Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 21, 2012

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

February 18, 2012

FCFS SAVI: First-Come First-Serve Source-Address Validation for Locally Assigned IPv6 Addresses
draft-ietf-savi-fcfs-14

Abstract

This memo describes FCFS SAVI a mechanism to provide source address validation for IPv6 networks using the First-Come First-Serve principle. The proposed mechanism is intended to complement ingress filtering techniques to help detect and prevent source address spoofing.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on August 21, 2012.

Copyright Notice

Table of Contents

1. Introduction

This memo describes FCFS SAVI, a mechanism to provide source address validation for IPv6 networks using the First-Come First-Serve 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.

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 there usually are hosts and routers attached. Hosts generate packets with their own address as the source address. This is called the 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 the 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 procedure, but there is a specific mechanism tailored for dealing with DHCP assigned addresses defined in [I-D.ietf-savi-dhcp]. Additional considerations about how to use FCFS SAVI depending on the type of address

management used and the nature of the addresses is discussed in the framework document [I-D.ietf-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 hosts which source address is to be verified. Any verification solely relies in the usage of already available protocols. In other words, FCFS SAVI does not define a new protocol nor define any new message on existing protocols nor require that a host uses an existent protocol message in a different way. In other words, the requirement is no host changes.

FCFS SAVI validation is performed by the FCFS SAVI function. Such function can be placed in different type of devices, including a router or a layer-2 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 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 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 the address ownership proof is based in the First-Come First- Serve 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 which consists on 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 it 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 behaviour implies the verification whether the host is still reachable using the previous binding anchor. In order to do that FCFS SAVI uses the Neighbour 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 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 a 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 to 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 addresses 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 significant memory burden on the FCFS SAVI devices. In order to reduce the memory requirements imposed to 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 criteria: 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 a FCFS SAVI device. The binding information for that particular source address will be stored in this first FCFS SAVI device the packet reaches to.

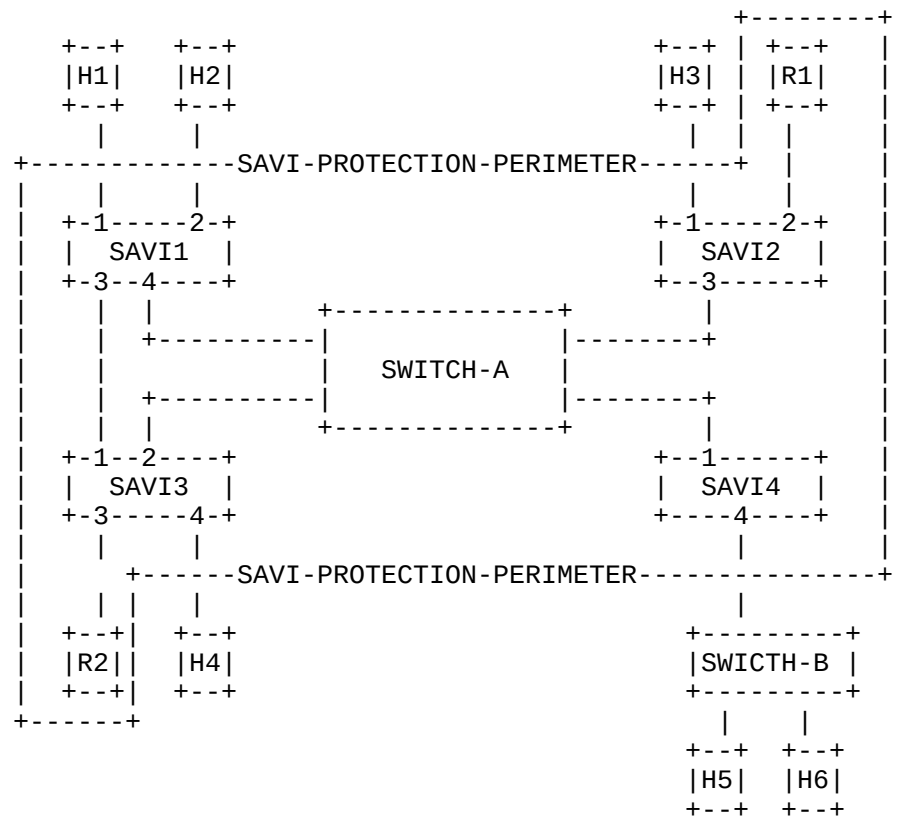
The result is that the FCFS SAVI binding information will be distributed across multiple devices. In order to provide proper source address validation, it is critical that the information distributed among the different FCFS SAVI devices is coherent. In particular, it is important to avoid that the same source address is 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 Neighbour Discovery (ND) protocol [RFC4861] is used, in particular the Neighbour Solicitation (NS) and Neighbour Advertisement (NA) messages. As a simplified example of how this might work: before creating a FCFS SAVI binding in the local FCFS SAVI database, the FCFS SAVI device will send a NS message querying for the address involved. Should any host reply to that message with a 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.

