SIDR Operations I. van Beijnum
Internet-Draft
BGPexpert.com
Updates: RFC 3779, RFC 8210 (if approved) June 20, 2019
Intended status: Experimental
Expires: December 22, 2019

Path validation with RPKI
draft-van-beijnum-sidrops-pathrpki-00

Abstract

This memo adds the capability to validate the full BGP AS path to the RPKI mechanism.

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 December 22, 2019.

Copyright Notice

1. Introduction

With RPKI, it's possible for BGP routers to validate the origin AS found in the BGP AS path attribute for a given IP prefix. However, RPKI can't validat the rest of the AS path, allowing for route leaks types 1 - 4 as described in RFC 7908 [RFC7908].

This specification extends RPKI to allow for validating the full BGP AS path, based on the observation that each AS in a valid AS path has either a trust relation with the origin AS or has a trust relation with the local AS (the AS performing validation). I.e., each intermediary AS provides transit service to either the origin AS or the local AS.

An extension to RFC 3779 [RFC3779] allows for binding a list of allowed transit ASes to a set of IP addresses. Operators of RPKI [RFC6480] relying party software add to this their list of locally allowed transit ASes through manual configuration. An update to the RPKI-router protocol [RFC8210] lets relying party software propagate the thus created list of allowed ASes for the prefix(es) in question so BGP routers can validate the corresponding AS paths.

1.1. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Changes to the ROA certificate format

RFC 3779 [RFC3779] specifies an extension to X.509 certificates that contains a set of AS numbers: id-pe-autonomousSysIds. This is the set of valid origin ASes for a given set of IP addresses.

This memo adds another extension to X.509 certificates with the name id-pe-autonomousSysIdsPath. id-pe-autonomousSysIdsPath is identical in syntax to the existing id-pe-autonomousSysIds, allowing for code reuse.

An explicit specification of the id-pe-autonomousSysIdsPath extension will be added to a later version of this document.

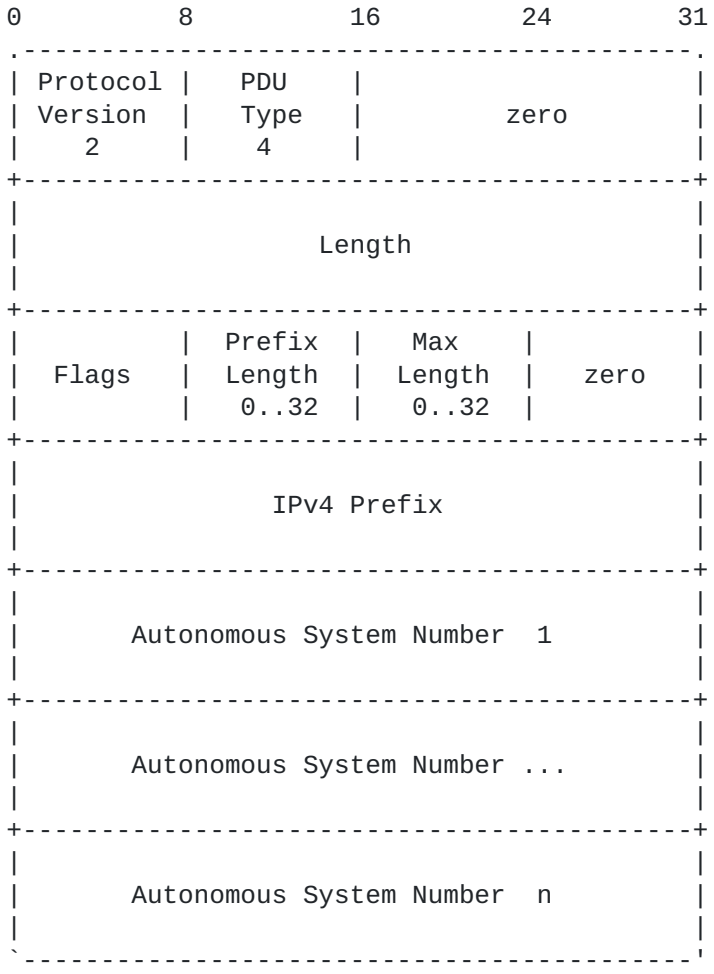
3. Changes to the RPKI-router protocol

This memo updates the RPKI-router protocol [RFC8210] by adding version 2 of the RPKI-router protocol. Version 2 is a superset of version 1; all implemenations that support version 2 MUST also support version 1. Version negotiation is performed as specified in

RFC 8210 [RFC8210], with the addition that version 2 may now be advertised and used if advertised by both sides.

Version 2 extends the IPv4 Prefix PDU and IPv6 Prefix PDU. All version 1 PDUs (including the IPv4 and IPv6 Prefix PDUs) may also be used without changes by version 2, and are transmitted with version number 1.

