Workgroup: Network Working Group
Internet-Draft:
draft-ietf-sidrops-aspa-verification-14
Published: 19 April 2023
Intended Status: Standards Track
Expires: 21 October 2023
Authors: A. Azimov, Yandex; E. Bogomazov, Qrator Labs; R. Bush, IIJ & Arrcus; K. Patel, Arrcus; J. Snijders, Fastly; K. Sriram, USA NIST

BGP AS_PATH Verification Based on Autonomous System Provider Authorization (ASPA) Objects

Abstract

This document describes procedures that make use of Autonomous System Provider Authorization (ASPA) objects in the Resource Public Key Infrastructure (RPKI) to verify the Border Gateway Protocol (BGP) AS_PATH attribute of advertised routes. This type of AS_PATH verification provides detection and mitigation of route leaks and improbable AS paths. It also to some degree provides protection against prefix hijacks with forged-origin or forged-path-segment.

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.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Border Gateway Protocol (BGP) as originally designed is known to be vulnerable to prefix (or route) hijacks and BGP route leaks [RFC7908]. Some existing BGP extensions are able to partially solve these problems. For example, Resource Public Key Infrastructure (RPKI) based route origin validation (RPKI-ROV) [RFC6480] [RFC6482] [RFC6811] [RFC9319] can be used to detect and filter accidental mis-originations. [RFC9234] or [I-D.ietf-grow-route-leak-detection-mitigation] can be used to detect and mitigate accidental route leaks. While RPKI-ROV can prevent accidental prefix hijacks, malicious forged-origin prefix hijacks can still occur [RFC9319]. RFC9319 includes some recommendations for reducing the attack surface for forged-origin prefix hijacks.

This document describes procedures that make use of Autonomous System Provider Authorization (ASPA) objects [I-D.ietf-sidrops-aspa-profile] in the RPKI to verify the BGP AS_PATH attribute of advertised routes. These new ASPA-based procedures automatically detect invalid AS_PATHs in announcements that are received from customers, lateral peers (defined in [RFC7908]), transit providers, IXP Route Servers (RS), RS-clients, and siblings. This type of AS_PATH verification provides detection and mitigation of route leaks and improbable AS paths. It also to some degree provides protection against prefix hijacks with forged-origin or forged-path-segment (Section 8). The protections provided by these procedures (together with RPKI-ROV) are based on cryptographic techniques, and they are effective against a majority of accidental and malicious actions.

ASPA objects are cryptographically signed registrations of customer-to-provider relationships and stored in a distributed database [I-D.ietf-sidrops-aspa-profile]. ASPA-based path verification is an incrementally deployable technique and provides benefits to early adopters in the context of limited deployment.

The procedures described in this document are applicable only for BGP routes with {AFI, SAFI} combinations {AFI 1 (IPv4), SAFI 1} and {AFI 2 (IPv6), SAFI 1} [IANA-AF]. SAFI 1 represents NLRI used for unicast forwarding [IANA-SAF].

For brevity, the term "provider" is often used instead of "transit provider" in this document; they mean the same.

1.1. Anomaly Propagation

Both route leaks and hijacks have similar effects on ISP operations - they redirect traffic and can result in denial of service (DoS), eavesdropping, increased latency, and packet loss. The level of risk, however, depends significantly on the extent of propagation of the anomalies. For example, a route leak or hijack that is propagated only to customers may cause bottlenecking within a particular ISP's customer cone, but if the anomaly propagates through lateral (i.e., non-transit) peers and transit providers, or reaches global distribution through transit-free networks, then the ill effects will likely be amplified and experienced across continents.

The ability to constrain the propagation of BGP anomalies to transit providers and lateral peers - without requiring support from the source of the anomaly (which is critical if the source has malicious intent) - should significantly improve the robustness of the global inter-domain routing system.

2. BGP Roles

For path verification purposes in this document, the BGP roles an AS can have in relation to a neighbor AS are customer, provider, lateral peer, RS, RS-client, and sibling. These relationships, except sibling, are defined in [RFC9234]. Sibling ASes MAY export everything (both customer and non-customer routes) to each other, i.e., consider each other as a customer. For sibling ASes, the customer-to-provider relationship applies in each direction.

All roles are configured locally and used for the registration of ASPA objects (Section 3, Section 4) and/or for path verification (Section 6). The procedure of BGP Role capability [RFC9234] in the BGP OPEN message to verify the role with a neighbor is RECOMMENDED. The procedure is not applied for verifying a sibling-to-sibling role since it is not specified in [RFC9234]. However, there is little concern about a pair of sibling ASes, since they have a trusted relationship. In fact, they are typically managed by a single entity.