So, summarizing, the proposed FCFS SAVI approach relies on the following design choices:

- FCFS SAVI provides perimetrical security, so some interfaces of a 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.
- A FCFS SAVI device only verifies packets coming through an interface connected to the untrusted part of the topology.
- A 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.
- FCFS SAVI uses the 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. that no data packet can get inside the perimeter without passing through a FCFS SAVI device). Packets that cross the perimeter will be validated while packets that do no 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 picture:



In the figure above, the FCFS SAVI protection perimeter is provided by 4 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, the 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 in them.

Trusted ports are used for connections with the trusted infrastructure, including the communication between FCFS SAVI devices, the communication with routers and the communication of other switches that while they are not FCFS SAVI devices, they only

connect to trusted infrastructure (i.e. other FCFS SAVI devices, routers or other trusted nodes). So, in the figure above, 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 the connect to SWITCH-A to which only trusted nodes are connected. In the figure above, 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 the figure above, ports 1 and 2 of SAVI1, port 1 of SAVI2, 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 which 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. Even the case where the forwarding of packets between different prefixes involve a router is 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 source address, then, from the SAVI device perspective this is equivalent to two hosts with a single interface each with an IP address each. This is supported by SAVI without 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 configure 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 a FCFS SAVI network. A typical example would

be a mobile phone connecting to a 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 (as 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 scenario.

3. FCFS SAVI specification

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].

3.1. FCFS SAVI Data structures

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 FCFS SAVI Data Base (DB). The FCFS SAVI DB will contain a set of entries about the currently used IP source addresses. So each entry will contain the following information:

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

In addition to this, FCFS SAVI needs to know what are the prefixes that 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. A FCFS SAVI device MUST support the following two methods for populating the Prefix List: Manual configuration and Router Advertisement, as detailed next.

Manual configuration: A 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: A FCFS SAVI device MUST support discovery of on-link prefixes through Router Advertisement messages in Trusted Ports. For the 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 the packet has been received through:

- o If the data packet is received through a Trusted port, the data packet is forwarded and no SAVI processing performed to the packet.
- o 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 the 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 describe next how the local traffic, including both control and data packets are 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. So 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:

- IP ADDRESS: IPAddr
- BINDING ANCHOR: P
- LIFETIME: LT

The possible states are:

- NO_BIND
- TENTATIVE
- 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 address 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 it.

TENTATIVE: The binding for a source address for which a data packet or a 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 in this state due to one of two reasons:

When a Duplicate Address Detection Neighbour 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, in order to determine if the binding information should be discarded.

TESTING_VP: A binding for a source address enters in this state when a Duplicate Address Detection Neighbour Solicitation or a data packet has been received through a Validating port other than the one address is currently bound to. This implies that a host is performing the DAD procedure for that source address through a different port. This may due to an attack or to 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.

We describe next 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 below.

