Internet Engineering Task Force (IETF)
Request for Comments: 6620
Category: Standards Track
ISSN: 2070-1721

E. Nordmark
Cisco Systems
M. Bagnulo
UC3M
E. Levy-Abegnoli
Cisco Systems

May 2012

FCFS SAVI: First-Come, First-Served Source Address Validation Improvement for Locally Assigned IPv6 Addresses

Abstract

This memo describes First-Come, First-Served Source Address Validation Improvement (FCFS SAVI), a mechanism that provides source address validation for IPv6 networks using the FCFS principle. The proposed mechanism is intended to complement ingress filtering techniques to help detect and prevent source address spoofing.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This memo describes FCFS SAVI, a mechanism that provides source address validation for IPv6 networks using the FCFS principle. The proposed mechanism is intended to complement ingress filtering techniques to help detect and prevent source address spoofing. Section 2 gives the background and description of FCFS SAVI, and Section 3 specifies the FCFS SAVI protocol.

1.1. Terminology

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. Background to FCFS SAVI

2.1. Scope of FCFS SAVI

The application scenario for FCFS SAVI is limited to the local link. Hence, the goal of FCFS SAVI is to verify that the source address of the packets generated by the hosts attached to the local link have not been spoofed.

In a link, hosts and routers are usually attached. Hosts generate packets with their own address as the source address. This is called "local traffic". Routers send packets containing a source IP address other than their own, since they are forwarding packets generated by other hosts (usually located in a different link). This is called "transit traffic".

The applicability of FCFS SAVI is limited to the local traffic, i.e., to verify if the traffic generated by the hosts attached to the local link contains a valid source address. The verification of the source address of the transit traffic is out of the scope of FCFS SAVI. Other techniques, like ingress filtering [RFC2827], are recommended to validate transit traffic. In that sense, FCFS SAVI complements ingress filtering, since it relies on ingress filtering to validate transit traffic, but it provides validation of local traffic, which is not provided by ingress filtering. Hence, the security level is increased by using these two techniques.

In addition, FCFS SAVI is designed to be used with locally assigned IPv6 addresses, in particular with IPv6 addresses configured through Stateless Address Autoconfiguration (SLAAC) [RFC4862]. Manually configured IPv6 addresses can be supported by FCFS SAVI, but manual configuration of the binding on the FCFS SAVI device provides higher security and seems compatible with manual address management. FCFS

SAVI can also be used with IPv6 addresses assigned via DHCPv6, since they ought to perform the Duplicate Address Detection (DAD) procedure, but there is a specific mechanism tailored for dealing with DHCP-assigned addresses defined in [SAVI-DHCP]. Additional considerations about how to use FCFS SAVI depending on the type of address management used and the nature of the addresses are discussed in the framework document [SAVI-FRAMEWORK].

2.2. Constraints for FCFS SAVI Design

FCFS SAVI is designed to be deployed in existing networks requiring a minimum set of changes. For that reason, FCFS SAVI does not require any changes in the host whose source address is to be verified. Any verification solely relies on the usage of already available protocols. That is, FCFS SAVI does not define a new protocol, define any new message on existing protocols, or require that a host use an existent protocol message in a different way. In other words, no host changes are required.

FCFS SAVI validation is performed by the FCFS SAVI function. The function can be placed in different types of devices, including a router or a Layer 2 (L2) bridge. The basic idea is that the FCFS SAVI function is located in the points of the topology that can enforce the correct usage of the source address by dropping the non-compliant packets.

2.3. Address Ownership Proof

The main function performed by FCFS SAVI is to verify that the source address used in data packets actually belongs to the originator of the packet. Since the FCFS SAVI scope is limited to the local link, the originator of the packet is attached to the local link. In order to define a source address validation solution, we need to define the meaning of "address ownership", i.e., what it means that a given host owns a given address in the sense that the host is entitled to send packets with that source address. With that definition, we can define how a device can confirm that the source address in a datagram is owned by the originator of the datagram.

In FCFS SAVI, proof of address ownership is based on the First-Come, First-Served principle. The first host that claims a given source address is the owner of the address until further notice. Since no host changes are acceptable, we need to find the means to confirm address ownership without requiring a new protocol. So, whenever a source address is used for the first time, a state is created in the device that is performing the FCFS SAVI function binding the source address to a binding anchor that consists of Layer 2 information that the FCFS SAVI box has available (e.g., the port in a switched LAN).

Subsequent data packets containing that IP source address can be checked against the same binding anchor to confirm that the originator owns the source IP address.

There are, however, additional considerations to be taken into account. For instance, consider the case of a host that moves from one segment of a LAN to another segment of the same subnetwork and keeps the same IP address. In this case, the host is still the owner of the IP address, but the associated binding anchor may have changed. In order to cope with this case, the defined FCFS SAVI behavior implies verification of whether or not the host is still reachable using the previous binding anchor. In order to do that, FCFS SAVI uses the Neighbor Discovery (ND) protocol. If the host is no longer reachable at the previously recorded binding anchor, FCFS SAVI assumes that the new location is valid and creates a new binding using the new binding anchor. In case the host is still reachable using the previously recorded binding anchor, the packets coming from the new binding anchor are dropped.

Note that this only applies to local traffic. Transit traffic generated by a router would be verified using alternative techniques, such as ingress filtering. FCFS SAVI checks would not be fulfilled by the transit traffic, since the router is not the owner of the source address contained in the packets.

2.4. Binding Anchor Considerations

Any SAVI solution is not stronger than the binding anchor it uses. If the binding anchor is easily spoofable (e.g., a Media Access Control (MAC) address), then the resulting solution will be weak. The treatment of non-compliant packets needs to be tuned accordingly. In particular, if the binding anchor is easily spoofable and the FCFS SAVI device is configured to drop non-compliant packets, then the usage of FCFS SAVI may open a new vector of Denial-of-Service (DoS) attacks, based on spoofed binding anchors. For that reason, in this specification, only switch ports MUST be used as binding anchors. Other forms of binding anchors are out of the scope of this specification, and proper analysis of the implications of using them, should be performed before their usage.

2.5. FCFS SAVI Protection Perimeter

FCFS SAVI provides perimetrical security. FCFS SAVI devices form what can be called an FCFS SAVI protection perimeter, and they verify that any packet that crosses the perimeter is compliant (i.e., the source address is validated). Once the packet is inside the perimeter, no further validations are performed on the packet. This model has implications both on how FCFS SAVI devices are deployed in the topology and on the configuration of the FCFS SAVI boxes.

The implication of this perimetrical security approach is that there is part of the topology that is inside the perimeter and part of the topology that is outside the perimeter. So, while packets coming from interfaces connected to the external part of the topology need to be validated by the FCFS SAVI device, packets coming from interfaces connected to the internal part of the topology do not need to be validated. This significantly reduces the processing requirements of the FCFS SAVI device. It also implies that each FCFS SAVI device that is part of the perimeter must be able to verify the source addresses of the packets coming from the interfaces connected to the external part of the perimeter. In order to do so, the FCFS SAVI device binds the source address to a binding anchor.