3. Autonomous System Provider Authorization

An ASPA record is a digitally signed object that binds a set of Provider AS numbers to a Customer AS (CAS) number (in terms of BGP announcements) and is signed by the CAS [I-D.ietf-sidrops-aspa-profile]. The CAS can choose to specify an AFI (i.e., afiLimit = 1 for IPv4 or 2 for IPv6) in the ASPA or it may omit it in which case the ASPA applies to both IPv4 and IPv6. The ASPA attests that the CAS has a Set of Provider ASes (SPAS) as specified in the ASPA. The definition of Provider AS is given in Section 1 of [I-D.ietf-sidrops-aspa-profile]. A function of a Provider AS is to propagate a CAS's route announcements onward, i.e., to the Provider's upstream providers, lateral peers, or customers. Another function is to offer outbound (customer to Internet) data traffic connectivity to the Customer. The ASPA object profile is described in [I-D.ietf-sidrops-aspa-profile].

The notation (AS x, [{AS y1, afiLimit a1}, {AS y2, afiLimit a2}, ...]), is used to represent an ASPA object for a CAS denoted as AS x. In this notation, the set {AS y1, AS y2, ...} represent the Set of Provider ASes (SPAS) of AS x and each Provider AS has an associated afiLimit (shown as a1, a2,... etc., respectively). The afiLimit may have a value of either 1 or 2 (meaning AFI = 1 or AFI = 2). It may also be left unspecified, in which case the Provider AS applies for both AFI = 1 and AFI = 2. A CAS is expected to register a single ASPA listing all its Provider ASes (see Section 4). If a CAS has a single ASPA, then the SPAS for the CAS is the set of Provider ASes listed in that ASPA. In case a CAS has multiple ASPAs, then the SPAS is the union of the Provider ASes listed in all ASPAs of the CAS.

Verified ASPA Payload (VAP) refers to the payload in a cryptographically verified (i.e., X.509 valid [RFC3779] [RFC5280]) ASPA object. In the procedures for the AS path verification described in this document (Section 5, Section 6), VAP-SPAS refers to the set of provider ASes derived from the VAP(s) of the CAS in consideration.

4. ASPA Registration Recommendations

It is RECOMMENDED that the afiLimit parameter in the ASPA object be left unspecified (unless there is a compelling reason to specify) so that the ASPA applies to both IPv4 and IPv6 prefixes. This gives the CAS significant flexibility, e.g., the need to scramble to modify the ASPA registrations can be averted when adding or moving IPv4 and IPv6 route announcements across different providers.

An ASPA object showing only AS 0 as a provider AS is referred to as an AS0 ASPA. A non-transparent Route Server AS (RS AS) is one that includes its AS number in the AS_PATH. Registering as AS0 ASPA is a statement by the registering AS that it has no transit providers, and it is also not an RS-client at a non-transparent RS AS. If that statement is true for both AFIs (IPv4 and IPv6), then the AS MUST register an AS 0 ASPA including only AS 0 as a provider. If that statement is true for only one AFI, then the AS MUST include in its ASPA only AS 0 as a provider for that AFI and applicable other ASes as providers for the other AFI. In general, an AS MUST include in its ASPA all its provider ASes and any non-transparent RS AS(es) at which it is an RS-client. A compliant AS, including a Route Server AS (RS AS), MUST have an ASPA. An AS SHOULD NOT have more than one ASPA. An RS AS SHOULD register an AS 0 ASPA without afiLimit.

If, despite the above recommendations, the ASPA(s) of a CAS includes SPAS for one AFI but not for the other AFI (not even an AS 0), the ASPA SHALL NOT be rejected just for that reason. However, such an ASPA(s) will be presumed to imply that the CAS has no providers (equivalent to AS 0 SPAS) for the AFI that they neglected to include.

As mentioned before, the set of provider ASes contained in the VAP(s) is referred to as the VAP-SPAS of the AS registering the ASPA(s). Normally, a VAP-SPAS is not expected to contain both an AS 0 and other Provider ASes for the same AFI, but an unexpected presence of AS 0 has no influence on the AS path verification procedures (see Section 5, Section 6).

Each of the two ASes in a sibling pair MUST register its ASPA including the other AS in its SPAS. If one of the ASes in the pair does this registration but the other does not, that contributes to the risk of not getting the correct AS path verification result for routes that include the pair.

The ASes on the boundary of an AS Confederation MUST register ASPAs using the Confederation's global ASN as the CAS.

As specified earlier, a compliant AS should maintain a single ASPA object that includes all its provider ASes, including any non-transparent RS ASes. Such a practice helps prevent race conditions during ASPA updates that might affect prefix propagation. The software that provides hosting for ASPA records SHOULD support enforcement of this practice. During a transition process between different certificate authority (CA) registries, the ASPA records SHOULD be kept identical in all registries.

