Source Address Validation Table Abstraction and Application

draft-huang-savnet-sav-table-00

M. Huang, T. Zhou, N. Geng, D. Li, L. Chen, J. Wu

November 2022

Motivation

□ SAV tables on routers can be generated or implemented differently

□ It is important to learn how a typical SAV table looks and how to properly use one

□ However, existing SAV mechanisms

a) have core data structures coupled with implementation

- ➢ **not easy to do analysis**

b) have no unified data structure of SAV table, which is suitable to any scenarios

- ➢ **not easy to know which kind of SAV tables** can be generated and enabled in data plane

c) usually take either "permit" action or "block" action

- ➢ sometimes **not flexible enough** for diversified operation requirements in practice

About the Draft

□ Main content:

- ◆ An SAV table **abstraction** which can express any existing SAV tables
- ◆ Four typical validation **modes** with application scenarios/conditions
- ◆ Multiple **actions** for diversified operation requirements

□ Usage

- ◆ **Help clarify** the design goals of SAV mechanisms
- ◆ **Provide guidance** to operators on the choice of SAV table modes and SAV mechanisms

* Notes: How to generate and implement SAV tables is not in the scope of the draft

SAV Table Abstraction

- □ Key observation: For any SAV tables, the basic idea of SAV is to check whether a source prefix arrives from a valid interface.
- □ SAV table abstraction: 1) two dimensions, i.e., source prefix and interface; 2) each cell indicates the validity state

Interfaces under consideration

```
+-------------------------------------------------------+
+  Source prefix  |  Intf 1  |  Intf 2  |  Intf 3  | ... +
+-------------------------------------------------------+
+  P1             | state_11 | state_12 | state_13 | ... +
+  P2             | state_21 | state_22 | state_23 | ... +
+  P3             | state_31 | state_32 | state_33 | ... +
+  ...            |   ...    |   ...    |   ...    | ... +
+  Pn             | state_n1 | state_n2 | state_n3 | ... +
+  default        | state_*1 | state_*2 | state_*3 | ... +
+-------------------------------------------------------+
*state: valid, invalid, or unknown
```

source prefix

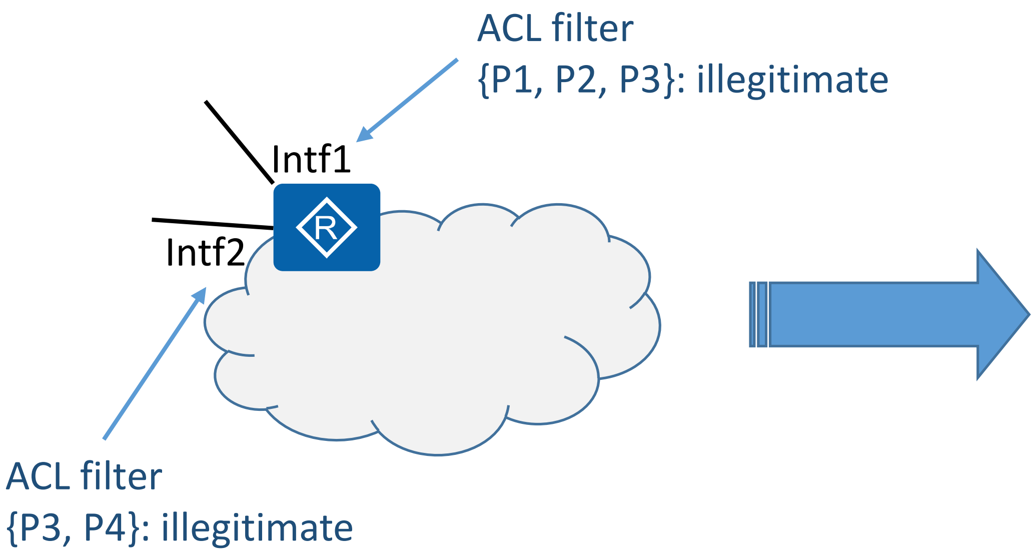
The task is to fill each cell. The more complete the better. The more accurate the better.

The row of "default prefix" is usually filled by manual configuration.