One possible approach would be for every FCFS SAVI device to store binding information about every source address in the subnetwork. In this case, every FCFS SAVI device would store a binding for each source address of the local link. The problem with this approach is that it imposes a significant memory burden on the FCFS SAVI devices. In order to reduce the memory requirements imposed on each device, the FCFS SAVI solution described in this specification distributes the storage of FCFS SAVI binding information among the multiple FCFS SAVI devices of a subnetwork. The FCFS SAVI binding state is distributed across the FCFS SAVI devices according to the following criterion: each FCFS SAVI device only stores binding information about the source addresses bound to anchors corresponding to the interfaces that connect to the part of the topology that is outside of the FCFS SAVI protection perimeter. Since all the untrusted packet sources are by definition in the external part of the perimeter, packets generated by each of the untrusted sources will reach the perimeter through an interface of an FCFS SAVI device. The binding information for that particular source address will be stored in the first FCFS SAVI device the packet reaches.

The result is that the FCFS SAVI binding information will be distributed across multiple devices. In order to provide proper source address validation, it is critical that the information distributed among the different FCFS SAVI devices be coherent. In particular, it is important to avoid having the same source address bound to different binding anchors in different FCFS SAVI devices. Should that occur, then it would mean that two hosts are allowed to send packets with the same source address, which is what FCFS SAVI is trying to prevent. In order to preserve the coherency of the FCFS SAVI bindings distributed among the FCFS SAVI devices within a realm, the Neighbor Discovery (ND) protocol [RFC4861] is used, in particular the Neighbor Solicitation (NS) and Neighbor Advertisement (NA) messages. Following is a simplified example of how this might work. Before creating an FCFS SAVI binding in the local FCFS SAVI database, the FCFS SAVI device will send an NS message querying for the address involved. Should any host reply to that message with an NA message, the FCFS SAVI device that sent the NS will infer that a binding for that address exists in another FCFS SAVI device and will not create a local binding for it. If no NA message is received as a reply to the NS, then the local FCFS SAVI device will infer that no binding for that address exists in other FCFS SAVI device and will create the local FCFS SAVI binding for that address.

To summarize, the proposed FCFS SAVI approach relies on the following design choices:

- o An FCFS SAVI provides perimetrical security, so some interfaces of an FCFS SAVI device will connect to the internal (trusted) part of the topology, and other interfaces will connect to the external (untrusted) part of the topology.

- o An FCFS SAVI device only verifies packets coming through an interface connected to the untrusted part of the topology.

- o An FCFS SAVI device only stores binding information for the source addresses that are bound to binding anchors that correspond to interfaces that connect to the untrusted part of the topology.

- o An FCFS SAVI uses NS and NA messages to preserve the coherency of the FCFS SAVI binding state distributed among the FCFS SAVI devices within a realm.

So, in a link that is constituted of multiple L2 devices, some of which are FCFS SAVI capable and some of which are not, the FCFS-SAVI-capable devices MUST be deployed forming a connected perimeter (i.e., no data packet can get inside the perimeter without passing through an FCFS SAVI device). Packets that cross the perimeter will be validated while packets that do not cross the perimeter are not validated (hence, FCFS SAVI protection is not provided for these packets). Consider the deployment of FCFS SAVI in the topology depicted in the following figure:

```
                                                +--------+
      +--+   +--+                          +--+ | +--+   |
      |H1|   |H2|                          |H3| | |R1|   |
      +--+   +--+                          +--+ | +--+   |
        |     |                              |  |  |     |
   +-------------SAVI-PROTECTION-PERIMETER------+  |     |
   |    |     |                              |     |     |
   |  +-1-----2-+                          +-1-----2-+   |
   |  |  SAVI1  |                          |  SAVI2  |   |
   |  +-3--4----+                          +--3------+   |
   |    |  |          +--------------+        |          |
   |    |  +----------|              |---------+          |
   |    |             |   SWITCH-A   |                    |
   |    |  +----------|              |---------+          |
   |    |  |          +--------------+        |          |
   |  +-1--2----+                          +--1------+   |
   |  |  SAVI3  |                          |  SAVI4  |   |
   |  +-3-----4-+                          +----4----+   |
   |    |     |                                 |        |
   |      +------SAVI-PROTECTION-PERIMETER---------------+
   |    | |   |                                 |
   |  +--+|  +--+                            +---------+
   |  |R2||  |H4|                            |SWITCH-B |
   |  +--+|  +--+                            +---------+
   +------+                                    |    |
                                             +--+  +--+
                                             |H5|  |H6|
                                             +--+  +--+
```

Figure 1: SAVI Protection Perimeter

In Figure 1, the FCFS SAVI protection perimeter is provided by four FCFS SAVI devices, namely SAVI1, SAVI2, SAVI3, and SAVI4. These devices verify the source address and filter packets accordingly.

FCFS SAVI devices then have two types of ports: Trusted Ports and Validating Ports.

- Validating Ports (VPs) are those in which FCFS SAVI processing is performed. When a packet is received through one of the Validating Ports, FCFS SAVI processing and filtering will be executed.

- Trusted Ports (TPs) are those in which FCFS SAVI processing is not performed. So, packets received through Trusted Ports are not validated, and no FCFS SAVI processing is performed on them.

Trusted Ports are used for connections with trusted infrastructure, including the communication between FCFS SAVI devices, the communication with routers, and the communication of other switches that, while not FCFS SAVI devices, only connect to trusted infrastructure (i.e., other FCFS SAVI devices, routers, or other trusted nodes). So, in Figure 1, Port 3 of SAVI1 and Port 1 of SAVI3 are trusted because they connect two FCFS SAVI devices. Port 4 of SAVI1, Port 3 of SAVI2, Port 2 of SAVI3, and Port 1 of SAVI4 are trusted because they connect to SWITCH-A, to which only trusted nodes are connected. In Figure 1, Port 2 of SAVI2 and Port 3 of SAVI3 are Trusted Ports because they connect to routers.

Validating Ports are used for connection with non-trusted infrastructure. In particular, hosts are normally connected to Validating Ports. Non-SAVI switches that are outside of the FCFS SAVI protection perimeter also are connected through Validating Ports. In particular, non-SAVI devices that connect directly to hosts or that have no SAVI-capable device between themselves and the hosts are connected through a Validating Port. So, in Figure 1, Ports 1 and 2 of SAVI1, Port 1 of SAVI2, and Port 4 of SAVI 3 are Validating Ports because they connect to hosts. Port 4 of SAVI4 is also a Validating Port because it is connected to SWITCH-B, which is a non-SAVI-capable switch that is connected to hosts H5 and H6.

2.6. Special Cases

Multi-subnet links: In some cases, a given subnet may have several prefixes. This is directly supported by SAVI as any port can support multiple prefixes. Forwarding of packets between different prefixes involving a router is even supported, as long as the router is connected to a Trusted Port, as recommended for all the routers.