5. Hop-Check Function

Let AS(i) and AS(j) represent adjacent unique ASes in an AS_PATH, and thus (AS(i), AS(j)) represents an AS hop. A hop-check function, hop(AS(i), AS(j), AFI), checks if the ordered pair of ASNs, (AS(i), AS(j)), has the property that AS(j) is an attested provider of AS(i) per VAP-SPAS of AS(i) for the specified AFI. The VAP-SPAS table is assumed to be organized in such a way that it can be queried to check (1) if for a specified CAS = AS(i), there is an entry (i.e., SPAS listed), or (2) if for a given (AS(i), AS(j), AFI) tuple, AS(j) is listed in the VAP-SPAS as a provider associated with CAS = AS(i) for the specified AFI value. A provider AS ID included in the SPAS can correspond to a Provider, a non-transparent RS, or a Sibling. A non-transparent RS is effectively a Provider to its RS-client. Siblings regard each other as a Provider (see Section 4). The term "Provider+" in the definition of the hop-check function is meant to encompass all three possibilities: Provider, non-transparent RS, or Sibling. This function is specified as follows:

```
                          /
                          | "No Attestation" if there is no entry
                          |   in VAP-SPAS table for CAS = AS(i)
                          |
hop(AS(i), AS(j), AFI) =  / Else, "Provider+" if VAP-SPAS entry for
                          \   CAS = AS(i) for the mentioned AFI includes
                          |
                          | Else, "Not Provider+"
                          \
```

Figure 1: Hop-check function.

To be clear, this function checks if AS(j) is included in the VAP-SPAS of AS(i), and in doing so it does not need to distinguish between Provider, non-transparent RS, and Sibling.

The hop-check function is AFI dependent because an AS may have different SPAS for different AFI. This function is used in the ASPA-based AS_PATH verification algorithms described in Section 6.1 and Section 6.2. For simplicity, while describing the algorithms, the function hop(AS(i), AS(j), AFI) is replaced with hop(AS(i), AS(j)) by dropping the AFI since it is understood that the algorithms are run for a specific AFI at a time (AFI = 1 or AFI = 2).

For purposes such as computational efficiency, memory savings, etc., an implementation may make its own choice regarding maintaining a single VAP-SPAS table or two separate tables (i.e., one per AFI).

6. AS_PATH Verification

The procedures described in this document are applicable only to four-octet AS number compatible BGP speakers [RFC6793]. If such a BGP speaker receives both AS_PATH and AS4_PATH attributes in an UPDATE, then the procedures are applied on the reconstructed AS path (Section 4.2.3 of [RFC6793]). So, the term AS_PATH is used in this document to refer to the usual AS_PATH [RFC4271] as well as the reconstructed AS path.

If an attacker creates a route leak intentionally, they may try to strip their AS from the AS_PATH. To partly guard against that, a check is necessary to match the most recently added AS in the AS_PATH to the BGP neighbor's ASN. This check MUST be performed as specified in Section 6.3 of [RFC4271]. If the check fails, then the AS_PATH is considered a Malformed AS_PATH and the UPDATE is considered to be in error (Section 6.3 of [RFC4271]). The case of transparent RS MUST also be appropriately taken care of (e.g., by suspending the neighbor ASN check). The check fails also when the

AS_PATH is empty (zero length) and such UPDATEs will also be considered to be in error.

[I-D.ietf-idr-deprecate-as-set-confed-set] specifies that "treat-as-withdraw" error handling SHOULD be applied to routes with AS_SET in the AS_PATH. In this document, routes with AS_SET are given Invalid evaluation in the AS_PATH verification procedures (Section 6.1 and Section 6.2).

Wherever AFI is mentioned in the AS_PATH verification algorithms, it refers to the AFI of the prefix in the route for which the AS_PATH verification is performed. When an AS_PATH is evaluated as Valid, Invalid, or Unknown, it pertains only to the AFI for which the verification was performed. The same AS_PATH can have a different verification outcome for a different AFI. Since it is understood that the algorithms described here are run for a single AFI at a time (pertaining to the route(s) being verified), the AFI in the function hop(AS(i), AS(j), AFI) is not shown explicitly for the sake of simplicity.

6.1. Algorithm for Upstream Paths

The upstream verification algorithm described here is applied when a route is received from a customer or lateral peer, or is received by an RS from an RS-client, or is received by an RS-client from an RS. In all these cases, the receiving/validating AS expects the AS_PATH to consist of only customer-to-provider hops successively from the origin AS to the neighbor AS (most recently added).

The basic principles of the upstream verification algorithm are stated here. Let the sequence {AS(N), AS(N-1),..., AS(2), AS(1)} represent the AS_PATH in terms of unique ASNs, where AS(1) is the origin AS and AS(N) is the most recently added AS and neighbor of the receiving/validating AS. For each hop AS(i-1) to AS(i) in this sequence, the hop-check function, hop(AS(i-1), AS(i)), must equal "Provider+" (Section 5) for the AS_PATH to be Valid. If the hop-check function for at least one of those hops is "Not Provider+", then the AS_PATH is deemed Invalid. If the AS_PATH verification outcome is neither Valid nor Invalid (per the above principles), then it is evaluated as Unknown.

