Source Address Validation Using BGP UPDATEs, ASPA, and ROA (BAR-SAV)


Kotikalapudi Sriram, Igor Lubashev, and Doug Montgomery

Email: ksriram@nist.gov  ilubashe@akamai.com  dougm@nist.gov

IETF 114 – OPSEC – July 2022
Much interest in the community to improve Source Address Validation (SAV) to reduce unauthorized source address spoofing:

• 2000 – BCP 38: “Networks shall filter out invalid traffic on ingress”

• 2004 – RFC 3704: “Networks can filter traffic on router interfaces using ...”
  - ACLs – Manually maintained, so unwieldy and gets stale fast
  - Strict RPF – Symmetrical routing only, so cannot be used by most non-trivial networks
  - Feasible Path RPF – No route filtering (announce all routes to everyone), so better but still no
  - Loose RPF – Lets everything but Martians though, so very ineffective

• 2020 – RFC 8704: “Better Feasible Path RPF using paths with common origin AS”
  - EFP-uRPF Alg A – At least one path w/ the same origin AS is Feasible for the Interface
    Can still block legitimate traffic if AS announce no prefixes to the ISP
    ... + ROA – Also use paths with the same origin AS learned from ROA (not just from BGP)
  - EFP-uRPF Alg B – At least one path w/ the same origin AS is Feasible for any Customer Interface
    Can permit too much, since Customers can spoof each other’s addresses

• 2022 – Savnet at IETF-113 and IETF-114; EFP-uRPF improvements (like BAR-SAV)
The Problem: RFC 8704 still blocks too much

- P2 is *not* detected by RFC 8704 Alg. A or Alg. B

Why is this so hard?

- We are trying to infer data-plane forwarding information from a BGP signal designed for a different purpose (reachability information).
- ROA and ASPA information, while also not designed for SAV purpose, can help a lot.
The Problem: CDN using DSR

Anycast/Edge Hybrid – Direct Server Return

1. Anycast POPs lookup “best” edge POP for each new connection (using the actual user IP)

2. Anycast POPs tunnel packets to edge POPs

3. Edge servers send data to users directly – Direct Server Return (DSR)
The Problem: Direct Server Return

Use Cases
- CDN Anycast/Edge Hybrid
- Mobile Roaming
- Low-Latency Gaming
- Traffic Scrubbing Center of a Security Provider
- Central Datacenter of a Multinational Corporation

Data packet: \{P3 \rightarrow P4\}

Data packet: \{P4 \rightarrow P3\}

Edge AS2

Home AS1

Service IP is in P3

“Home” POP

DSR POP

IP-over-IP tunneling \{P1 \rightarrow P2\}
The Problem: Some Stats

Percentage of networks doing SAV (by Akamai’s estimates):

- 2015 – 15%
- 2022 – 15%

Why?

- Could be economics, but even most smaller networks do not filter
- Likely SAV today is either ineffective or just breaks too many things
  - Is EFP-uRPF (RFC 8704) just too recent, and it will gain use in ~5 years? Or Alg. A is seen as too risky (blocks too much), and Alg. B is too loose?
BAR-SAV (BGP, ASPA, ROA - SAV)

• An improvement on EFP-uRPF Alg. A [RFC 8704]
  ➢ Improved BGP AS_PATH processing (make use of all ASes, not just origin AS)
  ➢ Makes complementary use of BGP UPDATEs, ASPAs, and ROAs

• BAR-SAV advances the technology for SAV filter design
  ✓ Significantly improves the ability to detect hidden prefixes
  ✓ Provides a solution to the CDN/Direct Server Return (DSR) problem

• No changes to any protocol on the wire
  ➢ Offers immediate benefits to early adopters
Goal: Construct Permissible Ingress Prefix List for SAV (at AS7)

The methodology is the same for a Customer or Lateral (i.e., non-transit) Peer* Interface

* See RFC 7908 for definitions of peering relations
SAV Using BGP UPDATE, ASPA, and ROA (BAR-SAV)
Construction of Permissible Ingress Prefix List for SAV (at AS7)

A. Iteratively obtain the set of ASNs in the Customer’s customer cone (CC) using “customer-of” and “previous-AS” relationships in ASPAs and AS_PATHs.

