Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 26, 2021

A. Azimov
Yandex
E. Bogomazov
Qrator Labs
R. Bush
Internet Initiative Japan & Arrcus
K. Patel
Arrcus, Inc.
J. Snijders
NTT

February 22, 2021

Verification of AS_PATH Using the Resource Certificate Public Key Infrastructure and Autonomous System Provider Authorization draft-ietf-sidrops-aspa-verification-07

Abstract

This document defines the semantics of an Autonomous System Provider Authorization object in the Resource Public Key Infrastructure to verify the AS_PATH attribute of routes advertised in the Border Gateway Protocol.

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

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 August 26, 2021.

Table of Contents

1. Introduction

The Border Gateway Protocol (BGP) was designed without mechanisms to validate BGP attributes. Two consequences are BGP Hijacks and BGP Route Leaks [RFC7908]. BGP extensions are able to partially solve these problems. For example, ROA-based Origin Validation [RFC6483] can be used to detect and filter accidental mis-originations, and [I-D.ietf-idr-bgp-open-policy] or [I-D.ietf-grow-route-leak-detection-mitigation] can be used to detect accidental route leaks. While these upgrades to BGP are quite useful, they still rely on transitive BGP attributes, i.e. AS_PATH, that can be manipulated by attackers.

BGPSec [RFC8205] was designed to solve the problem of AS_PATH validation. Unfortunately, strict cryptographic validation brought expensive computational overhead for BGP routers. BGPSec also proved vulnerable to downgrade attacks that nullify the benefits of AS_PATH signing. As a result, to abuse the AS_PATH or any other signed transit attribute, an attacker merely needs to downgrade to 'old' BGP-4.

An alternative approach was introduced with soBGP [I-D.white-sobgp-architecture]. Instead of strong cryptographic AS_PATH validation, it created an AS_PATH security function based on a shared database of AS adjacencies. While such an approach has reasonable computational cost, the two side adjacencies don't provide a way to automate anomaly detection without high adoption rate - an attacker can easily create a one-way adjacency. SO-BGP transported data about adjacencies in new additional BGP messages, which was recursively complex thus significantly increasing adoption complexity and risk. In addition, the general goal to verify all AS_PATHs was not achievable given the indirect adjacencies at internet exchange points.

Instead of checking AS_PATH correctness, this document focuses on solving real-world operational problems - automatic detection of malicious hijacks and route leaks. To achieve this new AS_PATH verification procedures are defined to automatically detect invalid (malformed) AS_PATHs in announcements that are received from customers, peers, providers, RS and RS-clients. This procedures uses a shared signed database of customer-to-provider relationships using a new RPKI object - Autonomous System Provider Authorization (ASPA). This technique provides benefits for participants even during early and incremental adoption.

2. Anomaly Propagation

Both route leaks and hijacks have similar effects on ISP operations - they redirect traffic, resulting in increased latency, packet loss, or possible MiTM attacks. But the level of risk depends significantly on the propagation of the anomalies. For example, a hijack that is propagated only to customers may concentrate traffic in a particular ISP's customer cone; while if the anomaly is propagated through peers, upstreams, or reaches Tier-1 networks, thus distributing globally, traffic may be redirected at the level of entire countries and/or global providers.

The ability to constrain propagation of BGP anomalies to upstreams and peers, without requiring support from the source of the anomaly (which is critical if source has malicious intent), should

significantly improve the security of inter-domain routing and solve the majority of problems.

3. Autonomous System Provider Authorization

As described in [RFC6480], the RPKI is based on a hierarchy of resource certificates that are aligned to the Internet Number Resource allocation structure. Resource certificates are X.509 certificates that conform to the PKIX profile [RFC5280], and to the extensions for IP addresses and AS identifiers [RFC3779]. A resource certificate is a binding by an issuer of IP address blocks and Autonomous System (AS) numbers to the subject of a certificate, identified by the unique association of the subject's private key with the public key contained in the resource certificate. The RPKI is structured so that each current resource certificate matches a current resource allocation or assignment.