The upstream path verification procedure is specified as follows:

1. If the AS_PATH has an AS_SET, then the procedure halts with the outcome "Invalid".

2. Collapse prepends in the AS_SEQUENCE(s) in the AS_PATH (i.e., keep only the unique AS numbers). Let the resulting ordered sequence be represented by {AS(N), AS(N-1), ..., AS(2), AS(1)},

where AS(1) is the first-added (i.e., origin) AS and AS(N) is the last-added AS and neighbor to the receiving/validating AS.

3. If N = 1, then the procedure halts with the outcome "Valid". Else, continue.

4. At this step, N >= 2. If there is an i such that 2 <= i <= N and hop(AS(i-1), AS(i)) = "Not Provider+", then the procedure halts with the outcome "Invalid". Else, continue.

5. If there is an i such that 2 <= i <= N and hop(AS(i-1), AS(i)) = "No Attestation", then the procedure halts with the outcome "Unknown". Else, the procedure halts with the outcome "Valid".

6.2. Algorithm for Downstream Paths

The downstream verification algorithm described here is applied when a route is received from a transit provider or sibling AS. As described in Section 4, a sending sibling AS acts towards its receiving sibling AS in a manner similar to that of a provider towards its customer.

It is not essential, but the reader may take a look at the illustrations and formal proof in [sriram1] to develop a clearer understanding of the algorithm described here.

Here again (as in Section 6.1), let the AS_PATH be simplified and represented by the ordered sequence of unique ASNs as {AS(N), AS(N-1),..., AS(2), AS(1)}.

If 1 <= N <= 2, then the AS_PATH is trivially Valid.

The rest of the section assumes that the AS_PATH contains 3 or more unique ASNs (N >= 3).

Determination of Invalid AS_PATH:

Given the above-mentioned ordered sequence, if there exist indices u and v such that (1) u <= v, (2) hop(AS(u-1), AS(u)) = "Not Provider+", and (3) hop(AS(v+1), AS(v)) = "Not Provider+", then the AS_PATH is Invalid.

Determination of Valid AS_PATH:

As shown in Figure 2, assume that the ASes in the AS_PATH are in the same physical (locational) order as in the sequence representation {AS(N), AS(N-1),..., AS(2), AS(1)}, i.e., AS(N) is the left-most and AS(1) the right-most.

```
                 AS(L) ............. AS(K)
                  /                    \
              AS(L+1)                AS(K-1)
                 .                      .
                .                        .
 (down-ramp)   .                          .   (up-ramp)
              .                            .
             .                              .
           AS(N-1)                        AS(2)
             /                              \
          AS(N)                             AS(1)
           /                              (Origin AS)
 Receiving & Validating AS

     Each ramp has consecutive ASPA-attested
     customer-to-provider hops in the bottom-to-top direction
```

Figure 2: Illustration of up-ramp and down-ramp.

Looking at Figure 2, the UPDATE is received from a provider or a sibling (i.e., AS(N) is a provider or sibling). The AS_PATH may have both an up-ramp (on the right starting at AS(1)) and a down-ramp (on the left starting at AS(N)). The ramps are described as a sequence of ASes that consists of consecutive customer-to-provider hops. The up-ramp starts at AS(1) and each AS hop, (AS(i), AS(i+1)), in it has the property that hop(AS(i), AS(i+1)) = "Provider+" for i = 1, 2,... , K-1. If such a K does not exist, then K is set to 1. The up-ramp ends (reaches its apex) at AS(K) because hop(AS(K), AS(K+1)) = "Not Provider+" or "No Attestation". The down-ramp runs backward from AS(N) to AS(L). Each AS hop, (AS(j), AS(j-1)), in it has the property that hop(AS(j), AS(j-1)) = "Provider+" for j = N, N-1,... , L+1. If such an L does not exist, then L is set to N. The down-ramp ends at AS(L) because hop(AS(L), AS(L-1)) = "Not Provider+" or "No Attestation". Thus, the apex of the down-ramp is AS(L).

If there is an up-ramp that runs across all ASes in the AS_PATH (i.e., K = N), then clearly the AS_PATH is Valid. Similarly, if there is a down-ramp that runs across all ASes in the AS_PATH (i.e., L = 1), then also the AS_PATH is Valid. However, if both ramps exist in an AS_PATH with K < N and L > 1, then the AS_PATH is Valid if and only if L-K <= 1. Note that K could be greater than L (i.e., L-K has a negative value), which means that the up-ramp and down-ramp overlap, and that could happen when some adjacent AS pairs in the AS_PATH have mutually registered sibling relationships (i.e., include each other in their respective SPAS) (see Section 4). If L-K = 0, it means that the apexes of the up-ramp and down-ramp are at the same AS. If L-K = 1, it means that the apexes are at adjacent ASes. In summary, the AS_PATH is Valid if L-K is 0 or 1 or has a negative value.