The prefixes not known by the SAV table

Example: An SAV Table of ACL Ingress Filtering

- Left: an application of ACL ingress filtering
- Right: the expression in the unified SAV table



Source prefix	Intf 1	Intf 2
P1	invalid	valid
P2	invalid	valid
P3	invalid	invalid
P4	valid	invalid
default	valid	valid

*state: valid, invalid, or unknown

* Another example "An SAV Table of Strict uRPF" can be found in backup slides

Validation Modes

□ What are modes?

- ◆ Modes are typical **validation process** for the SAV table abstraction

□ Why need modes?

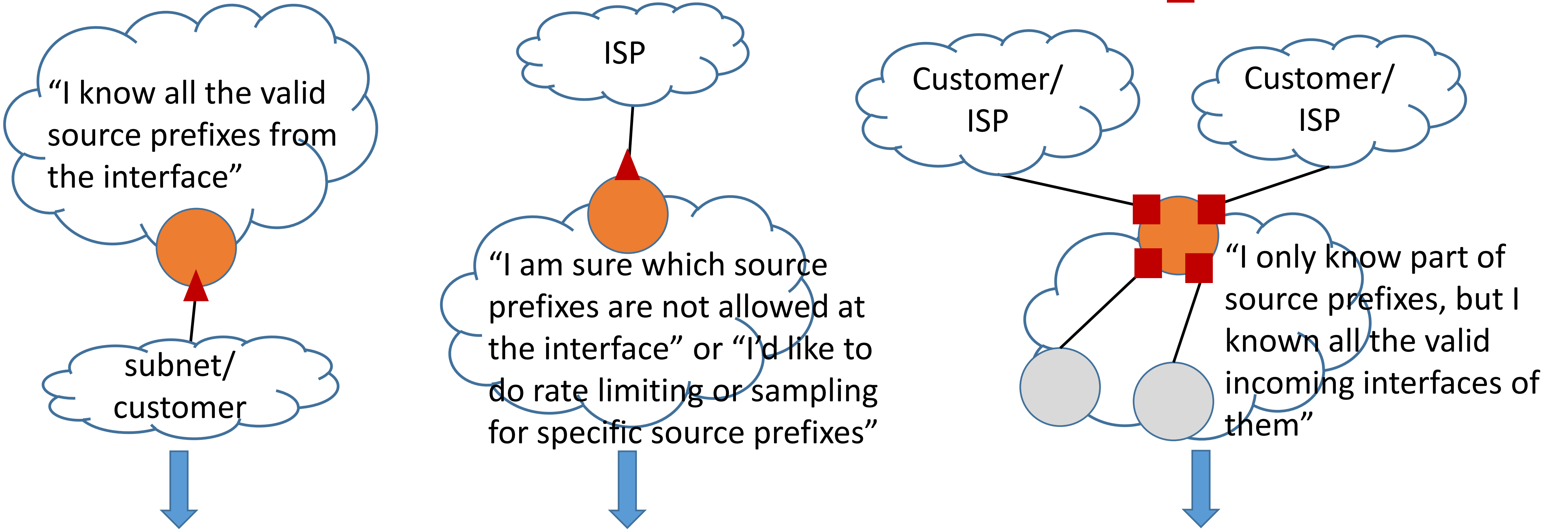
- ◆ **The accuracy and strictness of SAV tables varies under different application scenarios**
- ◆ **Modes help easily express or agree on important questions such as which kind of SAV tables can be generated and enabled in the data plane**

Four Validation Modes

● Validation Router

▲ Interface-based validation

■ Prefix-based validation



"I know all the valid source prefixes from the interface"

"I am sure which source prefixes are not allowed at the interface" or "I'd like to do rate limiting or sampling for specific source prefixes"

"I only know part of source prefixes, but I known all the valid incoming interfaces of them"

Mode 1: Interface-based prefix allowlist

*Strict ingress filtering

Mode 2: Interface-based prefix blocklist

*Proactive filtering or reactive filtering

Mode 3 (or 4): Prefix-based interface allowlist (or blocklist)

