ASPA: IETF109

Alexander Azimov, Yandex

a.e.azimov@gmail.com
ASPA Object Profile

• There MUST be single object for each (AS, AFI)!
• There SHOULD be single object for each (AS, AFI) per registry!
• All ASPA objects for an (AS, AFI) SHOULD be the same!
ASPA Pair Verification

- 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."
- If the set of Candidate Providers is empty, then the procedure exits with an outcome of "Unknown."
- If AS2 is included in the Candidate Providers, then the procedure exits with an outcome of "Valid."
- Otherwise, the procedure exits with an outcome of "Invalid."
ASPA Verification Procedure

- Beautiful ASCII drawings;
- Python code for verification procedures;
- Improved wording (special thanks to Jay Borkenhagen);
Terms

• Line goes up – route is announced from customer to provider;
• Line goes down – route is announced from provider to customer;
• Line goes straight – route is announced from peer to peer;
• The arrow shows the order of the ASPA check, not the route advertisement!
Upstream Path: Example

(1, 2), (2,3), (3,4) are Valid
The path is Valid

(1, 2) is Valid, (2, 3) is Invalid
The path is Invalid
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 \neq Seg(I).value and customer-provider verification procedure 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 \neq Seg(I).value and customer-provider verification procedure 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".
Downstream Paths: Example

(1, 2), (2, 3), (3, 4) are **Valid**
(4,5) is **Invalid**, but it’s **OK**!
(6,5), (7,6) are **Valid**
The path is **Valid**

(1, 2), (2, 3), (3, 4) are **Valid**
(4,5) is **Invalid**, but it’s **OK**!
(6,5) is **Valid**, (7,6) is **Invalid**
The path is **Invalid**
Downstream Paths: Formal Procedure

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 \( I_{\text{MIN}} \) as the minimal index for which \( \text{Seg}(I-1).\text{type} \) and \( \text{Seg}(I).\text{type} \) equal to AS_SEQUENCE, its values aren't equal and the verification procedure for \( (\text{Seg}(I-1).\text{value}, \text{Seg}(I).\text{value}, \text{AFI}) \) returns "Invalid".

4. If \( I_{\text{MIN}} \) doesn't exist put the length of AS_PATH in \( I_{\text{MIN}} \) variable and jump to 5.

5. If there exists \( J > I_{\text{MIN}} \) such that both \( \text{Seg}(J-1).\text{type}, \text{Seg}(J).\text{type} \) equal to AS_SEQUENCE, \( \text{Seg}(J-1).\text{value} \neq \text{Seg}(J).\text{value} \) and the customer-provider verification procedure returns "Invalid" for \( (\text{Seg}(J).\text{value}, \text{Seg}(J-1).\text{value}, \text{AFI}) \), then the procedure halts with the outcome "Invalid";

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

7. If there exists \( J > I_{\text{MIN}} \) such that both \( \text{Seg}(J-1).\text{type}, \text{Seg}(J).\text{type} \) equal to AS_SEQUENCE, \( \text{Seg}(J-1).\text{value} \neq \text{Seg}(J).\text{value} \) and the customer-provider verification procedure returns "Unknown" for \( (\text{Seg}(J).\text{value}, \text{Seg}(J-1).\text{value}, \text{AFI}) \), then the procedure halts with the outcome "Unknown";

8. If there exists \( I_{\text{MIN}} > J \) such that both \( \text{Seg}(J-1).\text{type}, \text{Seg}(J).\text{type} \) equal to AS_SEQUENCE, \( \text{Seg}(J-1).\text{value} \neq \text{Seg}(J).\text{value} \) and the customer-provider verification procedure returns "Unknown" for \( (\text{Seg}(J-1).\text{value}, \text{Seg}(J).\text{value}, \text{AFI}) \), then the procedure halts with the outcome "Unknown";

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

• Is WG happy with ASPA algorithm?
• Is WG happy with its formal language?
• Does Python code improve readability?
• Is WG willing to start WGLC?