ASPA is digitally signed object that bind, for a selected AFI, a Set of Provider AS numbers to a Customer AS number (in terms of BGP announcements not business), and are signed by the holder of the Customer AS. An ASPA attests that a Customer AS holder (CAS) has authorized Set of Provider ASes (SPAS) to propagate the Customer's IPv4/IPv6 announcements onward, e.g. to the Provider's upstream providers or peers. The ASPA record profile is described in [I-D.ietf-sidrops-aspa-profile]. For a selected Customer AS SHOULD exist only single ASPA object at any time. In this document we will use ASPA(AS1, AFI, [AS2, ...]) as notation to represent ASPA object for AS1 in the selected AFI.

4. Customer-Provider Verification Procedure

This section describes an abstract procedure that checks that a pair of ASNs (AS1, AS2) is included in the set of signed ASPAs. The semantics of its use is defined in next section. The procedure takes (AS1, AS2, AFI) as input parameters and returns one of three results: "Valid", "Invalid" and "Unknown".

A relying party (RP) must have access to a local cache of the complete set of cryptographically valid ASPAs when performing customer-provider verification procedure.

1. Retrieve all cryptographically valid ASPAs in a selected AFI with a customer value of AS1. The union of SPAS forms the set of "Candidate Providers."

2. If the set of Candidate Providers is empty, then the procedure exits with an outcome of "Unknown."

3. If AS2 is included in the Candidate Providers, then the procedure exits with an outcome of "Valid."

4. Otherwise, the procedure exits with an outcome of "Invalid."

Since an AS1 may have different set of providers in different AFI, it should also have different PCAS in corresponding ASPAs. In this case, the output of this procedure with input (AS1, AS2, AFI) may have different output for different AFI values.

5. AS_PATH Verification

The AS_PATH attribute identifies the autonomous systems through which an UPDATE message has passed. AS_PATH may contain two types of components: AS_SEQUENCEs and AS_SETs, as defined in [RFC4271].

We will use index of AS_PATH segments, where Seg(0) stands for the segment of originating AS. We will use Seg(I).value and Seg(I).type to represent Ith segment value and type respectively.

The below procedures are applicable only for 32-bit AS number compatible BGP speakers.

5.1. Upstream Paths

When a route is received from a customer, a literal peer, or by a RS at an IX, each consecutive AS_SEQUENCE pair MUST be equal (prepend policy) or belong to customer-provider or mutual transit relationship (Section 7). If there are other types of relationships, it means that the route was leaked or the AS_PATH attribute was malformed. The goal of the procedure described below is to check the correctness of this statement.

The following Python function and algorithm describes the procedure that MUST be applied on routes with AFI received from a customer, peer or RS-client:

```
def check_upflow_path(aspath, neighbor_as, afi):
    if len(aspath) == 0:
        return Invalid

    if aspath[-1].type == AS_SEQUENCE and aspath[-1].value != neighbor_as:
        return Invalid

    semi_state = Valid

    as1 = 0
    for segment in aspath:
        if segment.type != AS_SEQUENCE:
            as1 = 0
            semi_state = Unverifiable
        elif segment.type == AS_SEQUENCE:
            if not as1:
                as1 = segment.value
            elif as1 == segment.value:
                continue
            else:
                pair_check = verify_pair(as1, segment.value, afi)
                if pair_check == Invalid:
                    return Invalid
                elif pair_check == Unknown and semi_state == Valid:
                    semi_state = pair_check
                as1 = segment.value
    return semi_state
```

1. If the AS_PATH has zero length then procedure halts with the outcome "Invalid";

2. If the last segment in the AS_PATH has type AS_SEQUENCE and its value isn't equal to receiver's neighbor AS then procedure halts with the outcome "Invalid";

3. If there exists I such that Seg(I-1).type and Seg(I).type equal to AS_SEQUENCE, Seg(I-1).value != Seg(I).value and customer-provider verification procedure (Section 4) with parameters (Seg(I-1).value, Seg(I).value, AFI) returns "Invalid" then the procedure also halts with the outcome "Invalid";

