass verification, or if the TAK object does not include a successor key, the RP cancels the existing acceptance timer for this TA (if applicable).

An RP MUST NOT use a successor key for top-down validation outside of the process described above, except for the purpose of testing that the new key is working correctly. This allows a TA to publish a successor key for a period of time, allowing RPs to test it, while still being able to rely on RPs using the current key for their production RPKI operations.

A successor key may have the same SubjectPublicKeyInfo value as the current key: this will be the case where a TA is updating the certificateURIs for that key.
6. Maintaining Multiple TA Keys

Although an RP that can process TAK objects will only ever use one key for validation (either the current key, or the successor key, once the relevant acceptance timer has expired), an RP that cannot process TAK objects will continue to use the key details per its TAL (or equivalent manual configuration) indefinitely. As a result, even when a TA is using a TAK object in order to migrate clients to a new key, the TA may have to maintain the previous key for a period of time alongside the new key in order to ensure continuity of service for older clients.

For each TA key that a TA is maintaining, the signed material for these keys MUST be published under different directories in the context of the ‘id-ad-caRepository’ and ‘id-ad-rpkiManifest’ Subject Information Access descriptions contained on the CA certificates [RFC6487]. Publishing objects under the same directory is potentially confusing for RPs, and could lead to object invalidity in the event of file name collisions.

Also, the CA certificates for each maintained key, and the contents published by each key, MUST be equivalent (except for the TAK object). In other words, for the purposes of RPKI validation, it MUST NOT make a difference which of the keys is used as a starting point.

This means that the IP and AS resources contained on all current CA certificates for the maintained TA keys MUST be the same. Furthermore, for any delegation of IP and AS resources to a child, the TA MUST have an equivalent CA certificate published under each of its keys. Any updates in delegations MUST be reflected under each of its keys. A TA SHOULD NOT publish any other objects besides a CRL, a Manifest, a single TAK object, and any number of CA certificates for delegation to child CAs.

If a TA uses a single remote publication server for its keys, per [RFC8181], then it MUST include all <publish/> and <withdraw/> PDUs for the products of each of its keys in a single query, in order to ensure that they will reflect the same content at all times.

If a TA uses multiple publication servers, then it is by definition inevitable that the content of different keys will be out of sync at times. In such cases, the TA SHOULD ensure that the duration of these moments are limited to the shortest possible time. Moreover, the following should be observed:
* In cases where a CA certificate is revoked completely, or replaced by a certificate with a reduced set of resources, these changes will not take effect fully until all the relevant repository publication points have been updated. Given that TA key operations are normally performed infrequently, this is unlikely to be a problem: if the revocation or shrinking of an issued CA certificate is staged for days/weeks, then experiencing a delay of several minutes for the repository publication points to be updated is fairly insignificant.

* In cases where a CA certificate is replaced by a certificate with an extended set of resources, the TA MUST inform the receiving CA only after all of its repository publication points have been updated. This ensures that the receiving CA will not issue any products that could be invalid if an RP uses a TA key just before the CA certificate was due to be updated.

Finally, note that the publication locations of CA certificates for delegations to child CAs under each key will be different, and therefore the Authority Information Access ‘id-ad-caissuers’ values (section 4.8.7 of [RFC6487]) on certificates issued by the child CAs may not be as expected when performing top-down validation, depending on the TA key that is used. However, these values are not critical to top-down validation, so RPs performing such validation MUST NOT reject a certificate simply because this value is not as expected.

7. Performing TA Key Rolls

In this section we will describe how present-day RPKI TAs that use only one key pair, and that do not use TAK objects, can use a TAK object to perform a planned key roll.

7.1. Phase 1: Add a TAK for Key ‘A’

Before adding a successor key, a TA may want to confirm that it can maintain a TAK object for its current key only. We will refer to this key as key ‘A’ throughout this section.

7.2. Phase 2: Add a Key ‘B’

The TA can now generate a new key pair for key ‘B’. This key MUST now be used to create a new CA certificate for this key, and to issue equivalent CA certificates for delegations to child CAs, as described in Section 6.