NO_BIND

- Upon the reception through a Validating Port (VP) of a Neighbour 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 millisconds 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.
- 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 Neighbour 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 Neighbour 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 Neighbour Solicitation messages is T_WAIT Milliseconds). The implications of not following the recommended behaviour 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 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. 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 received through a Trusted port MUST NOT 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 are forwarded (if any).
- o If a Neighbour Advertisement (NA) is received through a Trusted Port with Target Address set to IPAddr, then 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 a NA is received through a Validating Port with 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 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 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, the state remains in TENTATIVE, but 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 a 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 it 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 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. 2 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 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 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 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. 2 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 DAD_NA coming from port P and containing IPAddr as the target address are forwarded as usual.

TESTING_TP-LT

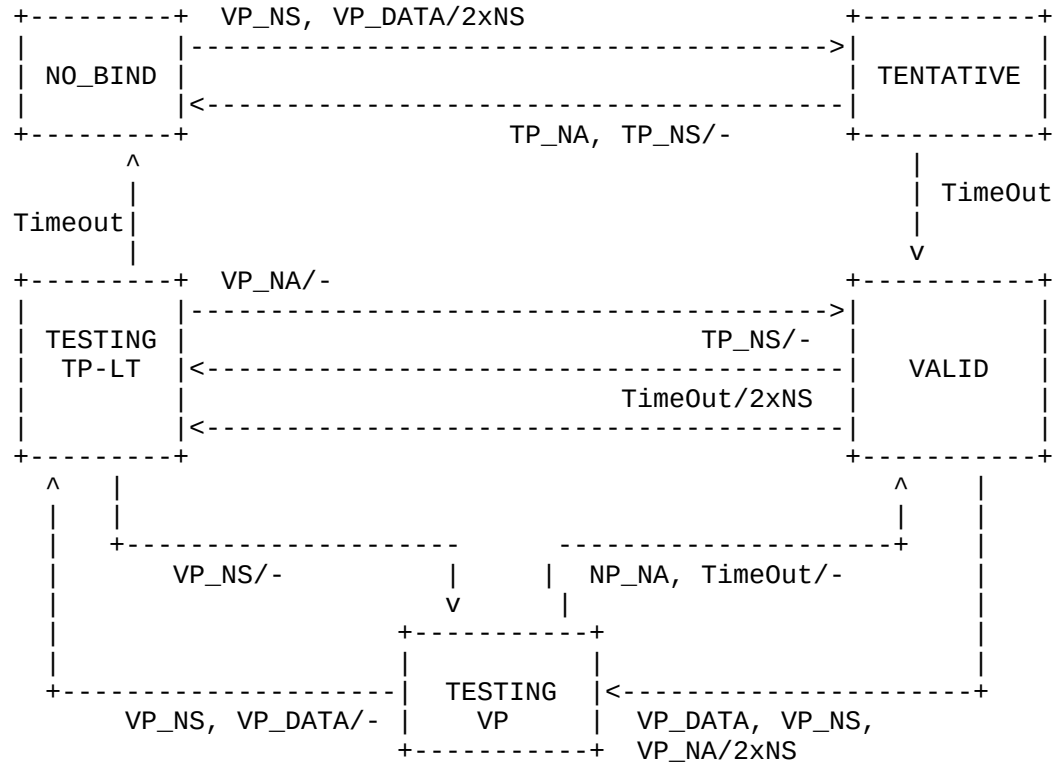
- o If the LIFETIME expires, the BINDING ANCHOR is cleared and the state is changed to NO_BIND
- o If a NA message containing the IPAddr as target address is received through the Validating port P as a reply to the DAD_NS message, then the NA is forwarded as usual and the state is changed to VALID. The 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, the state is moved to TESTING_VP.
- o If a data packet is received through a Validating Port port P’ that is other than port P, then the packet is discarded.
- o If a data packet is received through a Trusted Port 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. Data packet stored coming from P’ are forwarded.