4. If the AS_PATH has at least one AS_SET segment then procedure halts with the outcome "Unverifiable";

5. If there exists I such that Seg(I-1).type and Seg(I).type equal to AS_SEQUENCE, Seg(I-1).value != Seg(I).value and customer-provider verification procedure (Section 4) with parameters

(Seg(I-1).value, Seg(I).value, AFI) returns "Unknown" then the procedure also halts with the outcome "Unknown";

6. Otherwise, the procedure halts with an outcome of "Valid".

5.2. Downstream Paths

When route is received from provider it may have both Upstream and Downstream fragments, where a Downstream follows an Upstream fragment. If the path differs from this rule, e.g. the Downstream fragment is followed by Upstream fragment it means that the route was leaked or the AS_PATH attribute was malformed. The first unequal pair of AS_SEQUENCE segments that has an "Invalid" outcome of the customer-provider verification procedure indicates the end of the Upstream fragment. All subsequent reverse pairs of AS_SEQUENCE segments MUST be equal (prepend policy) or belong to a customer-provider or mutual transit relationship Section 7, thus can be also verified using ASPA objects.

The following Python function and algorithm describe the procedure that MUST be applied on routes with AFI received from a provider:

```
    def check_downflow_path(aspath, neighbor_as, afi):
        if len(aspath) == 0:
            return Invalid

        if aspath[-1].type == AS_SEQUENCE and aspath[-1].value != neighbor_as:
            return Invalid
        else:
            semi_state = Valid

        as1 = 0
        upflow_fragment = True
        for segment in aspath:
            if segment.type != AS_SEQUENCE:
                as1 = 0
                semi_state = Unverifiable
            elif segment.type == AS_SEQUENCE:
                if not as1:
                    as1 = segment.value
                elif as1 == segment.value:
                    continue
                else:
                    if upflow_fragment:
                        pair_check = verify_pair(as1, segment.value, afi)
                        if pair_check == Invalid:
                            upflow_fragment = False
                        elif pair_check == Unknown and semi_state == Valid:
                            semi_state = Unknown
                    else:
                        pair_check = verify_pair(segment.value, as1, afi)
                        if pair_check == Invalid:
                            return Invalid
                        elif pair_check == Unknown and semi_state == Valid:
                            semi_state = pair_check
                    as1 = segment.value

        return semi_state
```

1. If the AS_PATH has zero length then procedure halts with the outcome "Invalid";

2. If a route is received from a provider and the last segment in the AS_PATH has type AS_SEQUENCE and its value isn't equal to receiver's neighbor AS, then the procedure halts with the outcome "Invalid";

3. Let's define I_MIN as the minimal index for which Seg(I-1).type and Seg(I).type equal to AS_SEQUENCE, its values aren't equal and the verification procedure for (Seg(I-1).value, Seg(I).value,

AFI) returns "Invalid". If I_MIN doesn't exist put the length of AS_PATH in I_MIN variable and jump to 5.

4. If there exists J > I_MIN such that both Seg(J-1).type, Seg(J).type equal to AS_SEQUENCE, Seg(J-1).value != Seg(J).value and the customer-provider verification procedure (Section 4) returns "Invalid" for (Seg(J).value, Seg(J-1).value, AFI), then the procedure halts with the outcome "Invalid";

5. If the AS_PATH has at least one AS_SET segment then procedure halts with the outcome "Unverifiable";

6. If there exists J > I_MIN such that both Seg(J-1).type, Seg(J).type equal to AS_SEQUENCE, Seg(J-1).value != Seg(J).value and the customer-provider verification procedure (Section 4) returns "Unknown" for (Seg(J).value, Seg(J-1).value, AFI), then the procedure halts with the outcome "Unknown";

7. If there exists I_MIN > J such that both Seg(J-1).type, Seg(J).type equal to AS_SEQUENCE, Seg(J-1).value != Seg(J).value and the customer-provider verification procedure (Section 4) returns "Unknown" for (Seg(J-1).value, Seg(J).value, AFI), then the procedure halts with the outcome "Unknown";

8. Otherwise, the procedure halts with an outcome of "Valid".

5.3. Paths from Route Server

A route received from a RS at IX has much in common with route received from a provider. A valid route from RS contains Upflow fragment and MAY contain Downflow fragment that contains IX AS. The ambiguity is created by transparent IXes that by default don't add their AS in the AS_PATH. In this case, a route will have only Upflow segment, though even 'transparent' IXes may support control communities that give a way to explicitly add IX AS in the path.