At this point, the TA can also construct a new TAL file [RFC8630] for key ‘B’, and test locally that the validation outcome for the new key is equivalent to that of the other current key(s).
When the TA is certain that both keys are equivalent, and wants to initiate the migration from 'A' to 'B', it issues a new TAK object under key 'A', with key 'A' as the current key for that object, key 'B' as the successor key, and no predecessor key. It also issues a TAK object under key 'B', with key 'B' as the current key for that object, key 'A' as the predecessor key, and no successor key. Once this has happened, RP clients will start seeing the new key and setting acceptance timers accordingly.

7.3. Phase 3: Update TAL to point to 'B'

At about the time that the TA expects clients to start setting key 'B' as the current key, the TA must release a new TAL file for key 'B'. It SHOULD use a different set of URIs in the TAL compared to the TAK file, so that the TA can learn the proportion of RPs that can successfully validate and use the updated TAK objects.

To support RPs that do not take account of TAK objects, the TA should continue operating key 'A' for a period of time after the expected migration of clients to 'B'. The length of that period of time is a local policy matter for that TA: it might operate the key until no clients are attempting to validate using it, for example.

7.4. Phase 4: Remove Key 'A'

The TA SHOULD now remove all content from the repository used by key 'A', and destroy the private key for key 'A'. RPs attempting to rely on a TAL for key 'A' from this point will not be able to perform RPKI validation for the TA, and will have to update their local state manually, by way of a new TAL file.

8. Using TAK objects to distribute TAL data

Relying Parties must be configured with RPKI Trust Anchor data in order to function correctly. This Trust Anchor data is typically distributed in the Trust Anchor Locator (TAL) format defined in RFC 8630. A TAK object can also serve as a format for distribution of this data, though, because the TAKey data stored in the TAK object contains the same data that would appear in a TAL for the associated Trust Anchor.

Relying Parties may support conversion of TAK objects into TAL files. Relying Parties that support conversion MUST validate the TAK object using the process from section 3.3. One exception to the standard validation process in this context is that a Relying Party MAY treat a TAK object as valid, even though it is associated with a Trust Anchor that the Relying Party is not currently configured to trust.
If the Relying Party is relying on this exception when converting a given TAK object, the Relying Party MUST communicate that fact to the user.

When converting a TAK object, a Relying Party MUST default to producing a TAL file based on the 'current' TAKey in the TAK object, though it MAY optionally support producing TAL files based on the 'predecessor' and 'successor' TAKeys.

If TAK object validation fails, then the Relying Party MUST NOT produce a TAL file based on the TAK object.

9. Deployment Considerations

Including TAK objects while RPs do not support this standard will result in those RPs rejecting these objects. It is not expected that this will result in the invalidation of any other object under a Trust Anchor.

The mechanism introduced here can only be relied on once a majority of RPs support it. Defining when that moment arrives is something that cannot be established at the time of writing this document. The use of unique URIs for keys in TAK objects, different from those used for the corresponding TAL files, should help TAs understand the proportion of RPs that support this mechanism.

Some RPs may purposefully not support this mechanism: for example, they may be implemented or configured such that they are unable to update local current key data. TAs should take this into consideration when planning key rollover. However, these RPs would ideally still notify their operators of planned key rollovers, so that the operator could update the relevant configuration manually.

10. Security Considerations

A TA needs to consider the length of time for which it will maintain previously-current keys and their associated repositories. An RP that is seeded with old TAL data will run for 30 days using the previous key before migrating to the next key, due to the acceptance timer requirements, and this 30-day delay applies to each new key that has been issued since the old TAL data was initially published. It may be better in these instances to have the old publication URLs simply fail to resolve, so that the RP reports an error to its operator and the operator seeds it with up-to-date TAL data immediately.
Once a TA has decided not to maintain a previously-current key and its associated repository, it needs to consider how to protect against an adversary gaining access to that key and its associated publication points in order to send invalid/incorrect data to RPs seeded with the TAL data for that key. One possible mitigation here is to reuse the TA CA certificate URLs from that TAL data for newer keys.

The use of acceptance timers means that an adversary that gains access to a TA’s current key is not able to migrate RPs to a new key without delay. Although access to that key does permit arbitrary action within the corresponding TA (assuming that the adversary has control over the relevant publication points), being unable to migrate RPs to a new key means that it is possible for the TA operator to regain control over the key and the TA itself, such that it may not be necessary for all RPs to carry out manual reconfiguration.