*Focus on protecting specific source prefixes

A Brief Summary of the Four Modes

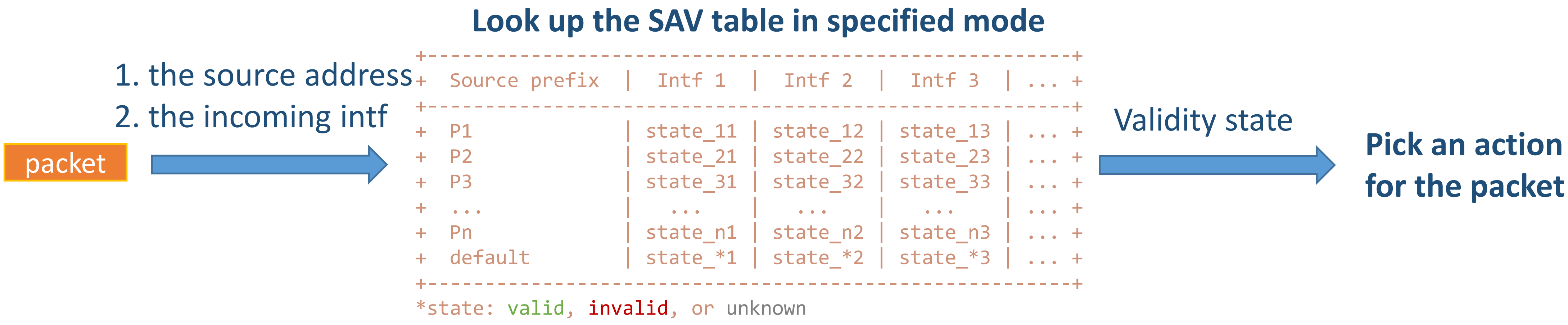
Mode	Description	Application Scenario	Relationship
Mode 1: Interface-based prefix allowlist	For an interface, only the listed prefixes are valid	Only when the complete set of valid source prefixes is known by the interface	The two modes are complementary for the IP address space
Mode 2: Interface-based prefix blocklist	For an interface, only the listed prefixes are invalid	Proactive filtering and reactive filtering	
Mode 3: Prefix-based interface allowlist	For a prefix, only the listed interfaces are valid	Focus on protecting specific source prefixes	The two modes are complementary for the set of interfaces
Mode 4: Prefix-based interface blocklist	For a prefix, only the listed interfaces are invalid	Focus on protecting specific source prefixes	

Choose suitable modes for different scenarios to make as much protection as possible

* More details in backup slides

Validation Procedure

- □ Step 1: look up the SAV table to get the validity state of the packet
- □ Step 2: get the action for the packet according to the validity state



Look up the SAV table in specified mode

1. the source address
2. the incoming intf

packet

```
+----------------------------------------------------------+
+  Source prefix  |  Intf 1  |  Intf 2  |  Intf 3  | ... +
+----------------------------------------------------------+
+  P1             | state_11 | state_12 | state_13 | ... +
+  P2             | state_21 | state_22 | state_23 | ... +
+  P3             | state_31 | state_32 | state_33 | ... +
+  ...            |   ...    |   ...    |   ...    | ... +
+  Pn             | state_n1 | state_n2 | state_n3 | ... +
+  default        | state_*1 | state_*2 | state_*3 | ... +
+----------------------------------------------------------+
*state: valid, invalid, or unknown
```

Validity state

Pick an action for the packet

Notes: For an interface, only when SAV is enabled on the interface, the packets arriving at this interface will be validated.

Available Actions

- Actions for packets
  - Permit action: forward the packet normally
  - Block action: drop the packet directly
  - Rate limiting action: enforces an upper bound of traffic rate
  - Sampling action: capture the packet and report it to remote servers
  - etc.

```
+-----------------------------------------------------------+
+ Validity | Available Action             | Optional Action +
+-----------------------------------------------------------+
+ valid    | permit                       | sampling        +
+ invalid  | permit, block, rate limiting | sampling        +
+ unknown  | permit, block, rate limiting | sampling        +
+-----------------------------------------------------------+
```

- Why multiple actions available?
  - **Meet diversified operation requirements**

Conclusion

□ Main content:

- ◆ An SAV table abstraction which can express any existing SAV tables
- ◆ Four typical validation modes with application scenarios/conditions
- ◆ Multiple actions for diversified operation requirements

□ Usage

- ◆ Help clarify the design goals of SAV mechanisms
- ◆ Provide guidance to operators on the choice of SAV table modes and SAV mechanisms

□ What is out of scope?