Routes from RS MAY be processed the same way as routes from Providers, but in the case of full IX 'transparency', it will limit the opportunity of IX members to detect and filter route leaks. This document suggests using the presence of IX AS as a token to distinguish if Upflow or Downflow path verification procedure should be applied.

The following Python function and algorithm describe the procedure that SHOULD be applied on routes with AFI received from a RS:

```
def check_ix_path(aspath, neighbor_as, afi):
    if len(aspath) == 0:
        return Invalid

    if aspath[-1].value != neighbor_as:
        return check_upflow_path(aspath, aspath[-1].value, afi)
    else:
        return check_downflow_path(aspath, neighbor_as, afi)
```

1. If the AS_PATH has zero length then procedure halts with the outcome "Invalid";

2. If a route is received from a RS and the last segment in the AS_PATH isn't equal to receiver's neighbor AS, the result equals to the outcome of upflow verification procedure applied to AS_PATH with neighbor_as replaced with the value of the last AS_PATH segment Section 5.1;

3. If a route is received from a RS and the last segment in the AS_PATH is equal to receiver's neighbor AS, the result equals to the outcome of downflow verification procedure applied to AS_PATH Section 5.2;

5.4. Mitigation

If the output of the AS_PATH verification procedure is "Invalid" the route MUST be rejected.

If the output of the AS_PATH verification procedure is 'Unverifiable' it means that AS_PATH can't be fully checked. Such routes should be treated with caution and SHOULD be processed the same way as "Invalid" routes. This policy goes with full correspondence to [I-D.kumari-deprecate-as-set-confed-set].

The above AS_PATH verification procedure is able to check routes received from customer, peers, providers, RS, and RS-clients. The ASPA mechanism combined with BGP Roles [I-D.ietf-idr-bgp-open-policy] and ROA-based Origin Validation [RFC6483] can provide a fully automated solution to detect and filter hijacks and route leaks, including malicious ones.

6. Disavowal of Provider Authorizaion

An ASPA is a positive attestation that an AS holder has authorized its providers to redistribute received routes to the provider's providers and peers. This does not preclude the provider ASes from redistribution to its other customers. By creating an ASPA with providers set of [0], the customer indicates that no provider should

further announce its routes. Specifically, AS 0 is reserved to identify provider-free networks, Internet exchange meshes, etc.

An ASPA(AS, AFI, [0]) is a statement by the customer AS that its routes should not be received by any relying party AS from any of its customers or peers.

By convention, an ASPA(AS, AFI, [0]) should be the only ASPA issued by a given AS holder in the selected AFI; although this is not a strict requirement. An AS 0 may coexist with other provider ASes in the same ASPA (or other ASPA records in the same AFI); though in such cases, the presence or absence of the provider AS 0 in ASPA does not alter the AS_PATH verification procedure.

7. Mutual Transit (Complex Relations)

There are peering relationships which can not be described as strictly simple peer-peer or customer-provider; e.g. when both parties are intentionally sending prefixes received from each other to their peers and/or upstreams.

In this case, two corresponding records ASPA(AS1, AFI, [AS2, ...]), ASPA(AS2, AFI, [AS1, ...]) must be created by AS1 and AS2 respectively.

8. Comparison to Peerlock

ASPA has much in common with [Peerlock]. Peerlock is a BGP Flexsealing [Flexsealing] protection mechanism commonly deployed by global-scale Internet carriers to protect other large-scale carriers.