Multihomed hosts: A multihomed host is a host with multiple interfaces. The interaction between SAVI and multihomed hosts is as follows. If the different interfaces of the host are assigned different IP addresses and packets sent from each interface always carry the address assigned to that interface as the source address, then from the perspective of a SAVI device, this is equivalent to two hosts with a single interface, each with an IP address. This is supported by SAVI without the need for additional considerations. If the different interfaces share the same IP address or if the interfaces have different addresses but the host sends packets using the address of one of the interfaces through any of the interfaces, then SAVI does not directly support it. It would require either connecting at least one interface of the multihomed host to a Trusted Port or manually configuring the SAVI bindings to allow binding the address of the multihomed host to multiple anchors simultaneously.

Untrusted routers: One can envision scenarios where routers are dynamically attached to an FCFS SAVI network. A typical example would be a mobile phone connecting to an FCFS SAVI switch where the mobile phone is acting as a router for other personal devices that are accessing the network through it. In this case, the router does not seem to directly fall in the category of trusted infrastructure (if this was the case, it is likely that all devices would be trusted); hence, it cannot be connected to a Trusted Port and if it is connected to a Validating Port, the FCFS SAVI switch would discard all the packets containing an off-link source address coming from that device. As a result, the default recommendation specified in this specification does not support such a scenario.

3. FCFS SAVI Specification

3.1. FCFS SAVI Data Structures

The FCFS SAVI function relies on state information binding the source address used in data packets to the binding anchor that contained the first packet that used that source IP address. Such information is stored in an FCFS SAVI database (DB). The FCFS SAVI DB will contain a set of entries about the currently used IP source addresses. Each entry will contain the following information:

- o IP source address
- o Binding anchor: port through which the packet was received
- o Lifetime
- o Status: either TENTATIVE, VALID, TESTING_VP, or TESTING_TP-LT
- o Creation time: the value of the local clock when the entry was firstly created

In addition, FCFS SAVI needs to know what prefixes are directly connected, so it maintains a data structure called the FCFS SAVI Prefix List, which contains:

- Prefix

- Interface where prefix is directly connected

3.2. FCFS SAVI Algorithm

3.2.1. Discovering On-Link Prefixes

In order to distinguish local traffic from transit traffic, the FCFS SAVI device relies on the FCFS SAVI Prefix List, which contains the set of on-link IPv6 prefixes. An FCFS SAVI device MUST support the following two methods for populating the Prefix List: manual configuration and Router Advertisement, as detailed next.

Manual configuration: An FCFS SAVI device MUST support manual configuration of the on-link prefixes included in the Prefix List. For example, this can be used when there are no prefixes being advertised on the link.

Router Advertisement: An FCFS SAVI device MUST support discovery of on-link prefixes through Router Advertisement messages in Trusted Ports. For Trusted Ports, the FCFS SAVI device will learn the on-link prefixes following the procedure defined for a host to process the Prefix Information options described in Section 6.3.4 of [RFC4861] with the difference that the prefixes will be configured in the FCFS SAVI Prefix List rather than in the ND Prefix List. In addition, when the FCFS SAVI device boots, it MUST send a Router Solicitation message as described in Section 6.3.7 of [RFC4861], using the unspecified source address.

3.2.2. Processing of Transit Traffic

The FCFS SAVI function is located in a forwarding device, such as a router or a Layer 2 switch. The following processing is performed depending on the type of port through which the packet has been received:

- If the data packet is received through a Trusted Port, the data packet is forwarded, and no SAVI processing performed on the packet.

- If the data packet is received through a Validating Port, then the FCFS SAVI function checks whether the received data packet is local traffic or transit traffic. It does so by verifying if the source address of the packet belongs to one of the directly connected prefixes available in the receiving interface. It does so by searching the FCFS SAVI Prefix List.

   * If the IP source address does not belong to one of the on-link prefixes of the receiving interface, the data packet is transit traffic, and the packet SHOULD be discarded. (If for some reason, discarding the packets is not acceptable, logging or triggering of alarms MAY be used). The FCFS SAVI function MAY send an ICMP Destination Unreachable Error back to the source address of the data packet, and ICMPv6, code 5 (Source address failed ingress/egress policy), should be used.

   * If the source address of the packet does belong to one of the prefixes available in the receiving port, then the FCFS SAVI local traffic validation process is executed as described below.

   * If the source address of the packet is an unspecified address, the packet is forwarded, and no SAVI processing is performed except for the case of the Neighbor Solicitation messages involved in the Duplicate Address Detection, which are treated as described in Section 3.2.3.

3.2.3. Processing of Local Traffic

We next describe how local traffic, including both control and data packets, is processed by the FCFS SAVI device using a state machine approach.

The state machine described is for the binding of a given source IP address (called IPAddr) in a given FCFS SAVI device. This means that all the packets described as inputs in the state machine above refer to that given IP address. In the case of data packets, the source address of the packet is IPAddr. In the case of the DAD_NS packets, the Target Address is IPAddr. The key attribute is the IP address. The full state information is as follows:

o IP ADDRESS: IPAddr

o BINDING ANCHOR: P

o LIFETIME: LT

The possible states are as follows:

o NO_BIND

o TENTATIVE

o VALID

- TESTING_TP-LT
- TESTING_VP

We will use VP for Validating Port and TP for Trusted Port.

After bootstrapping (when no binding exists), the state for all source IP addresses is NO-BIND, i.e., there is no binding for the IP address to any binding anchor.

NO_BIND: The binding for a source IP address entry is in this state when it does not have any binding to an anchor. All addresses are in this state by default after bootstrapping, unless bindings were created for them.

TENTATIVE: The binding for a source address for which a data packet or an NS generated by the Duplicate Address Detection (DAD) procedure has been received is in this state during the waiting period during which the DAD procedure is being executed (either by the host itself or the FCFS SAVI device on its behalf).

VALID: The binding for the source address is in this state after it has been verified. It means that it is valid and usable for filtering traffic.

TESTING_TP-LT: A binding for a source address enters this state due to one of two reasons:

- When a Duplicate Address Detection Neighbor Solicitation has been received through a Trusted Port. This implies that a host is performing the DAD procedure for that source address in another switch. This may be due to an attack or to the fact that the host may have moved. The binding in this state is then being tested to determine which is the situation.
- The lifetime of the binding entry is about to expire. This is due to the fact that no packets have been seen by the FCFS SAVI device for the LIFETIME period. This may be due to the host simply being silent or because the host has left the location. In order to determine which is the case, a test is performed to determine if the binding information should be discarded.

TESTING_VP: A binding for a source address enters this state when a Duplicate Address Detection Neighbor Solicitation or a data packet has been received through a Validating Port other than the one address to which it is currently bound. This implies that a host is performing the DAD procedure for that source address through a different port. This may be due to an attack, the fact that the host may have moved, or just because another host tries to configure an address already used. The binding in this state is then being tested to determine which is the situation.

Next, we describe how the different inputs are processed depending on the state of the binding of the IP address (IPAddr).

A simplified figure of the state machine is included in Figure 2 below.