The format of the version 2 IPv4 Prefix PDU is as follows:



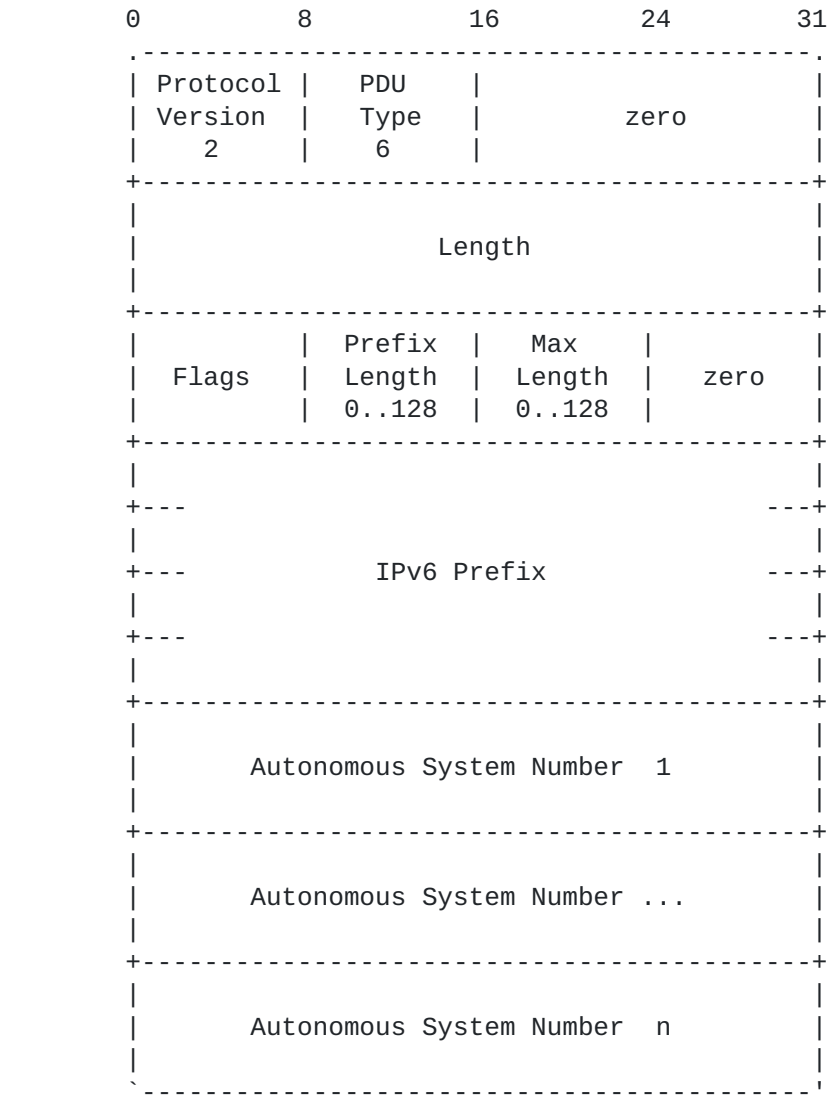
Version: 2

Length: the length of the PDU: 16 + 4 * the number of Autonomous System Numbers present.

Autonomous System Numbers: AS numbers allowed in the BGP AS path. See the section "path filter semantics" later in this document. At least one Autonomous System Number must be present.

All other fields are the same as in version 1.

The format of the version 2 IPv6 Prefix PDU is as follows:



Version: 2

Length: the length of the PDU: 28 + 4 * the number of Autonomous System Numbers present.

Autonomous System Numbers: AS numbers allowed in the BGP AS path. See the section "path filter semantics" later in this document. At least one Autonomous System Number must be present.

All other fields are the same as in version 1.

For the purposes of determining uniqueness of IPvX PDUs, only the fields {Prefix, Len, Max-Len, ASN} are considered, as per RFC 8210 [RFC8210], where ASN is the first Autonomous System Number in a version 2 IPvX PDU.

So for a given {Prefix, Len, Max-Len, ASN} either a version 1 IPvX PDU or a version 2 IPvX PDU may be transmitted, but not both. The relying party software generates a version 2 IPvX PDU when an id-pe-autonomousSysIdsPath with one or more ASes is present in a ROA and generates a version 1 IPvX PDU otherwise.

4. Path filter semantics

The order in which allowed ASes appear in a ROA is relevant. This order, and the order of the locally allowed ASes, is retained in the list of ASes sent to routers using the RPKI-router protocol version 2. A BGP AS path validates when all unique AS numbers in the path are present in the filter in the same order, ignoring AS numbers present in the filter that are missing from the BGP AS path.

If an AS path contains an AS_SET with more than one AS number in it and the implementation doesn't perform AS_SET sorting specified as an option in the BGP protocol specification [RFC4271] appendix F.4., then filtering behavior is undefined.