Determination of Unknown AS_PATH:

If L-K >= 2, then the AS_PATH is either Invalid (route leak) or Unknown (see illustrations and proof in [sriram1]). However, if L-K >= 2 and an Invalid outcome was not found by the process described earlier in this section, then the AS_PATH is determined to be Unknown.

The downstream path verification procedure is formally specified as follows:

1. If the AS_PATH has an AS_SET, then the procedure halts with the outcome "Invalid".

2. Collapse prepends in the AS_SEQUENCE(s) in the AS_PATH (i.e., keep only the unique AS numbers). Let the resulting ordered sequence be represented by {AS(N), AS(N-1), ..., AS(2), AS(1)}, where AS(1) is the first-added (i.e., origin) AS and AS(N) is the last-added AS and neighbor to the receiving/validating AS.

3. If 1 <= N <= 2, then the procedure halts with the outcome "Valid". Else, continue.

4. At this step, N >= 3. Given the above-mentioned ordered sequence, find the lowest value of u (2 <= u <= N) for which hop(AS(u-1), AS(u)) = "Not Provider+". Call it u_min. If no such u_min exists, set u_min = N+1. Find the highest value of v (N-1 >= v >= 1) for which hop(AS(v+1), AS(v)) = "Not Provider+". Call it v_max. If no such v_max exists, then set v_max = 0. If u_min <= v_max, then the procedure halts with the outcome "Invalid". Else, continue.

5. Up-ramp: For 2 <= i <= N, determine the largest K such that hop(AS(i-1), AS(i)) = "Provider+" for each i in the range 2 <= i <= K. If such a largest K does not exist, then set K = 1.

6. Down-ramp: For N-1 >= j >= 1, determine the smallest L such that hop(AS(j+1), AS(j)) = "Provider+" for each j in the range N-1 >= j >= L. If such smallest L does not exist, then set L = N.

7. If L-K <= 1, then the procedure halts with the outcome "Valid". Else, the procedure halts with the outcome "Unknown".

In the above procedure, the computations in Steps 4, 5, and 6 can be done at the same time.

7. AS_PATH Verification Recommendations

Conforming implementations of this specification are not required to implement the AS_PATH verification procedures (step-wise lists) exactly as described in Section 6.1 and Section 6.2 but MUST provide functionality equivalent to the external behavior resulting from those procedures. In other words, the algorithms used in a specific implementation may differ, for example, for computational efficiency purposes, but the AS_PATH verification outcomes MUST be identical to those obtained by the procedures described in Section 6.1 and Section 6.2.

The above applies to eBGP routers in general, including those on the boundary of an AS Confederation facing external ASes. However, the procedures for ASPA-based AS_PATH verification in this document are NOT RECOMMENDED for use on eBGP links internal to the Confederation.

The procedures described in this document MUST be applied to BGP routes with {AFI, SAFI} combinations {AFI 1 (IPv4), SAFI 1} and {AFI 2 (IPv6), SAFI 1} [IANA-AF]. The procedures MUST NOT be applied to other address families by default.

8. Mitigation

If the AS_PATH is determined to be Invalid based on the verification procedures specified above (Section 6), then the route SHOULD be rejected. Also, for any route with an Invalid AS_PATH, the cause of the invalidity SHOULD be logged for monitoring and diagnostic purposes.

The ASPA-based path verification procedures are able to check routes received from customers, lateral peers, transit providers, RSes, RS-clients, and siblings. These procedures combined with BGP Roles [RFC9234] and RPKI-ROV [RFC6811] [RFC9319] can provide a fully automated solution to detect and filter many of the ordinary prefix hijacks, route leaks, and prefix hijacks with forged-origin or forged-path-segment (see Property 3 below).

The ASPA-based path verification has the following properties (detection capabilities):

Property 1: Let AS A and AS B be any two ASes in the Internet doing ASPA (registration and verification) and no assumption is made about the deployment status of other ASes. Consider a route propagated from AS A to a customer or lateral peer. The route is subsequently leaked by an offending AS in the AS path before being received at AS B on a customer or lateral peer interface. The ASPA-based path verification at AS B always detects such a route leak though it may not be able to identify the AS that originated the leak. This assertion is true even when the sender

AS A (or receiver AS B) is an RS AS and the neighbor AS that AS A sent to (or AS B received from) is an RS-client.

Property 2: Again, let AS A and AS B be any two ASes in the Internet doing ASPA (registration and verification) and no assumption is made about the deployment status of other ASes. Consider a route received at AS B on a customer or lateral peer interface that is a forged-origin prefix hijack involving AS A as the forged-origin. The ASPA-based path verification at AS B always detects such a forged-origin prefix hijack.