Peerlock, unfortunately, 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 PKIX [RFC5280] allows for automated, scalable, and ubiquitous deployment, making the protection mechanism available to a wider range of Internet Number Resource holders.

ASPA mechanics implemented in code instead of Peerlock AS_PATH regular expressions also provides a way to detect anomalies coming from transit providers and internet exchange route servers.

ASPA is intended to be a complete solution and replacement for existing Peerlock deployments.

9. Security Considerations

The proposed mechanism is compatible only with BGP implementations that can process 32-bit ASNs in the AS_PATH. This limitation should not have a real effect on operations - such legacy BGP routers are rare and it's highly unlikely that they support integration with the RPKI.

ASPA issuers should be aware of the verification implication in issuing an ASPA - an ASPA implicitly invalidates all routes passed to upstream providers other than the provider ASs listed in the ASPA record. It is the Customer AS's duty to maintain a correct set of providers in ASPA record(s).

While it's not restricted, but it's highly recommended maintaining for selected Customer AS a single ASPA object that covers all its providers. Such policy should prevent race conditions during ASPA updates that might affect prefix propagation. The software that provides hosting for ASPA records SHOULD support enforcement of this rule. In the case of the transition process between different CA registries, the ASPA records SHOULD be kept identical in all registries.

While the ASPA is able to detect both mistakes and malicious activity for routes received from customers, RS-clients, or peers, it provides only detection of mistakes for routes that are received from upstream providers and RS(s).

Since an upstream provider becomes a trusted point, it will be able to send hijacked prefixes of its customers or send hijacked prefixes with malformed AS_PATHs back. While it may happen in theory, it's doesn't seem to be a real scenario: normally customer and provider have a signed agreement and such policy violation should have legal consequences or customer can just drop relation with such a provider and remove the corresponding ASPA record.

10. Acknowledgments

The authors wish to thank authors of [RFC6483] since its text was used as an example while writing this document. The also authors wish to thank Iljitsch van Beijnum for giving a hint about Downstream paths.

11. References

11.1. Normative References

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

[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>.

11.2. Informative References

[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", draft-ietf-grow-route-leak-detection-mitigation-00 (work in progress), April 2019.

[I-D.ietf-idr-bgp-open-policy] Azimov, A., Bogomazov, E., Bush, R., Patel, K., and K. Sriram, "Route Leak Prevention using Roles in Update and Open messages", draft-ietf-idr-bgp-open-policy-05 (work in progress), February 2019.

[I-D.ietf-sidrops-aspa-profile] Azimov, A., Uskov, E., Bush, R., Patel, K., Snijders, J., and R. Housley, "A Profile for Autonomous System Provider Authorization", draft-ietf-sidrops-aspa-profile-00 (work in progress), May 2019.

[I-D.kumari-deprecate-as-set-confed-set] Kumari, W. and K. Sriram, "Deprecation of AS_SET and AS_CONFED_SET in BGP", draft-kumari-deprecate-as-set-confed-set-12 (work in progress), July 2018.

[I-D.white-sobgp-architecture] White, R., "Architecture and Deployment Considerations for Secure Origin BGP (soBGP)", draft-white-sobgp-architecture-02 (work in progress), June 2006.

[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>.

[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>.

[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>.

[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>.

[RFC6483] Huston, G. and G. Michaelson, "Validation of Route Origination Using the Resource Certificate Public Key Infrastructure (PKI) and Route Origin Authorizations (ROAs)", RFC 6483, DOI 10.17487/RFC6483, February 2012, <https://www.rfc-editor.org/info/rfc6483>.

[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>.

[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>.

Authors' Addresses

Alexander Azimov
Yandex

Email: a.e.azimov@gmail.com

Eugene Bogomazov
Qrator Labs

Email: eb@qrator.net

Randy Bush
Internet Initiative Japan & Arrcus

Email: randy@psg.com

Keyur Patel
Arrcus, Inc.

Email: keyur@arrcus.com

Job Snijders
NTT Communications
Theodorus Majofskistraat 100
Amsterdam 1065 SZ
The Netherlands

Email: job@ntt.net