If an adversary gains access to the key listed as the successor to a TA’s current key (i.e. listed as the successor, but the acceptance timer period has not yet elapsed since it was listed), the TA operator can recover from this by simply removing the successor key from the TAK object.

In general, the risk of key compromise can be mitigated by the use of Hardware Security Modules (HSMs) by TAs, which will guard against theft of a private key, as well as operational processes to guard against (accidental) misuse of the keys in an HSM by operators.

Alternate models of TAL update exist and can be complementary to this mechanism. For example, TAs can liaise directly with validation software developers to include updated and reissued TAL files in new code releases, and use existing code update mechanisms in the RP community to distribute the changes.

11. IANA Considerations

11.1. OID

IANA is asked to add the following to the "RPKI Signed Objects" registry:

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Description</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD</td>
<td>Trust Anchor Key</td>
<td>[section 3.1]</td>
</tr>
</tbody>
</table>

Martinez, et al. Expires 11 January 2023
11.2. File Extension

IANA is asked to add an item for the Signed TAL file extension to the "RPKI Repository Name Scheme" created by [RFC6481] as follows:

<table>
<thead>
<tr>
<th>Extension</th>
<th>RPKI Object</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>.tak</td>
<td>Trust Anchor Key</td>
<td>[this document]</td>
</tr>
</tbody>
</table>

11.3. Module Identifier

IANA is asked to register an object identifier for one module identifier in the "SMI Security for S/MIME Module Identifier (1.2.840.113549.1.9.16.0)" registry as follows:

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Description</th>
<th>References</th>
</tr>
</thead>
<tbody>
<tr>
<td>TBD</td>
<td>Trust Anchor Key</td>
<td>[section 3.1]</td>
</tr>
</tbody>
</table>

* Description: RPKISignedTrustAnchorList-2021
* OID: 1.2.840.113549.1.9.16.0.TBD
* Specification: [this document]

12. Implementation Status

NOTE: Please remove this section and the reference to RFC 7942 prior to publication as an RFC.

This section records the status of known implementations of the protocol defined by this specification at the time of posting of this Internet-Draft, and is based on a proposal described in [RFC7942]. The description of implementations in this section is intended to assist the IETF in its decision processes in progressing drafts to RFCs. Please note that the listing of any individual implementation here does not imply endorsement by the IETF. Furthermore, no effort has been spent to verify the information presented here that was supplied by IETF contributors. This is not intended as, and must not be construed to be, a catalog of available implementations or their features. Readers are advised to note that other implementations may exist.
According to RFC 7942, "this will allow reviewers and working groups to assign due consideration to documents that have the benefit of running code, which may serve as evidence of valuable experimentation and feedback that have made the implemented protocols more mature. It is up to the individual working groups to use this information as they see fit".

12.1. APNIC

* Responsible Organization: Asia-Pacific Network Information Centre
* Location: https://github.com/APNIC-net/rpki-signed-tal-demo
* Description: A proof-of-concept for relying party TAK usage.
* Level of Maturity: This is a proof-of-concept implementation.
* Coverage: This implementation includes all of the features described in version 08 of this specification. The repository includes a link to various test TALs that can be used for testing TAK scenarios, too.
* Contact Information: Tom Harrison, tomh@apnic.net

13. Revision History

03 - Last draft under Tim’s authorship.
04 - First draft with George’s authorship. No substantive revisions.
05 - First draft with Tom’s authorship. No substantive revisions.
06 - Rob Kisteleki’s critique.
07 - Switch to two-key model.
08 - Keepalive.
09 - Acceptance timers, predecessor keys, no long-lived CRL/MFT.
10 - Using TAK objects for distribution of TAL data.
14. Acknowledgments

The authors wish to thank Martin Hoffmann for a thorough review of the document, Russ Housley for multiple reviews of the ASN.1 definitions and for providing a new module for the TAK object, and Job Snijders for the suggestion about using TAK objects for distribution of TAL data.