Property 3: This is an extension of Property 2 above to the case of prefix hijacking with a forged-path-segment. Such hijacking refers to the forging of multiple contiguous ASes in an AS path beginning with the origin AS. Again, let AS A and AS B be any two ASes in the Internet doing ASPA (registration and verification). Let AS A's providers, AS P and AS Q, also be registering ASPA. No assumption is made about the ASPA deployment status of any other ASes in the Internet. Consider a route received at AS B on a customer or lateral peer interface that is a prefix hijack with a forged-path-segment {AS P, AS A} or {AS Q, AS A}. That is, the hijacker attaches this path-segment at the beginning of their route announcement. The ASPA-based path verification at AS B always detects such a forged-path-segment prefix hijack. For a chance to be successful (remain undetected by AS B), the hijacker may resort to a forged-path-segment with three ASes including a provider AS of AS P (or AS Q). But even that can be foiled (detected) if the providers of AS P and AS Q also register ASPA. Having to use a longer forged-path-segment to avoid detection by AS B diminishes the ability of the hijacked route to compete with the corresponding legitimate route in path selection.

Property 4: Let AS A and AS B be any two ASes in the Internet doing ASPA (registration and verification). Assume that AS B does not drop a route detected as a leak, but only lowers its LOCAL_PREF [RFC4271]. Let such a route, selected and forwarded by AS B, be subsequently received at AS Z which is also doing ASPA. No assumption is made about the ASPA compliance of the ASes in the intervening path from AS B to AS Z. The ASPA-based path verification at AS Z always detects such received route as a leak regardless of the direction (type of peer) it was received from.

In the description of the properties listed above, the term "customer" can be replaced with "RS-client".

An observation that follows from Property #1 above is that if any two ISP ASes register ASPAs and implement the detection and mitigation procedures, then any route received from one of them and leaked to the other by a common customer AS (ASPA compliant or not) will be automatically detected and mitigated. In effect, if most major ISPs are compliant, the propagation of route leaks in the Internet will be severely limited.

The above properties show that ASPA-based path verification offers significant benefits to early adopters. Limitations of the method with regard to some forms of malicious AS path manipulations are discussed in Section 12.

9. Operational Considerations

9.1. 4-Byte AS Number Requirement

The procedures specified in this document are compatible only with BGP implementations that support 4-byte ASNs in the AS_PATH. This limitation should not have a real effect on operations since legacy BGP routers are rare, and it is highly unlikely that they support integration with the RPKI.

9.2. Correctness of the ASPA

ASPA issuers should be aware of the implications of ASPA-based AS path verification. Network operators must keep their ASPA objects correct and up to date. Otherwise, for example, if a provider AS is left out of the Set of Provider ASes (SPAS) in the ASPA, then routes containing the CAS (in the ASPA) and said provider AS may be incorrectly labeled as route leaks and rejected.

9.3. Make Before Break

ASPA issuers SHOULD apply the make-before-break principle while updating an ASPA registration. For example, when adding new Provider AS(es) in the SPAS, if the new ASPA is meant to replace a previously created ASPA, the latter SHOULD be decommissioned only after allowing sufficient time for the new ASPA to propagate to Relying Parties (RP) through the global RPKI system.

10. Comparison to Other Technologies

10.1. BGPsec

BGPsec [RFC8205] was designed to solve the problem of AS_PATH verification by including cryptographic signatures in BGP Update messages. It offers protection against unauthorized path modifications and assures that the BGPsec Update actually traveled the path shown in the BGPsec_PATH Attribute. However, it does not detect route leaks (valley-free violations). In comparison, the ASPA-based path verification described in this document detects if the AS path is improbable and focuses on detecting route leaks (including malicious cases) and forged-origin hijacks.

BGPsec and ASPA are complementary technologies.

10.2. Peerlock

The Peerlock mechanism [Peerlock] [Flexsealing] has a similar objective as the APSA-based route leak protection mechanism described in this document. It is commonly deployed by large Internet carriers to protect each other from route leaks. Peerlock depends on a laborious manual process in which operators coordinate the distribution of unstructured Provider Authorizations through out-of-band means in a many-to-many fashion. On the other hand, ASPA's use of the RPKI allows for automated, scalable, and ubiquitous deployment, making the protection mechanism available to a wider range of network operators.

The ASPA mechanism implemented in router code (in contrast to Peerlock's AS_PATH regular expressions) also provides a way to detect anomalies propagated from transit providers and IX route servers. ASPA is intended to be a complete solution and replacement for existing Peerlock deployments.

11. IANA Considerations

This document includes no request to IANA.

12. Security Considerations

While the ASPA-based mechanism is able to detect and mitigate the majority of mistakes and malicious activity affecting routes, it might fail to detect some malicious path modifications, especially for routes that are received from transit providers.