NO_BIND

- o Upon the reception through a Validating Port (VP) of a Neighbor Solicitation (NS) generated by the Duplicate Address Detection (DAD) procedure (hereafter named DAD_NS) containing Target Address IPAddr, the FCFS SAVI device MUST forward the NS, and T_WAIT milliseconds later, it MUST send a copy of the same message. These DAD_NS messages are not sent through any of the ports configured as Validating Ports. The DAD_NS messages are sent through the Trusted Ports (but, of course, subject to usual switch behavior and possible Multicast Listener Discovery (MLD) snooping optimizations). The state is moved to TENTATIVE. The LIFETIME is set to TENT_LT (i.e., LT:=TENT_LT), the BINDING ANCHOR is set to VP (i.e., P:=VP), and the Creation time is set to the current value of the local clock.

- o Upon the reception through a Validating Port (VP) of a DATA packet containing IPAddr as the source address, the SAVI device SHOULD execute the process of sending Neighbor Solicitation messages of the Duplicate Address Detection process as described in Section 5.4.2 of [RFC4862] for the IPAddr using the following default parameters: DupAddrDetectTransmits set to 2 (i.e., 2 Neighbor Solicitation messages for that address will be sent by the SAVI device) and RetransTimer set to T_WAIT milliseconds (i.e., the time between two Neighbor Solicitation messages is T_WAIT milliseconds). The implications of not following the recommended behavior are described in Appendix A. The DAD_NS messages are not sent through any of the ports configured as Validating Ports. The DAD_NSOL messages are sent through Trusted Ports (but, of course, subject to usual switch behavior and possible MLD snooping optimizations). The SAVI device MAY discard the data packets while the DAD procedure is being executed, or it MAY store them until the binding is created. In any case, it MUST NOT forward the data packets until the binding has been verified. The state is moved to TENTATIVE. The LIFETIME is set to TENT_LT (i.e., LT: =TENT_LT), the BINDING ANCHOR is set to VP (i.e., P:=VP), and the Creation time is set to the current value of the local clock.

- o Data packets containing IPAddr as the source address received through Trusted Ports are processed and forwarded as usual (i.e., no special SAVI processing).

- o DAD_NS packets containing IPAddr as the Target Address that are received through a Trusted Port MUST NOT be forwarded through any of the Validating Ports, but they are sent through the Trusted Ports (but, of course, subject to usual switch behavior and possible MLD snooping optimizations).

- o Neighbor Advertisement packets sent to all nodes as a reply to the DAD_NS (hereafter called DAD_NA) containing IPAddr as the Target Address coming through a Validating Port are discarded.

- o Other signaling packets are processed and forwarded as usual (i.e., no SAVI processing).

TENTATIVE

- o If the LIFETIME times out, the state is moved to VALID. The LIFETIME is set to DEFAULT_LT (i.e., LT:= DEFAULT_LT). Stored data packets (if any) are forwarded.

- o If a Neighbor Advertisement (NA) is received through a Trusted Port with the Target Address set to IPAddr, then the message is forwarded through port P, the state is set to NO_BIND, and the BINDING ANCHOR and the LIFETIME are cleared. Data packets stored corresponding to this binding are discarded.

- o If an NA is received through a Validating Port with the Target Address set to IPAddr, the NA packet is discarded

- o If a data packet with source address IPAddr is received with binding anchor equal to P, then the packet is either stored or discarded.

- o If a data packet with source address IPAddr is received through a Trusted Port, the data packet is forwarded. The state is unchanged.

- o If a data packet with source address IPAddr is received through a Validating Port other than P, the data packet is discarded.

- o If a DAD_NS is received from a Trusted Port, with the Target Address set to IPAddr, then the message is forwarded to the Validating Port P, the state is set to NO_BIND, and the BINDING ANCHOR and LIFETIME are cleared. Data packets stored corresponding to this binding are discarded.

- o If a DAD_NS with the Target Address set to IPAddr is received from a Validating Port P' other than P, the message is forwarded to the Validating Port P and to the Trusted Ports, and the state remains in TENTATIVE; however, the BINDING ANCHOR is changed from P to P', and LIFETIME is set to TENT_LT. Data packets stored corresponding to the binding with P are discarded.

- o Other signaling packets are processed and forwarded as usual (i.e., no SAVI processing).

VALID

- o If a data packet containing IPAddr as the source address arrives from Validating Port P, then the LIFETIME is set to DEFAULT_LT and the packet is forwarded as usual.

- o If a DAD_NS is received from a Trusted Port, then the DAD_NS message is forwarded to port P and is also forwarded to the Trusted Ports (but, of course, subject to usual switch behavior and possible MLD snooping optimizations). The state is changed to TESTING_TP-LT. The LIFETIME is set to TENT_LT.

- o If a data packet containing source address IPAddr or a DAD_NA packet with the Target Address set to IPAddr is received through a Validating Port P' other than P, then the SAVI device will execute the process of sending DAD_NS messages as described in Section 5.4.2 of [RFC4862] for the IPAddr using the following default parameters: DupAddrDetectTransmits set to 2 (i.e., two NS messages for that address will be sent by the SAVI device) and RetransTimer set to T_WAIT milliseconds (i.e., the time between two NS messages is T_WAIT milliseconds). The DAD_NS message will be forwarded to the port P. The state is moved to TESTING_VP. The LIFETIME is set to TENT_LT. The SAVI device MAY discard the data packet while the DAD procedure is being executed, or it MAY store them until the binding is created. In any case, it MUST NOT forward the data packets until the binding has been verified.

- o If a DAD_NS packet with the Target Address set to IPAddr is received through a Validating Port P' other than P, then the SAVI device will forward the DAD_NS packet, and T_WAIT milliseconds later, it will execute the process of sending DAD_NS messages as described in Section 5.4.2 of [RFC4862] for the IPAddr using the following default parameters: DupAddrDetectTransmits set to 1 and RetransTimer set to T_WAIT milliseconds. The DAD_NS messages will be forwarded to the port P. The state is moved to TESTING_VP. The LIFETIME is set to TENT_LT. The SAVI device MAY discard the data packets while the DAD procedure is being executed, or it MAY store them until the binding is created. In any case, it MUST NOT forward the data packets until the binding has been verified.

- o If the LIFETIME expires, then the SAVI device will execute the process of sending DAD_NS messages as described in Section 5.4.2 of [RFC4862] for the IPAddr using the following default parameters: DupAddrDetectTransmits set to 2 (i.e., two NS messages for that address will be sent by the SAVI device) and RetransTimer set to T_WAIT milliseconds (i.e., the time between two NS messages is T_WAIT milliseconds). The DAD_NS messages will be forwarded to the port P. The state is changed to TESTING_TP-LT, and the LIFETIME is set to TENT_LT.

- o If a data packet containing IPAddr as a source address arrives from Trusted Port, the packet MAY be discarded. The event MAY be logged.

- o Other signaling packets are processed and forwarded as usual (i.e., no SAVI processing). In particular, a DAD_NA coming from port P and containing IPAddr as the Target Address is forwarded as usual.

TESTING_TP-LT

- o If the LIFETIME expires, the BINDING ANCHOR is cleared, and the state is changed to NO_BIND.