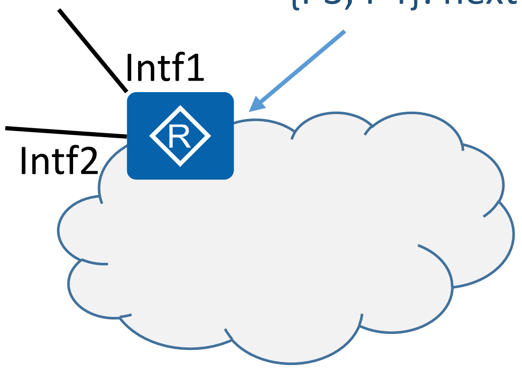
- ◆ Do not focus on how to generate and implement SAV tables

Thanks!

Backup Slides

Example: An SAV Table of Strict uRPF

- Left: an application of strict uRPF
- Right: the expression in the unified SAV table



FIB info
{P1, P2, P3}: next-hop intf2
{P3, P4}: next-hop intf1

```
+---------------------------------------+
+  Source prefix  |  Intf 1  |  Intf 2  +
+---------------------------------------+
+  P1             |  invalid |  valid   +
+  P2             |  invalid |  valid   +
+  P3             |  valid   |  valid   +
+  P4             |  valid   |  invalid +
+  default        |  unknown |  unknown +
+---------------------------------------+
*state: valid, invalid, or unknown
```

May depend on configuration.

Mode 1: Interface-based prefix allowlist

- Mode 1 is an interface-scale mode
- It indicates which set of source prefixes are valid for interface X, and any other source prefixes will all be considered as invalid

```
+----------------------------+
+   Source prefix  |  Intf X  +
+----------------------------+
+   P1             |  valid   +
+   P2             |  valid   +
+   P3             |  valid   +
+   ...            |  valid   +
+   Pn             |  valid   +
+   default        |  invalid +
+----------------------------+
```

A column of the SAV table abstraction can be easily converted to the form of Mode 1 table

- When to use Mode 1?
  - Require to known the complete set of legitimate prefixes connected to the interface.
  - Potential scenarios: the interface connecting to a subnet, a stub AS, or a customer cone.

Mode 2: Interface-based prefix blocklist

□ Mode 2 is also an interface-scale mode

□ It indicates which set of source prefixes are invalid for interface X, and any other source prefixes will all be considered as valid

```
+----------------------------+
+  Source prefix  |  Intf X  +
+----------------------------+
+  P1             |  invalid +
+  P2             |  invalid +
+  P3             |  invalid +
+  ...            |  invalid +
+  Pn             |  invalid +
+  default        |  valid   +
+----------------------------+
```

A column of the SAV table abstraction can be easily converted to the form of Mode 2 table

□ When to use Mode 2?

◆ Does not require the complete blocklist. Need known which source prefixes are sure to be invalid.

◆ Potential scenarios: proactive filtering and reactive filtering (e.g., DDoS elimination)

Mode 3: Prefix-based interface allowlist

- □ Mode 3 is an device-scale mode
- □ It indicates the set of valid incoming interfaces of each source prefix, and the default prefix from any interfaces will all be considered as unknown

```
+--------------------------------------+
+  Source prefix  |  Intf 4  | others  +
+--------------------------------------+
+  P1             |  valid   | invalid +
+--------------------------------------+
+--------------------------------------------------+
+  Source prefix  |  Intf 1  |  Intf 2  |  others  +
+--------------------------------------------------+
+  P2             |  valid   |  valid   |  invalid +
+--------------------------------------------------+
... ...
+---------------------------+
+  Source prefix  |  any    +
+---------------------------+
+  default        | unknown +
+---------------------------+
```

- □ When to use Mode 3?
  - ◆ Focuses on protecting specific source prefixes
  - ◆ When Mode 1 cannot be enabled, Mode 3 can still provide some extent of protection

Mode 4: Prefix-based interface blocklist

- Mode 4 is also an device-scale mode
- It indicates the set of invalid incoming interfaces of each source prefix, and the default prefix from any interfaces will all be considered as unknown

Source prefix	Intf 4	others
P1	valid	invalid

Source prefix	Intf 1	Intf 2	others
P2	valid	valid	invalid

... ...

Source prefix	any
default	unknown

- When to use Mode 4?
  - Focuses on protecting specific source prefixes
  - When Mode 1 cannot be enabled, Mode 4 can still provide some extent of protection