Since an upstream provider becomes a trusted point, in theory, it might be able to propagate some instances of hijacked prefixes with forged-origin or forged-path-segment or even routes with manipulated AS_PATHs, and such attacks might go undetected by its customers. This can be illustrated with some examples. In Figure 3, normally the receiving/validating AS located at the lower left side should receive a route with AS_PATH {AS(5), AS(4), AS(3), AS(2), AS(1)} and it would be Valid (Section 6.2) given all the ASPAs that are shown in the figure. However, if AS(5) which is a transit provider to the validating AS acts maliciously and sends the route with a shortened AS_PATH such as {AS(5), AS(3), AS(2), AS(1)} or {AS(5), AS(2), AS(1)}, such path manipulation would be undetectable (i.e., the AS_PATH would be considered Valid). Also, if AS(5) were to perform a forged-origin hijack by inserting an AS_PATH {AS(5), AS(1)}, that would also be undetectable.

```
                   AS(4) - AS(3)
                   /         \
   (down-ramp)    /           \    (up-ramp)
              AS(5)          AS(2)
                /              \
               /               AS(1)
              /             (Origin AS)
 Receiving & Validating AS

ASPAs: {AS(1), [AS(2)]}, {AS(2), [AS(3)]}, {AS(5), [AS(4)]},
       {AS(3), [AS 0]}, {AS(4), [AS 0]}
```

Figure 3: Illustration for discussion of undetectable AS_PATH manipulations.

While attacks like the examples above may happen, it does not seem to be a realistic scenario. Normally a customer and their transit provider would have a signed agreement, and a policy violation (of the above kind) should have legal consequences or the customer can just drop the relationship with such a provider and remove the corresponding ASPA record.

The key properties or strengths of the ASPA method were described in Section 8. If detection of any and all kinds of path manipulation attacks is the goal, then BGPsec [RFC8205] would need to be deployed complementary to the ASPA method. It may be noted that BGPsec in its current form lacks route leak detection capabilities.

13. Implementation Status

This section is to be removed before publishing 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. The inclusion of this section here follows the process 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 [RFC7942], "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".

* A BGP implementation OpenBGPD [bgpd] (version 7.8 and higher), written in C, was provided by Claudio Jeker, Theo Buehler, and Job Snijders.

* The implementation NIST-BGP-SRx [BGP-SRx] is a software suite that provides a validation engine (BGP-SRx) and a Quagga-based BGP router (Quagga-SRx). It includes unit test cases for testing the ASPA-based path verification. It was provided by Oliver Borchert, Kyehwan Lee, and their colleagues at US NIST. It requires some additional work to incorporate the latest changes in the draft specifications related to IXP RS AS and RS-client.

14. Acknowledgments

The authors wish to thank Claudio Jeker, Jakob Heitz, Amir Herzberg, Igor Lubashev, Ben Maddison, Russ Housley, Jeff Haas, Nan Geng, Nick Hilliard, Shunwan Zhuang, Yangyang Wang, Martin Hoffmann, Amreesh Phokeer, Aftab Siddiqui, Dai Zhibin, Doug Montgomery, Rich Compton, Andrei Robachevsky, and Iljitsch van Beijnum for comments, suggestions, and discussion on the path verification procedures or the text in the document. For the implementation and testing of the procedures in the document, the authors wish to thank Claudio Jeker and Theo Buehler [bgpd] as well as Kyehwan Lee and Oliver Borchert [BGP-SRx].

15. References

15.1. Normative References

[I-D.ietf-sidrops-aspa-profile]
Azimov, A., Uskov, E., Bush, R., Snijders, J., Housley, R., and B. Maddison, "A Profile for Autonomous System Provider Authorization", Work in Progress, Internet-Draft, draft-ietf-sidrops-aspa-profile-12, 29 January 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-sidrops-aspa-profile-12>.

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

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI

10.17487/RFC4271, January 2006, <https://www.rfc-editor.org/info/rfc4271>.

[RFC6480] Lepinski, M. and S. Kent, "An Infrastructure to Support Secure Internet Routing", RFC 6480, DOI 10.17487/RFC6480, February 2012, <https://www.rfc-editor.org/info/rfc6480>.

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

[RFC6793] Vohra, Q. and E. Chen, "BGP Support for Four-Octet Autonomous System (AS) Number Space", RFC 6793, DOI 10.17487/RFC6793, December 2012, <https://www.rfc-editor.org/info/rfc6793>.

[RFC6811] Mohapatra, P., Scudder, J., Ward, D., Bush, R., and R. Austein, "BGP Prefix Origin Validation", RFC 6811, DOI 10.17487/RFC6811, January 2013, <https://www.rfc-editor.org/info/rfc6811>.