- o If an NA message containing the IPAddr as the Target Address is received through the Validating Port P as a reply to the DAD_NS message, then the NA is forwarded as usual, and the state is changed to VALID. The LIFETIME is set to DEFAULT_LT

- o If a data packet containing IPAddr as the source address is received through port P, then the packet is forwarded and the state is changed to VALID. The LIFETIME is set to DEFAULT_LT.

- o If a DAD_NS is received from a Trusted Port, the DAD_NS is forwarded as usual.

- o If a DAD_NS is received from a Validating Port P' other than P, the DAD_NS is forwarded as usual, and the state is moved to TESTING_VP.

- o If a data packet is received through a Validating Port P' that is other than port P, then the packet is discarded.

- o If a data packet is received through a Trusted Port, then the packet MAY be discarded. The event MAY be logged.

TESTING_VP

- o If the LIFETIME expires, the BINDING ANCHOR is modified from P to P', the LIFETIME is set to DEFAULT_LT, and the state is changed to VALID. Stored data packet coming from P' are forwarded.

- o If an NA message containing the IPAddr as the Target Address is received through the Validating Port P as a reply to the DAD_NS message, then the NA is forwarded as usual and the state is changed to VALID. The LIFETIME is set to DEFAULT_LT.

- o If a data packet containing IPAddr as the source address is received through port P, then the packet is forwarded.

- o If a data packet containing IPAddr as the source address is received through a Validating Port P'' that is other than port P or P', then the packet is discarded.

- o If a data packet containing IPAddr as the source address is received through a Trusted Port (i.e., other than port P), the state is moved to TESTING_TP-LT, and the packet MAY be discarded.

- o If a DAD_NS is received through a Trusted Port, the packet is forwarded as usual, and the state is moved to TESTING_TP-LT.

- o If a DAD_NS is received through Validating Port P'' other than P or P', the packet is forwarded as usual, and P'' is stored as the tentative port, i.e., P':=P''. The state remains the same.

```
   +---------+  VP_NS, VP_DATA/2xNS                    +-----------+
   |         |---------------------------------------->|           |
   | NO_BIND |                                         | TENTATIVE |
   |         |<----------------------------------------|           |
   +---------+                    TP_NA, TP_NS/-       +-----------+
          ^                                                |
          |                                                | TimeOut
   Timeout|                                                |
          |                                                v
   +---------+  VP_NA/-                                +-----------+
   |         |---------------------------------------->|           |
   | TESTING |                                TP_NS/-  |           |
   |  TP-LT  |<----------------------------------------|    VALID    |
   |         |                           TimeOut/2xNS  |           |
   |         |<----------------------------------------|           |
   +---------+                                         +-----------+
     ^   |                                                ^    |
     |   |                                                |    |
     |   +---------------------      ---------------------+    |
     |       VP_NS/-          |     |  NP_NA, TimeOut/-        |
     |                        v     |                          |
     |                     +-----------+                       |
     |                     |           |                       |
     +---------------------|  TESTING  |<----------------------+
          VP_NS, VP_DATA/- |    VP     |  VP_DATA, VP_NS,
                           +-----------+  VP_NA/2xNS
```

Figure 2: Simplified State Machine

MLD Considerations

The FCFS SAVI device MUST join the solicited node multicast group for all the addresses with a state other than NO_BIND. This is needed to make sure that the FCFS SAVI device will receive the DAD_NS for those addresses. Please note that it may not be enough to rely on the host behind the Validating Port to do so, since the node may move, and after a while, the packets for that particular solicited node multicast group will no longer be forwarded to the FCFS SAVI device. Therefore, the FCFS SAVI device MUST join the solicited node multicast groups for all the addresses that are in a state other than NO_BIND.

3.2.4. FCFS SAVI Port Configuration Guidelines

The guidelines for port configuration in FCFS SAVI devices are as follows:

- o The FCFS SAVI realm (i.e., the realm that is inside the FCFS SAVI protection perimeter) MUST be connected. If this is not the case, legitimate transit traffic may be dropped.

- o Ports that are connected to another FCFS SAVI device MUST be configured as Trusted Ports. Not doing so will significantly increase the memory consumption in the FCFS SAVI devices and may result in legitimate transit traffic being dropped.

- o Ports connected to hosts SHOULD be configured as Validating Ports. Not doing so will allow the host connected to that port to send packets with spoofed source addresses. A valid exception is the case of a trusted host (e.g., a server) that could be connected to a Trusted Port, but untrusted hosts MUST be connected to Validating Ports.

- o Ports connected to routers MUST be configured as Trusted Ports. Configuring them as Validating Ports should result in transit traffic being dropped.

- o Ports connected to a chain of one or more legacy switches that have hosts connected SHOULD be configured as Validating Ports. Not doing so will allow the host connected to any of these switches to send packets with spoofed source addresses. A valid exception is the case where the legacy switch only has trusted hosts attached, in which case it could be connected to a Trusted Port, but if there is at least one untrusted hosts connected to the legacy switch, then it MUST be connected to Validating Ports.

- o Ports connected to a chain of one or more legacy switches that have other FCFS SAVI devices and/or routers connected but had no hosts attached to them MUST be configured as Trusted Ports. Not doing so will at least significantly increase the memory consumption in the FCFS SAVI devices, increase the signaling traffic due to FCFS SAVI validation, and may result in legitimate transit traffic being dropped.

3.2.5. VLAN Support

If the FCFS SAVI device is a switch that supports customer VLANs [IEEE.802-1Q.2005], the FCFS SAVI implementation MUST behave as if there was one FCFS SAVI process per customer VLAN. The FCFS SAVI process of each customer VLAN will store the binding information corresponding to the nodes attached to that particular customer VLAN.

3.3. Default Protocol Values

Following are the default values used in the FCFS SAVI specification.

TENT_LT is 500 milliseconds

DEFAULT_LT is 5 minutes

T_WAIT is 250 milliseconds

An implementation MAY allow these values to be modified, but tuning them precisely is considered out of the scope of this document.

4. Security Considerations

4.1. Denial-of-Service Attacks

There are two types of Denial-of-Service (DoS) attacks [RFC4732] that can be envisaged in an FCFS SAVI environment. On one hand, we can envision attacks against the FCFS SAVI device resources. On the other hand, we can envision DoS attacks against the hosts connected to the network where FCFS SAVI is running.

The attacks against the FCFS SAVI device basically consist of making the FCFS SAVI device consume its resources until it runs out of them. For instance, a possible attack would be to send packets with different source addresses, making the FCFS SAVI device create state for each of the addresses and waste memory. At some point, the FCFS SAVI device runs out of memory and needs to decide how to react. The result is that some form of garbage collection is needed to prune the entries. When the FCFS SAVI device runs out of the memory allocated for the FCFS SAVI DB, it is RECOMMENDED that it create new entries by deleting the entries with a higher Creation time. This implies that older entries are preserved and newer entries overwrite each other. In an attack scenario where the attacker sends a batch of data packets with different source addresses, each new source address is likely to rewrite another source address created by the attack itself. It should be noted that entries are also garbage collected using the LIFETIME, which is updated using data packets. The result is that in order for an attacker to actually fill the FCFS SAVI DB with false source addresses, it needs to continuously send data packets for all the different source addresses so that the entries grow old and compete with the legitimate entries. The result is that the cost of the attack is highly increased for the attacker.