15. References

15.1. Normative References


15.2. Informative References


Appendix A. ASN.1 Module

This appendix includes the ASN.1 module for the TAK object.
<CODE BEGINS>
RPKISignedTrustAnchorList-2021
   { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
     pkcs9(9) smime(16) mod(0) TBD }

DEFINITIONS EXPLICIT TAGS ::= BEGIN

IMPORTS

CONTENT-TYPE
   FROM CryptographicMessageSyntax-2009 -- in [RFC5911]
   { iso(1) member-body(2) us(840) rsadsi(113549) pkcs(1)
     pkcs-9(9) smime(16) modules(0) id-mod-cms-2004-02(41) }

SubjectPublicKeyInfo
   FROM PKIX1Explicit-2009 -- in [RFC5912]
   { iso(1) identified-organization(3) dod(6) internet(1)
     security(5) mechanisms(5) pkix(7) id-mod(0)
     id-mod-pkix1-explicit-02(51) } ;

ct-signedTAL CONTENT-TYPE ::= { TYPE TAK IDENTIFIED BY
   id-ct-signedTAL }

id-ct-signedTAL OBJECT IDENTIFIER ::= { iso(1) member-body(2)
   us(840) rsadsi(113549) pkcs(1) pkcs9(9) smime(16) ct(1) TBD }

CertificateURI ::= IA5String

TAKey ::= SEQUENCE {
   certificateURIs  SEQUENCE SIZE (1..MAX) OF CertificateURI,
   subjectPublicKeyInfo  SubjectPublicKeyInfo
}

TAK ::= SEQUENCE {
   version      INTEGER DEFAULT 0,
   current      TAKey,
   predecessor  [0] TAKey OPTIONAL,
   successor    [1] TAKey OPTIONAL
}

END

<CODE ENDS>

Authors’ Addresses

Martinez, et al. Expires 11 January 2023
Source Address Validation Using BGP UPDATEs, ASPA, and ROA (BAR-SAV)
draft-sriram-sidrops-bar-sav-00

Abstract

Designing an efficient source address validation (SAV) filter requires minimizing false positives (i.e., avoiding dropping legitimate traffic) while maintaining directionality (see RFC8704). This document advances the technology for SAV filter design through a method that makes use of BGP UPDATE messages, Autonomous System Provider Authorization (ASPA), and Route Origin Authorization (ROA). The proposed method’s name is abbreviated as BAR-SAV. BAR-SAV can be used by network operators to derive more robust SAV filters and thus improve network resilience.

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 https://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 17 December 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.
1. Introduction

Spoofed source addresses are often used in Denial of Service (DoS) and Distributed DoS (DDoS) attacks. Source address validation (SAV) filtering is used to drop packets with spoofed source addresses (see BCP 84 [RFC3704] [RFC8704]). A detailed review of unicast Reverse Path Forwarding (uRPF) techniques for SAV is provided in [RFC8704]). Also, [RFC8704] describes enhanced feasible-path uRPF (EFP-uRPF) methods that aim to minimize false positives (i.e., avoid dropping legitimate traffic) while maintaining directionality (see definitions in [RFC3704]).

New technology for securing the Border Gateway Protocol (BGP) [RFC4271] using Resource Public Key Infrastructure (RPKI) [RFC6480] is seeing increasing adoption. Two of the currently existing or proposed types of signed objects in the RPKI can be leveraged for a more accurate SAV filter design as well. These are the Route Origin Authorization (ROA) and the Autonomous System Provider Authorizations (ASPA) objects. A ROA is a cryptographically signed attestation by an IP address-resource holder listing their prefixes that are authorized to be originated in BGP by a specific autonomous system (AS) [RFC6482]. ROAs are currently used for Route Origin Validation.

Table of Contents

1. Introduction .................................................. 2
1.1. Requirements Language .................................. 3
2. Same Procedure Applies to Customers and Lateral Peers ...... 3
3. SAV Using ASPA and ROA (Procedure X) .................... 4
4. SAV using BGP UPDATE Messages, ASPA, and ROA (BAR-SAV) ... 5
5. Operational Recommendations ............................... 6
  5.1. Considerations for the CDN and DSR Scenario ........ 7
6. IANA Considerations ......................................... 8
7. Security Considerations ..................................... 9
8. References .................................................. 9
  8.1. Normative References .................................. 9
  8.2. Informative References ................................. 10
Acknowledgements .............................................. 11
Authors’ Addresses ........................................... 11
(ROV) [RFC6811]. An ASPA is a cryptographically signed attestation by an AS listing its transit provider AS numbers (ASNs) [I-D.ietf-sidrops-aspa-profile]. The ASPA data is designed to be used for a form of AS path validation that can detect and mitigate route leaks [I-D.ietf-sidrops-aspa-verification] [sriram1] [sriram2]. See [RFC7908] for the definition of route leaks.