[RFC7908] Sriram, K., Montgomery, D., McPherson, D., Osterweil, E., and B. Dickson, "Problem Definition and Classification of BGP Route Leaks", RFC 7908, DOI 10.17487/RFC7908, June 2016, <https://www.rfc-editor.org/info/rfc7908>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC9234] Azimov, A., Bogomazov, E., Bush, R., Patel, K., and K. Sriram, "Route Leak Prevention and Detection Using Roles in UPDATE and OPEN Messages", RFC 9234, DOI 10.17487/RFC9234, May 2022, <https://www.rfc-editor.org/info/rfc9234>.

15.2. Informative References

[BGP-SRx] NIST, "BGP Secure Routing Extension (BGP-SRx) Software Suite", NIST Open-Source Software , <https://www.nist.gov/services-resources/software/bgp-secure-routing-extension-bgp-srx-software-suite>.

[bgpd] Jeker, C., "OpenBGPD", <http://www.openbgpd.org/>.

[Flexsealing] McDaniel, T., Smith, J., and M. Schuchard, "Flexsealing BGP Against Route Leaks: Peerlock Active Measurement and Analysis", November 2020, <https://arxiv.org/pdf/2006.06576.pdf>.

[I-D.ietf-grow-route-leak-detection-mitigation]
Sriram, K. and A. Azimov, "Methods for Detection and Mitigation of BGP Route Leaks", Work in Progress, Internet-Draft, draft-ietf-grow-route-leak-detection-mitigation-08, 24 October 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-grow-route-leak-detection-mitigation-08>.

[I-D.ietf-idr-deprecate-as-set-confed-set]
Kumari, W. A., Sriram, K., Hannachi, L., and J. Haas, "Deprecation of AS_SET and AS_CONFED_SET in BGP", Work in Progress, Internet-Draft, draft-ietf-idr-deprecate-as-set-confed-set-10, 16 January 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-idr-deprecate-as-set-confed-set-10>.

[IANA-AF] IANA, "Address Family Numbers", <https://www.iana.org/assignments/address-family-numbers/address-family-numbers.xhtml>.

[IANA-SAF] IANA, "Subsequent Address Family Identifiers (SAFI) Parameters", <https://www.iana.org/assignments/safi-namespace/safi-namespace.xhtml>.

[Peerlock] Snijders, J., "Peerlock", June 2016, <https://www.nanog.org/sites/default/files/Snijders_Everyday_Practical_Bgp.pdf>.

[RFC3779] Lynn, C., Kent, S., and K. Seo, "X.509 Extensions for IP Addresses and AS Identifiers", RFC 3779, DOI 10.17487/RFC3779, June 2004, <https://www.rfc-editor.org/info/rfc3779>.

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

[RFC7942] Sheffer, Y. and A. Farrel, "Improving Awareness of Running Code: The Implementation Status Section", BCP 205, RFC 7942, DOI 10.17487/RFC7942, July 2016, <https://www.rfc-editor.org/info/rfc7942>.

[RFC8205] Lepinski, M., Ed. and K. Sriram, Ed., "BGPsec Protocol Specification", RFC 8205, DOI 10.17487/RFC8205, September 2017, <https://www.rfc-editor.org/info/rfc8205>.

[RFC9319] Gilad, Y., Goldberg, S., Sriram, K., Snijders, J., and B. Maddison, "The Use of maxLength in the Resource Public

Key Infrastructure (RPKI)", BCP 185, RFC 9319, DOI 10.17487/RFC9319, October 2022, <https://www.rfc-editor.org/info/rfc9319>.

[sriram1] Sriram, K. and J. Heitz, "On the Accuracy of Algorithms for ASPA Based Route Leak Detection", IETF SIDROPS Meeting, Proceedings of the IETF 110, March 2021, <https://datatracker.ietf.org/meeting/110/materials/slides-110-sidrops-sriram-aspa-alg-accuracy-01>.

Authors' Addresses

Alexander Azimov
Yandex
Ulitsa Lva Tolstogo 16
Moscow
119021
Russian Federation

Email: a.e.azimov@gmail.com

Eugene Bogomazov
Qrator Labs
1-y Magistralnyy tupik 5A
Moscow
123290
Russian Federation

Email: eb@qrator.net

Randy Bush
Internet Initiative Japan & Arrcus, Inc.
5147 Crystal Springs
Bainbridge Island, Washington 98110
United States of America

Email: randy@psg.com

Keyur Patel
Arrcus
2077 Gateway Place
Suite #400
San Jose, CA 95119
United States of America

Email: keyur@arrcus.com

Job Snijders
Fastly
Amsterdam

Netherlands

Email: job@fastly.com

Kotikalapudi Sriram
USA National Institute of Standards and Technology
100 Bureau Drive
Gaithersburg, MD 20899
United States of America

Email: ksriram@nist.gov