In addition, it is RECOMMENDED that an FCFS SAVI device reserves a minimum amount of memory for each available port (in the case where the port is used as part of the L2 anchor). The recommended minimum is the memory needed to store four bindings associated with the port. The motivation for this recommendation is as follows. An attacker attached to a given port of an FCFS SAVI device may attempt to launch a DoS attack towards the FCFS SAVI device by creating many bindings for different addresses. It can do so by sending DAD_NS for different addresses. The result is that the attack will consume all the memory available in the FCFS SAVI device. The above recommendation aims to reserve a minimum amount of memory per port, so that hosts located in different ports can make use of the reserved memory for their port even if a DoS attack is occurring in a different port.

As the FCFS SAVI device may store data packets while the address is being verified, the memory for data packet storage may also be a target of DoS attacks. The effects of such attacks may be limited to the lack of capacity to store new data packets. The effect of such attacks will be that data packets will be dropped during the verification period. An FCFS SAVI device MUST limit the amount of memory used to store data packets, allowing the other functions to have available memory even in the case of attacks such those described above.

The FCFS SAVI device generates two DAD_NS packets upon the reception of a DAD_NS or a data packet. As such, the FCFS SAVI device can be used as an amplifier by attackers. In order to limit this type of attack, the FCFS SAVI device MUST perform rate limiting of the messages it generates. Rate limiting is performed on a per-port basis, since having an attack on a given port should not prevent the FCFS SAVI device from functioning normally in the rest of the ports.

4.2. Residual Threats

FCFS SAVI performs its function by binding an IP source address to a binding anchor. If the attacker manages to send packets using the binding anchor associated to a given IP address, FCFS SAVI validation will be successful, and the FCFS SAVI device will allow the packet through. This can be achieved by spoofing the binding anchor or by sharing of the binding anchor between the legitimate owner of the address and the attacker. An example of the latter is the case where the binding anchor is a port of a switched network and a legacy switch (i.e., not a SAVI-capable switch) is connected to that port. All the source addresses of the hosts connected to the legacy switch will share the same binding anchor (i.e., the switch port). This means that hosts connected to the legacy switch can spoof each other's IP address and will not be detected by the FCFS SAVI device. This can be prevented by not sharing binding anchors among hosts.

FCFS SAVI assumes that a host will be able to defend its address when the DAD procedure is executed for its addresses. This is needed, among other things, to support mobility within a link (i.e., to allow a host to detach and reconnect to a different Layer 2 anchor of the same IP subnetwork without changing its IP address). So, when a DAD_NS is issued for a given IP address for which a binding exists in an FCFS SAVI device, the FCFS SAVI device expects to see a DAD_NA coming from the binding anchor associated to that IP address in order to preserve the binding. If the FCFS SAVI device does not see the DAD_NA, it may grant the binding to a different binding anchor. This means that if an attacker manages to prevent a host from defending its source address, it will be able to destroy the existing binding and create a new one, with a different binding anchor. An attacker may do so, for example, by intercepting the DAD_NA or launching a DoS attack to the host that will prevent it from issuing proper DAD replies.

Even if routers are considered trusted, nothing can prevent a router from being compromised and sending traffic with spoofed IP source addresses. Such traffic would be allowed with the present FCFS SAVI specification. A way to mitigate this issue could be to specify a new port type (e.g., Router Port (RP)) that would act as Trusted Port for the transit traffic and as Validating Port for the local traffic. A detailed solution about this issue is outside the scope of this document.

4.3. Privacy Considerations

Personally identifying information MUST NOT be included in the FCFS SAVI DB with the MAC address as the canonical example, except when there is an attack attempt involved. Moreover, compliant implementations MUST NOT log binding anchor information except where there is an identified reason why that information is likely to be involved in detection, prevention, or tracing of actual source address spoofing. Information that is not logged MUST be deleted as soon as possible (i.e., as soon as the state for a given address is back to NO_BIND). Information about the majority of hosts that never spoof SHOULD NOT be logged.

4.4. Interaction with Secure Neighbor Discovery

Even if the FCFS SAVI could get information from ND messages secured with Secure Neighbor Discovery (SEND) [RFC3971], in some case, the FCFS SAVI device must spoof DAD_NS messages but doesn't know the security credentials associated with the IPAddr (i.e., the private key used to sign the DAD_NS messages). So, when SEND is deployed, it is recommended to use SEND SAVI [SAVI-SEND] rather than FCFS SAVI.

5. Contributors

Jun Bi
CERNET
Network Research Center, Tsinghua University
Beijing 100084
China
EMail: junbi@cernet.edu.cn

Guang Yao
CERNET
Network Research Center, Tsinghua University
Beijing 100084
China
EMail: yaog@netarchlab.tsinghua.edu.cn

Fred Baker
Cisco Systems
EMail: fred@cisco.com

Alberto Garcia Martinez
University Carlos III of Madrid
EMail: alberto@it.uc3m.es

6. Acknowledgments

This document benefited from the input of the following individuals: Joel Halpern, Christian Vogt, Dong Zhang, Frank Xia, Jean-Michel Combes, Jari Arkko, Stephen Farrel, Dan Romascanu, Russ Housley, Pete Resnick, Ralph Droms, Wesley Eddy, Dave Harrington, and Lin Tao.

Marcelo Bagnulo is partly funded by Trilogy, a research project supported by the European Commission under its Seventh Framework Program.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, May 2000.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

[RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.

7.2. Informative References

[SAVI-FRAMEWORK] Wu, J., Bi, J., Bagnulo, M., Baker, F., and C. Vogt, "Source Address Validation Improvement Framework", Work in Progress, December 2011.

[SAVI-DHCP] Bi, J., Wu, J., Yao, G., and F. Baker, "SAVI Solution for DHCP", Work in Progress, February 2012.

[SAVI-SEND] Bagnulo, M. and A. Garcia-Martinez, "SEND-based Source-Address Validation Implementation", Work in Progress, March 2012.

[RFC1958] Carpenter, B., "Architectural Principles of the Internet", RFC 1958, June 1996.

[RFC3971] Arkko, J., Kempf, J., Zill, B., and P. Nikander, "SEcure Neighbor Discovery (SEND)", RFC 3971, March 2005.

[RFC4732] Handley, M., Rescorla, E., and IAB, "Internet Denial-of-Service Considerations", RFC 4732, December 2006.

[IEEE.802-1D.1998] Institute of Electrical and Electronics Engineers, "IEEE Standard for Local and Metropolitan Area Networks Media Access Control (MAC) Bridges", IEEE Standard 802.1D, 1998.

[IEEE.802-1D.2004]
Institute of Electrical and Electronics Engineers, "IEEE Standard for Local and Metropolitan Area Networks Media Access Control (MAC) Bridges", IEEE Standard 802.1D, 2004.