Example: the ROA for 192.0.2.0/24 lists the ASes 200 100. The local relying party software is configured to allow the transit AS 800 and the local AS 900. (The local AS normally doesn't appear in AS paths but may be prepended and SHOULD therefore be listed in the filter.) This results in the following sequence of Autonomous System Numbers in the RPKI-router protocol IPv4 Prefix PDU:

100 200 800 900

Conceptually, this results in the following regular expression BGP AS path filter:

^(900_)*(800_)*(200_)*(100_)+$

However, filtering can be performed more efficiently as shown in the example code in the appendix.

5. Internet exchange route servers

Many networks interconnect through internet exchanges. In many cases, rather than maintain a direct BGP neighbor relationship between the routers in both ASes, networks connected through an internet exchange interconnect through one or more route servers operated by the internet exchange.

As there are many internet exchanges throughout the world and connectivity is subject to change, it would be difficult to add all possible route servers ASes to ROAs. However, in practice this many not be an issue as many route servers don't include their own AS in the AS path.

6. IANA Considerations

This memo includes no request to IANA.

7. Security considerations

When two organizations that communicate over the internet both fully implement this specification, they have the ability to make sure to avoid paths containing ASes that neither of them has authorized to carry their their communication, as long as the BGP AS path attribute is an accurate reflection of the actual communication path.

In the absence of BGPsec [RFC8205], an AS that doesn't appear on the filter list may forge an AS path in order to reroute traffic through it. However, that AS must then transmit BGP updates with an AS path that doesn't match its own AS as configured by its neighbor. Some implementations check if the neighbor AS and the left most AS in AS paths are the same, as per the MAY in RFC 4271 [RFC4271] section 6.3. So these implementations would reject the forged AS path.

Another way to accomplish the same result of a valid looking BGP AS path but an invalid communication path would be for a malicious network to connect to other networks by presenting a falsified AS number when setting up a BGP session. It is unclear to what degree networks make sure the AS numbers that new customers or peers claim to hold are legitimate.

8. 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>.

[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>.

[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>.

[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>.

[RFC8210] Bush, R. and R. Austein, "The Resource Public Key Infrastructure (RPKI) to Router Protocol, Version 1", RFC 8210, DOI 10.17487/RFC8210, September 2017, <https://www.rfc-editor.org/info/rfc8210>.

Appendix A. Appendix: filter code example

```
#include <stdio.h>

struct pfxpdu
  {
    unsigned int version;
    unsigned int type;
    unsigned int length;
    unsigned int *path_filter;
  };

unsigned int path_filter[] = { 100, 200, 800, 900, 0 };
unsigned int as_path[]     = { 900, 900, 800, 300, 200, 100, 0 };

char *filter_as_path(struct pfxpdu *pfxpdu, unsigned int *as_path,
  int as_path_length)
{
  unsigned int path_filter_length = (pfxpdu->length - 16) / 4;
  int path_filter_index;
```

```
    int as_path_index;

    // if the path filter is empty we return status unknown
    if (path_filter_length < 1)
      return("unknown");

    // check the origin AS as per version 1 of the protocol
    if (as_path[as_path_length - 1] != pfxpdu->path_filter[0])
      return("invalid");

    // if the pfxpdu version == 2 then we check the entire path
    if (pfxpdu->version == 2)
      {
        path_filter_index = 0;
        for (as_path_index = as_path_length - 1; as_path_index >= 0;
          as_path_index--)
          {
            while (path_filter_index < path_filter_length &&
              as_path[as_path_index] !=
                pfxpdu->path_filter[path_filter_index])
              path_filter_index++;
            if (path_filter_index >= path_filter_length)
              return("invalid");
          }
      }
    return("valid");
  }

  unsigned int count_length(unsigned int *asns)
  {
    unsigned int length = 0;

    while (asns[length] != 0)
      length++;
    return length;
  }

  int main()
  {
    int i;
    int as_path_length;
    struct pfxpdu pfxpdu;

    as_path_length = count_length(as_path);

    pfxpdu.version = 2;
    pfxpdu.type = 4;
    pfxpdu.length = 16 + 4 * count_length(path_filter);
```

```
    pfxpdu.path_filter = path_filter;

    printf("Filter regexp: ^");
    for (i = (pfxpdu.length - 16) / 4 - 1; i > 0; i--)
      printf("(%d_)*", pfxpdu.path_filter[i]);
    printf("(%d_)+$\n", pfxpdu.path_filter[0]);

    printf("AS path:");
    for (i = 0; i < as_path_length; i++)
      printf(" %d", as_path[i]);
    printf("\n");

    printf("RPKI status: %s\n", filter_as_path(&pfxpdu, as_path,
      as_path_length));
  }
```

Author's Address

Iljitsch van Beijnum
BGPexpert.com

Email: iljitsch@muada.com