This document advances the technology for SAV filter design using methods that make use of ASPA, ROA, and/or BGP UPDATE data. A method is presented in Section 3 that makes use of only ASPA and ROA data to design the SAV filter. This method is for use in the future when the adoption of ROA and ASPA is considered to be ubiquitous. However, for use in the period before that, another method for SAV is presented in Section 4 that makes complementary use of BGP UPDATE messages along with ASPA and ROA data. Accordingly, the latter method’s name is abbreviated as BAR-SAV. It is hoped that just as the adoption of ROAs is growing at present [Monitor], the adoption of ASPA will also gain momentum in the near future. The BAR-SAV method additionally incorporates a refined version of Algorithm A of the EFP-uRPF technique (Section 3.1 of [RFC8704]). BAR-SAV can be used by network operators to derive more robust SAV filters and thus improve network resilience.

The focus of this document is on the design of ingress SAV filters for an interface facing a customer or lateral peer AS. The same procedure applies in both cases (Section 2).

The reader is encouraged to be familiar with [RFC8704], [RFC6811], and [I-D.ietf-sidrops-aspa-profile].

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Same Procedure Applies to Customers and Lateral Peers

The same procedure applies for the construction of a permissible ingress SAV filter for a customer or lateral peer interface. Customers and lateral peers should only transmit data packets with source addresses belonging to only the prefixes that are authorized to be used by the ASes in their respective customer cones (CC). The CC includes the AS belonging to the customer or lateral peer.
3. SAV Using ASPA and ROA (Procedure X)

The method/procedure (called Procedure X) described in this section is for future scenarios when ASPA and ROA adoption is ubiquitous. In that scenario, robust SAV filters can be generated from the RPKI information (ASPA and ROA data) alone. The procedure is applicable for ingress SAV filter design for customer and lateral peer interfaces. An ISP may use Procedure X on customer interfaces if it requires all its customers to register ROAs and ASPAs.

A description of Procedure X (one that makes use of only ASPA and ROA data):

* Step A: Compute the set of ASNs in the Customer’s or Lateral Peer’s customer cone using ASPA data.

* Step B: Compute from ROA data the set of unique prefixes authorized to be announced by the ASNs found in Step A. Keep only the unique prefixes. This set is the permissible prefix list for SAV for the interface in consideration.

A detailed description of Procedure X is as follows:

1. Let the Customer or Lateral Peer ASN be denoted as AS-k.

2. Let i = 1. Initialize: AS-set S(1) = {AS-k}.

3. Increment i to i+1.

4. Create AS-set S(i) of all ASNs whose ASPA data declares at least one ASN in AS-set S(i-1) as a Provider.

5. If AS-set S(i) is null, then set i_max = i - 1 and go to Step 6. Else, go to Step 3.

6. Form the union of the sets, S(i), i = 1, 2, ..., i_max, and name this union as AS-set A.

7. Select all ROAs in which the authorized origin ASN is equal to any ASN in AS-set A. Form the union of the sets of prefixes listed in the selected ROAs. Name this union set of prefixes as P-set.

8. Apply P-set as the list of permissible prefixes for SAV.
SAV using BGP UPDATE Messages, ASPA, and ROA (BAR-SAV) is described in this section and is meant for the period when there is a partial deployment of ROAs and ASPAs. To compensate for incomplete RPKI information, BAR-SAV augments ASPA data with BGP UPDATE AS_PATH data for discovering CC ASes, and it augments ROA data with BGP UPDATE data for discovering all prefixes associated with ASes in the CC. The details of this procedure are described below.

BAR-SAV additionally incorporates a refined version of Algorithm A of EFP-uRPF (Section 3.1 of [RFC8704]). Algorithm A in [RFC8704] picked only the originating ASes from AS_PATHs received on the customer or lateral peer interface in consideration and included them for SAV filter computation. The variant of Algorithm A in [RFC8704] used here includes all ASes in the AS_PATHs for the SAV filter computation. Unless there is a route leak [RFC7908], each AS is a customer of the AS added next in AS_PATHs of BGP UPDATE messages received from a customer or lateral peer. Further customer-provider AS relations within the CC are discovered by examining all unique ASes in the AS_PATHs in BGP UPDATEs received on all interfaces (from transit providers, customers, lateral peers, and IBGP peers). This is described in the step-by-step procedure later in this section.