[IEEE.802-1Q.2005]
Institute of Electrical and Electronics Engineers, "IEEE Standard for Local and metropolitan area networks - Virtual Bridged Local Area Networks", IEEE Standard 802.1Q, May 2005.

[IEEE.802-1X.2004]
Institute of Electrical and Electronics Engineers, "IEEE Standard for Local and metropolitan area networks - Port-Based Network Access Control", IEEE Standard 802.1X, 2004.

Appendix A. Implications of Not Following the Recommended Behavior

This section qualifies some of the SHOULDs that are included in this specification by explaining the implications of not following the recommended behavior. We start by describing the implication of not following the recommendation of generating DAD_NS upon the reception of a data packet for which there is no binding, and then we describe the implications of not discarding the non-compliant packets.

A.1. Implications of Not Generating DAD_NS Packets upon the Reception of Non-Compliant Data Packets

This specification recommends that SAVI implementations generate a DAD_NS message upon the reception of a data packet for which they have no binding. In this section, we describe the implications of not doing so and simply discarding the data packet instead.

The main argument against discarding the data packet is the overall robustness of the resulting network. The main concern that has been stated is that a network running SAVI that discards data packets in this case may end up disconnecting legitimate users from the network, by filtering packets coming from them. The net result would be a degraded robustness of the network as a whole, since legitimate users would perceive this as a network failure. There are three different causes that resulted in the lack of state in the binding device for a legitimate address, namely, packet loss, state loss, and topology change. We will next perform an analysis for each of them.

A.1.1. Lack of Binding State due to Packet Loss

The DAD procedure is inherently unreliable. It consists of sending an NS packet, and if no NA packet is received back, success is assumed, and the host starts using the address. In general, the lack of response is because no other host has that particular address configured in its interface, but it may also be the case that the NS packet or the NA packet has been lost. From the perspective of the sending host, there is no difference, and the host assumes that it can use the address. In other words, the default action is to allow the host to obtain network connectivity.

It should be noted that the loss of a DAD packet has little impact on the network performance, since address collision is very rare, and the host assumes success in that case. By designing a SAVI solution that would discard packets for which there is no binding, we are diametrically changing the default behavior in this respect, since the default would be that if the DAD packets are lost, then the node is disconnected from the network (as its packets are filtered). What is worse, the node has little clue of what is going wrong, since it has successfully configured an address, but it has no connectivity. The net result is that the overall reliability of the network has significantly decreased as the loss of a single packet would imply that a host is disconnected from the network.

The only mechanism that the DAD has to improve its reliability is sending multiple NSs. However, [RFC4862] defines a default value of 1 NS message for the DAD procedure, so requiring any higher value would imply manual configuration of all the hosts connected to the SAVI domain.

A.1.1.1. Why Initial Packets May Be (Frequently) Lost

The Case of LANs

Devices connecting to a network may experience periods of packet loss after the link-layer becomes available for two reasons: Invalid Authentication state and incomplete topology assessment. In both cases, physical-layer connection occurs initially and presents a medium where packets are transmissible, but frame forwarding is not available across the LAN.

For the authentication system, devices on a controlled port are forced to complete 802.1X authentication, which may take multiple round trips and many milliseconds to complete (see [IEEE.802-1X.2004]). In this time, initial DHCP, IPv6 Neighbor Discovery, Multicast Listener, or Duplicate Address Detection messages may be transmitted. However, it has also been noted that some devices have the ability for the IP stack to not see the port as up until 802.1X has completed. Hence, that issue needs investigation to determine how common it is now.

Additionally, any system that requires user input at this stage can extend the authentication time and thus the outage. This is problematic where hosts relying upon DHCP for address configuration time out.

Upon completion of authentication, it is feasible to signal upper-layer protocols as to LAN forwarding availability. This is not typical today, so it is necessary to assume that protocols are not aware of the preceding loss period.

For environments that do not require authentication, addition of a new link can cause loops where LAN frames are forwarded continually. In order to prevent loops, all LANs today run a spanning tree protocol, which selectively disables redundant ports. Devices that perform spanning tree calculations are either traditional Spanning Tree Protocol (STP) (see [IEEE.802-1D.1998]) or rapidly converging versions of the same (Rapid Spanning Tree Protocol (RSTP) / Multiple Spanning Tree Protocol (RSTP)) (see [IEEE.802-1D.2004] and [IEEE.802-1Q.2005]).

Until a port is determined to be an edge port (RSTP/MSTP), the rapid protocol speaker has identified its position within the spanning tree (RSTP/MSTP) or completed a Listening phase (STP), its packets are discarded.

For ports that are not connected to rapid protocol switches, it takes a minimum of three seconds to perform edge port determination (see [IEEE.802-1D.2004]). Alternatively, completion of the Listening phase takes 15 seconds (see [IEEE.802-1D.1998]). During this period, the link-layer appears available, but initial packet transmissions into and out of this port will fail.

It is possible to pre-assess ports as edge ports using manual configuration of all the involved devices and thus make them immediately transmissible. This is never default behavior though.

The Case of Fixed Access Networks

In fixed access networks such as DSL and cable, the end hosts are usually connected to the access network through a residential gateway (RG). If the host interface is initialized prior to the RG getting authenticated and connected to the access network, the access network is not aware of the DAD packets that the host sent out. As an example, in DSL networks, the Access Node (Digital Subscriber Link Access Multiplexer (DSLAM)) that needs to create and maintain binding state will never see the DAD message that is required to create such a state.

A.1.1.1.1. Special Sub-Case: SAVI Device Rate-Limiting Packets

A particular sub-case is the one where the SAVI device itself "drops" ND packets. In order to protect itself against DoS attacks and flash-crowds, the SAVI device will have to rate limit the processing of packets triggering the state-creation process (which requires processing from the SAVI device). This implies that the SAVI device may not process all the ND packets if it is under heavy conditions. The result is that the SAVI device will fail to create a binding for a given DAD_NS packet, which implies that the data packets coming from the host that sent the DAD_NS packet will be filtered if this approach is adopted. The problem is that the host will assume that the DAD procedure was successful and will not perform the DAD procedure again, which in turn will imply that the host will be disconnected from the network. While it is true that the SAVI device will also have to rate limit the processing of the data packets, the host will keep on sending data packets, so it is possible to recover from the alternative approach where data packets trigger the binding-creation procedure.

A.1.2. Lack of Binding State due to a Change in the Topology

If SAVI is deployed in a switched Ethernet network, topology changes may result in a SAVI device receiving packets from a legitimate user for which the SAVI device does not have a binding. Consider the following example:

```
          +------+            +--------+       +---------------+
          |SAVI I|-------------|SWITCH I|-------|rest of the net|
          +------+            +--------+       +---------------+
             |                    |
             |                +--------+
             |                | SAVI II|
             |                +--------+
             |   +----------+     |
             +---|SWITCH II |-----+
                 +----------+
                           |
                         +-----+
                         | Host|
                         +-----+
```

Figure 3: Topology Example

Suppose that after bootstrapping, all the elements are working properly and the spanning tree is rooted in the router and includes one branch that follows the path SWITCH I - SAVI I - SWITCH II, and another branch that follows SWITCH I-SAVI II.