B. Gather all prefixes in ROAs associated with the ASNs found in Step A.

C. Gather all prefixes in BGP UPDATE messages with originating ASN among ASNs found in Step A.

D. Combine sets found in Steps B and C. Keep only the unique prefixes. This is the permissible prefix list for SAV for the interface in consideration.
The next 3 slides illustrate the details of how BAR-SAV works
How BAR-SAV Works
Finding All ASes and Prefixes in Customer’s (or Peer’s) Customer Cone Using BGP Announcements (as seen at AS4), ASPA, and ROA

**ASPs:**
- AS3 {AS4, AS9}
- AS5 {AS1}
- AS6 {AS1}
- AS8 {AS2}

**ROAs:**
- P2a AS2
- P5 AS5
- P6 AS6
- P8 AS8

**Interface in consideration**
- P6 [AS1 AS6]
- P7 [AS1 AS7]
- P2 [AS2] Community: AS9
- P2 [AS3 AS2]
- P6 [AS3 AS1 AS6]
- P7 [AS3 AS1 AS7]

**AS4**
- P2c or p2p or c2p
- P2 [AS9 AS3 AS2]
- ...
Finding **All** ASes in the CC using BGP AS_PATH and ASPA

**INPUTS**

<table>
<thead>
<tr>
<th>ASPAs:</th>
<th>ROAs:</th>
<th>BGP UPDATE AS_PATHs:</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS3 {AS4, AS9}</td>
<td>P2a AS2</td>
<td>Interface in Consideration: AS3</td>
</tr>
<tr>
<td>AS5 {AS1}</td>
<td>P5 AS5</td>
<td>P6 [AS3 AS1 AS6]</td>
</tr>
<tr>
<td>AS6 {AS1}</td>
<td>P6 AS6</td>
<td>P7 [AS3 AS1 AS7]</td>
</tr>
<tr>
<td>AS8 {AS2}</td>
<td>P8 AS8</td>
<td>Other Interfaces:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>P2 [AS9 AS3 AS2]</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Customer Cone</th>
<th>New ASes from ASPA</th>
<th>New ASes from AS_PATH</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AS3</td>
<td>None</td>
<td>P6 [AS3 AS1 AS6] → AS1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P7 [AS3 AS1 AS7] → AS1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>P2 [AS9 AS3 AS2] → AS2</td>
</tr>
<tr>
<td>2</td>
<td>AS3, AS1, AS2</td>
<td>AS5 {AS1} → AS5</td>
<td>P6 [AS3 AS1 AS6] → AS6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AS6 {AS1} → AS6</td>
<td>P7 [AS3 AS1 AS7] → AS7</td>
</tr>
<tr>
<td></td>
<td></td>
<td>AS8 {AS2} → AS8</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>AS3, AS1, AS2, AS5, AS6, AS8, AS7</td>
<td>None</td>
<td>None</td>
</tr>
</tbody>
</table>
Finding **All** Prefixes in the CC using BGP Routes and ROA

### INPUTS

**ASPAAs:**
- AS3 {AS4, AS9}
- AS5 {AS1}
- AS6 {AS1}
- AS8 {AS2}

**ROAs:**
- P2a AS2
- P5 AS5
- P6 AS6
- P8 AS8

**BGP UPDATE AS_PATHs:**
- Interface in Consideration: AS3
  - P6 [AS3 AS1 AS6]
  - P7 [AS3 AS1 AS7]

**Other Interfaces:**
- P2 [AS9 AS3 AS2]
- P6 [AS3 AS1 AS7]
- P7 [AS3 AS1 AS7]

### OUTPUT

- AS1, AS2, AS3, AS5, AS6, AS7, AS8

<table>
<thead>
<tr>
<th>ASN</th>
<th>Prefixes from ROA</th>
<th>Prefixes from BGP</th>
</tr>
</thead>
<tbody>
<tr>
<td>AS1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AS2</td>
<td>(P2a AS2) → P2a</td>
<td>P2 [AS9 AS3 AS2] → P2</td>
</tr>
<tr>
<td>AS3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AS5</td>
<td>(P5 AS5) → P5</td>
<td></td>
</tr>
<tr>
<td>AS6</td>
<td>(P6 AS6) → P6</td>
<td>P6 [AS3 AS1 AS6] → P6</td>
</tr>
<tr>
<td>AS7</td>
<td></td>
<td>P7 [AS3 AS1 AS7] → P7</td>
</tr>
<tr>
<td>AS8</td>
<td>(P8 AS8) → P8</td>
<td></td>
</tr>
</tbody>
</table>