Note that if a multi-homed AS is present in an above-mentioned AS_PATH and did not originate any prefix in the CC in consideration but originated a prefix into an overlapping neighboring CC, then the AS and prefix will still be detected and included in the design of the SAV filter. This improves the accuracy of the SAV filter in the BAR-SAV method in comparison to Algorithm A in [RFC8704].

One should not compute a customer cone by separately processing ASPA data and AS_PATH data and then merging the two sets of ASes at the end. Doing so is likely to miss ASes from the customer cone. Instead, both ASPAs and AS_PATHs should be used to iteratively expand the discovered customer cone. When new ASes are discovered, both ASPA and AS_PATH data should be used to discover customers of those ASes. This process is repeated for newly discovered customer ASes until there are no new ASes to be found.

If a transit provider-to-customer relationship, e.g., from AS X to AS Y, is deduced from AS_PATH data but the ASPA data contradicts it (i.e., AS Y has ASPA and it does not include AS X as a transit provider), then the ASPA data prevails, and AS Y must not be considered to be a customer of X. This design principle is reflected in Step 5 of the procedure described below. (Please see discussion about route leaks in Section 7.)
A detailed description of the BAR-SAV procedure is as follows:

1. Let the Customer or Lateral Peer ASN be denoted as \( AS-k \).
2. Let \( i = 1 \). Initialize: \( AS\text{-set } Z(1) = \{ AS-k \} \).
3. Increment \( i \) to \( i+1 \).
4. Create \( AS\text{-set } A(i) \) of all ASNs whose ASPA data declares at least one ASN in \( AS\text{-set } Z(i-1) \) as a Provider.
5. Create \( AS\text{-set } B(i) \) of all "non-ASPA" customer ASNs each of which is a customer of at least one ASN in \( AS\text{-set } Z(i-1) \) according to unique AS_PATHs in Adj-RIBs-In [RFC4271] of all interfaces at the BGP speaker computing the SAV filter. "Non-ASPA" ASN are ASNs that declare no provider in ASPA data.
6. Form the union of \( AS\text{-sets } A(i) \) and \( B(i) \) and call it \( AS\text{-set } C \). From \( AS\text{-set } C \), remove any ASNs that are present in \( Z(j) \), for \( j=1 \) to \( j=(i-1) \). Call the resulting set \( Z(i) \).
7. If \( AS\text{-set } Z(i) \) is null, then set \( i_{\max} = i - 1 \) and go to Step 8. Else, go to Step 3.
8. Form the union of the \( AS\text{-sets}, Z(i), i = 1, 2, \ldots, i_{\max}, \) and name this union as \( AS\text{-set } D \).
9. Select all ROAs in which the authorized origin ASN is in \( AS\text{-set } D \). Form the union of the sets of prefixes listed in the selected ROAs. Name this union set of prefixes as Prefix-set \( P_1 \).
10. Using the routes in Adj-RIBs-In of all interfaces, create a list of all prefixes originated by any ASN in \( AS\text{-set } D \). Name this set of prefixes as Prefix-set \( P_2 \).
11. Form the union of Prefix-sets \( P_1 \) and \( P_2 \). Apply this union set as the list of permissible prefixes for SAV.

5. Operational Recommendations

Network operators SHOULD implement the BAR-SAV method (Section 4) for computing the permissible ingress prefix list for SAV on interfaces facing customers and lateral peers. BAR-SAV offers immediate incremental benefits to early adopters.
The operational recommendations provided in Section 3.2 of [RFC8704] are applicable and helpful for BAR-SAV (Section 4). Since Procedure X (Section 3) and the BAR-SAV procedure (Section 4) benefit from the registration of ROAs, network operators are RECOMMENDED to register ROAs and enable ROV in their ASes. When ASPA becomes available, network operators are also RECOMMENDED to register ASPAs at that time.