Suppose that the host boots at this moment and sends the DAD_NS. The message is propagated through the spanning tree and is received by SAVI I but not by SAVI II. SAVI I creates the binding.

Suppose that SAVI I fails and the spanning tree reconverges to SWITCH I - SAVI II - SWITCH II. Now, data packets coming from the host will be coursed through SAVI II, which does not have binding state and will drop the packets.

A.1.3. Lack of Binding State due to State Loss

The other reason a SAVI device may not have state for a legitimate address is simply because it lost it. State can be lost due to a reboot of the SAVI device or other reasons such as memory corruption. So, the situation would be as follows. The host performs the DAD procedure, and the SAVI device creates a binding for the host's address. The host successfully communicates for a while. The SAVI device reboots and loses the binding state. The packets coming from the host are now discarded as there is no binding state for that address. It should be noted that in this case, the host has been able to use the address successfully for a certain period of time.

Architecturally, the degradation of the network robustness in this case can be easily explained by observing that this approach to SAVI implementation breaks the fate-sharing principle. [RFC1958] reads:

> An end-to-end protocol design should not rely on the maintenance of state (i.e. information about the state of the end-to-end communication) inside the network. Such state should be maintained only in the endpoints, in such a way that the state can only be destroyed when the endpoint itself breaks (known as fate-sharing).

By binding the fate of the host's connectivity to the state in the SAVI device, we are breaking this principle, and the result is degraded network resilience.

Moving on to more practical matters, we can dig deeper into the actual behavior by considering two scenarios, namely, the case where the host is directly connected to the SAVI device and the case where there is an intermediate device between the two.

A.1.3.1. The Case of a Host Directly Connected to the SAVI Device

The considered scenario is depicted in the following diagram:

```
      +------+             +-----------+       +---------------+
      | Host |-------------|SAVI device|-------|rest of the net|
      +------+             +-----------+       +---------------+
```

Figure 4: Host Attached Directly to SAVI Device

The key distinguishing element of this scenario is that the host is directly connected to the SAVI device. As a result, if the SAVI device reboots, the host will see the carrier disappear and appear again.

[RFC4862] requires that the DAD procedure is performed when the IP address is assigned to the interface (see [RFC4862], Section 5.4):

   Duplicate Address Detection:

   Duplicate Address Detection MUST be performed on all unicast addresses prior to assigning them to an interface, regardless of whether they are obtained through stateless autoconfiguration, DHCPv6, or manual configuration, with the following exceptions: ...

However, it has been stated that some of the widely used OSs actually do perform DAD each time the link is up, but further data would be required for this to be taken for granted. Assuming that behavior, this implies that if the loss of state in the SAVI device also results in the link to the host going down, then the host using the tested OSs would redo the DAD procedure allowing the recreation of the binding state in the SAVI device and preserving the connectivity of the host. This would be the case if the SAVI device reboots. It should be noted, however, that it is also possible that the binding state is lost because of an error in the SAVI process and that the SAVI link does not goes down. In this case, the host would not redo the DAD procedure. However, it has been pointed out that it would be possible to require the SAVI process to flap the links of the device it is running, in order to make sure that the link goes down each time the SAVI process restarts and to improve the chances the host will redo the DAD procedure when the SAVI process is rebooted.

A.1.3.2. The Case of a Host Connected to the SAVI Device through One or More Legacy Devices

The considered scenario is depicted in the following diagram:

```
+------+    +-------------+     +-----------+    +---------------+
| Host |----|Legacy device|-----|SAVI device|----|rest of the net|
+------+    +-------------+     +-----------+    +---------------+
```

Figure 5: Host Attached to a Legacy Device

The key distinguishing element of this scenario is that the host is not directly connected to the SAVI device. As a result, if the SAVI device reboots, the host will not see any changes.

In this case, the host would get disconnected from the rest of the network since the SAVI device would filter all its packets once the state has gone. As the node will not perform the DAD procedure again, it will remain disconnected until it reboots.

As a final comment, it should be noted that it may not be obvious to the network admin which scenario its network is running. Consider the case of a campus network where all the switches in the network are SAVI capable. A small hub connected in the office would turn this into the scenario where the host is not directly connected to the SAVI device. Moreover, consider the case of a host running multiple virtual machines connected through a virtual hub. Depending on the implementation of such a virtual hub, this may turn a directly connected host scenario to the scenario where the multiple (virtual) hosts are connected through a legacy (virtual) hub.

A.1.3.2.1. Enforcing Direct Connectivity between the SAVI Device and the Host

It has been argued that enforcing direct connectivity between the SAVI device and the end host is actually a benefit. There are several comments that can be made in this respect:

o First, it may well be the case in some scenarios that this is desirable, but it is certainly not the case in most scenarios. Because of that, the issue of enforcing direct connectivity must be treated as orthogonal to how data packets for which there is no binding are treated, since a general solution must support directly connected nodes and nodes connected through legacy switches.

o Second, as a matter of fact, the resulting behavior described above would not actually enforce direct connectivity between the end host and the SAVI device as it would work as long as the SAVI device does not reboot. So, the argument being made is that this approach is not good enough to provide a robust network service, but it is not bad enough to enforce the direct connectivity of the host to the SAVI switch.

o Third, it should be noted that topology enforcement is not part of the SAVI problem space and that the SAVI problem by itself is complex enough without adding additional requirements.

A.2. Implications of Not Discarding Non-Compliant Data Packets

The FCFS SAVI mechanism is composed of two main functions, namely, the mechanisms for tracking compliant and non-compliant data packets and the actions to be performed upon the detection of a non-compliant packet. Throughout this specification, we recommend discarding non-compliant data packets. This is because forwarding non-compliant data packets is essentially allowing packets with spoofed source addresses to flow throughout the network. However, there are alternative actions that can be taken with respect to these packets.

For instance, it would be possible to forward the packets and trigger an alarm to network administrators to make them aware of the situation. Similarly, it would be possible to log these events and allow the tracking down cases where packets with spoofed addresses were used for malicious purposes. The reason a site deploying SAVI may not want to take milder actions like the ones mentioned above instead of discarding packets is because there may be cases where the non-compliant packets may be legitimate packets (for example, in the case that the SAVI device is malfunctioning and has failed to create the appropriate bindings upon the reception of a DAD packet).

Authors' Addresses

Erik Nordmark
Cisco Systems
510 McCarthy Blvd.
Milpitas, CA 95035
United States

EMail: nordmark@acm.org

Marcelo Bagnulo
Universidad Carlos III de Madrid
Av. Universidad 30
Leganes, Madrid 28911
Spain

Phone: 34 91 6248814
EMail: marcelo@it.uc3m.es
URI: http://www.it.uc3m.es

Eric Levy-Abegnoli
Cisco Systems
Village d'Entreprises Green Side - 400, Avenue Roumanille
Biot-Sophia Antipolis - 06410
France

EMail: elevyabe@cisco.com