**Customer Cone**

AS1, AS2, AS3, AS5, AS6, AS7, AS8

**SAV Prefixes**

P2, P2a, P5, P6, P7, P8
Refined Version of Algorithm A of EFP-uRPF [RFC 8704] 
Incorporated into BAR-SAV

- P2 is *not* detected by RFC 8704 Alg. A or Alg. B
- P2 *is* detected by BAR-SAV

```
AS3 -- P1 [AS3 AS2 AS1] --> AS4
    |                     p2p
    v                     ^
AS2 -- P2 [AS2] --> P1 [AS2 AS1]
    v
AS1 -- P1 [AS1]
```

EFP-uRPF = Enhanced Feasible Path uRPF

- Much better detection of “Hidden” prefixes in multihoming scenarios by BAR-SAV
Content Delivery Network (CDN) Application

Example of how the BAR-SAV method solves the CDN DSR blocking problem

Data packet: {P4 -> P3}

1. CDN owns P1, P2, P3. Creates ROAs:
   - {P1, P3} AS1
   - {P2, P3} AS2
   P2 [AS2]

2. IP-over-IP tunneling {P1 -> P2}
   P1 [AS1]
   P3 [AS1]

3. DSR POP
   P3 added in the ROA, but AS2 does not announce P3.

Data packet: {P3 -> P4}

BAR-SAV at AS9 permits P3!

CDN Anycast POP

CDN

P4

P3 is Anycast Prefix

AS1

AS2

P1

P2

P3

P3

15

P3 added in the ROA, but AS2 does not announce P3.
AS8 has an ASAP, but AS2 is not its provider.
- BAR-SAV refuses to infer AS8 as a customer of AS2 from BGP AS_PATH.
- Therefore, AS8 and AS5 are not in AS3’s Customer Cone
- Therefore, P5 is not in SAV list for AS3’s interface.

Note:
- Since the leaked route made it to AS4, BAR-SAV will infer that AS2 is a customer of AS9, and AS5 is a customer of AS8.
- Since ASPA deems the route “Invalid”, AS4 should not accept the leaked route for forwarding to P5.
Backup slides

Requirements for a Solution

- Improved fidelity – reduced improper block and improper permit
- Incrementally deployable – offers immediate benefits to early adopters
- Economical – benefits outweigh the costs (especially for early adopters)
- Network effect – late movers are feeling greater pressure to adopt
- Friendly to smaller networks – SAV is done best at the edge
SAV Using Only ASPA and ROA (Procedure X)
Construction of Permissible Ingress Prefix List for SAV (at AS7)

When ASPA and ROA adoption is ubiquitous (future)
Or if an ISP requires all its customers to register ROAs and ASPAs

A. Obtain the set of ASNs in the Customer’s customer cone (CC) using ASPAs by transitively discovering customers of the customer or lateral peer in consideration.
B. Gather all prefixes in ROAs associated with the ASNs found in Step A. Keep only the unique prefixes.
C. The set computed in Step B is the permissible prefix list for SAV for the interface in consideration.
A Note on Customer Cone Computation

- 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.
Detailed Procedure X
Creating the Permissible Prefix List for SAV for a Customer or Lateral Peer using only ASPA and ROA

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.

Note: Algorithm X is for future use when the deployment of ASPA and ROA is ubiquitous.
1. Let the Customer or Lateral Peer ASN be denoted as AS-k.
3. Increment $i$ to $i+1$.
4. Create AS-set $A(i)$ of all ASNs whose ASPA data declares at least one ASN in AS-set $Z(i-1)$ as a Provider.
5. Create AS-set $B(i)$ of all “non-ASPA” customer ASNs each of which is a customer of at least one ASN in AS-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-sets $A(i)$ and $B(i)$ and call it AS-set $C$.
   From AS-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-set $Z(i)$ is null, then set $i_{\text{max}} = i - 1$ and go to Step 8. Else, go to Step 3.
8. Form the union of the AS-sets, $Z(i)$, $i = 1, 2, ..., i_{\text{max}}$, and name this union as AS-set $D$.
9. Select all ROAs in which the authorized origin ASN is in AS-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-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.