The registration of ROAs and ASPAs helps with the detection and inclusion of otherwise hidden prefixes in the permissible list for SAV. As mentioned earlier, prefixes hidden in other techniques often arise from the use of multi-homing in conjunction with limited propagation of prefixes in a given CC (for example, by attaching NO_EXPORT to all prefixes announced from a customer AS to a transit provider AS). In these situations, the registration of ASPAs helps improve the accuracy of SAV.

5.1. Considerations for the CDN and DSR Scenario

Direct Server Return (DSR) is a common asymmetric routing scenario that is not supported by existing BCP-84 uRPF [RFC3704] and EFP-uRPF [RFC8704] SAV methods. DSR is commonly used by Content Delivery Networks (CDNs) that wish to use anycast service addresses but deliver data from edge locations that do not announce anycast addresses.

For example, in Figure 1, the CDN announces an anycast prefix P3 (from AS3) from a well-connected location with CDN control infrastructure. When a User from prefix P1 (AS1) establishes a connection to the anycast address and requests an object, an Anycast Server at the CDN may determine that the best location to serve the object is an Edge Server in a location close to the User. The Edge Server is reachable only via prefix P2 (AS2). The Anycast Server can forward packets arriving from the User to the Edge Server (via IP-IP tunneling or similar means), but the bulk data transmission would need to happen directly from the Edge Server to the User with an anycast source address (a P3 address).
Consider AS4 generating its SAV list
CDN’s ROAs: {P3 AS3}, {P3, AS2}, {P2, AS2}
AS2 should not/does not announce P3
With the SAV methods in this document,
AS4 correctly includes P2 and P3 in its SAV list

Figure 1: Illustration of how the solution functions for the CDN/DSR scenario.

Existing SAV methods of [RFC3704] and EFP-uRPF [RFC8704] would not allow AS4 to include P3 as a legitimate SA prefix on the interface to AS2. However, if the CDN (owner of prefix P3) registers a ROA object authorizing AS2 to originate P3, and AS4 uses an SAV procedure specified in this draft, then AS4 will use that ROA object to include P3 as a valid source prefix for the AS2 customer interface. The CDN may never want to announce a route to P3 from AS2, but the existence of this ROA would result in the construction of an SAV filter that would permit AS2 to send data packets with source addresses belonging to P3.

The CDN example above is just one DSR scenario. There are other cloud-based DSR scenarios that include low-latency gaming, mobile roaming, corporate networks of global enterprises, and others.

Recommendation: In a DSR scenario, a network operator SHOULD register ROAs authorizing edge server ASes to announce anycast service prefixes. This is in addition to registering a ROA authorizing the anycast server AS to announce the anycast prefix.

6. IANA Considerations

This document includes no request to IANA.
7. Security Considerations

The security considerations described in [RFC8704], [RFC6811], and [I-D.ietf-sidrops-aspa-profile] also apply to this document.

The security and robustness of BAR-SAV are strengthened by supporting mechanisms for detecting and dropping routes that are misoriginations or leaks. It is advised that the BGP UPDATEs received at BGP speakers are vetted using ROV (using ROAs and/or trusted IRR route objects) and prefix filtering (see [RFC6811] [RFC7454] [NIST-800-189]). It is also advised that one or more of the available methods to prevent, detect, and mitigate route leaks are also deployed (e.g., [RFC9234] [I-D.ietf-grow-route-leak-detection-mitigation] [I-D.ietf-sidrops-aspa-verification] [sriram1] [sriram2]).

8. References

8.1. Normative References

[I-D.ietf-sidrops-aspa-profile]


8.2. Informative References

[I-D.ietf-sidrops-aspa-verification]

[I-D.ietf-grow-route-leak-detection-mitigation]


[NIST-800-189]
Acknowledgements

The authors would like to thank Oliver Borchert for comments and discussion.

Authors’ Addresses

Kotikalapudi Sriram
USA National Institute of Standards and Technology
100 Bureau Drive
Gaithersburg, MD 20899
United States of America
Email: ksriram@nist.gov

Sriram, et al. Expires 17 December 2022
Igor Lubashev
Akamai Technologies
145 Broadway
Cambridge, MA 02142
United States of America
Email: ilubashe@akamai.com

Doug Montgomery
USA National Institute of Standards and Technology
100 Bureau Drive
Gaithersburg, MD 20899
United States of America
Email: dougm@nist.gov