- o If a NA message containing the IPAddr as 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 port (i.e. other than port P), the state is moved to TESTING_TP-LT, and the packet MAY is discarded.
- o If a DAD_NS is received through 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.

Simplified state machine figure



MLD considerations

The FCFS SAVI device MUST join the Solicited Node Multicast group for all the addresses which state is 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 doing 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. So, 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:

- 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.
- 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.
- 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 address. A valid exception is the case of a trusted host (e.g. a server) which could be connected to a Trusted port, but untrusted hosts MUST be connected to Validating ports.
- Ports connected to routers MUST be configured as Trusted ports. Configuring them as Validating ports should result in transit traffic being dropped.
- 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 address. 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.
- 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 and increase the signaling traffic due to FCFS SAVI validation and may results in legitimate transit traffic being dropped.

3.2.5. VLAN support

In the case 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 the nodes attached to that particular customer VLAN.

3.3. Default Protocol Values

The 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 that tuning them precisely is considered out of scope of this document.

4. Security Considerations

Denial of service attacks

There are two types of Denial of Service (DoS) attacks [RFC4732] that can be envisaged in a 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 on making the FCFS SAVI device to 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 to create state for each of the addresses and waste memory. At some point the FCFS SAVI device runs out of memory and it needs to decide how to react in this situation. The result is that some form of garbage collection is needed to prune the entries. It is RECOMMENDED that when the FCFS SAVI device runs out of the memory allocated for the FCFS SAVI DB, it creates new entries by deleting the entries which Creation Time is higher. 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 address, 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, in order for the entries to grow old and compete with the legitimate entries. The result is that the cost of the attack for the attacker is highly increased.

In addition, it is also RECOMMENDED that a 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 4 bindings associated to the port. The motivation for this recommendation is as follows: an attacker attached to a given port of a 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 attack will be then that data packets will be dropped during the verification period. A 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 as the above described.

The FCFS SAVI device generates 2 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 performs rate limiting of the messages it generates. The rate limiting is performed on a per port basis, since having an attack on a given port should not prevent the FCFS SAVI device to function normally in the rest of the ports.

Residual threats.

FCFS SAVI perform 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

because the binding anchor is shared among 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. no 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 this 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 a 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 to issue proper DAD replies.

Even if routers are considered as trusted, nothing can prevent that a router could be compromised and send traffic with spoofed IP source addresses. Such a 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.

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 attempt of attack involved. Moreover, compliant implementation 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 the state for a given address is back to NO_BIND). Information about the majority of hosts that never spoof SHOULD NOT be logged.

Interaction with Secure Neighbour Discovery

Even if FCFS SAVI could get information from ND messages secured with 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 [I-D.ietf-savi-send] rather than FCFS SAVI."

5. IANA Considerations

This document has no actions for IANA.

6. 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

7. Acknowledgments

This draft benefited from the input from: 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.

8. References

8.1. 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.

8.2. Informative References

[I-D.ietf-savi-framework]
Wu, J., Bi, J., Bagnulo, M., Baker, F., and C. Vogt, "Source Address Validation Improvement Framework", draft-ietf-savi-framework-06 (work in progress), December 2011.

[I-D.ietf-savi-dhcp]
Bi, J., Wu, J., Yao, G., and F. Baker, "SAVI Solution for DHCP", draft-ietf-savi-dhcp-12 (work in progress), February 2012.

[I-D.ietf-savi-send]
Bagnulo, M. and A. Garcia-Martinez, "SEND-based Source-Address Validation Implementation", draft-ietf-savi-send-06 (work in progress), October 2011.

[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.802-1Q.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.

Appendix A. Implications of not following the recommended behaviour

This section qualifies some of the SHOULDs that are included in this specification, by explaining the implications of not following the recommended behaviour. We start by describing the implication of not following the recommendation of generating DAD-NS upon the reception of a data packet for which the 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 SAVI implementations to generate a DAD_NS message upon the reception of a data packet for which they have no binding for. 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 discard data packets in this case may end up disconnecting legitimate users from the network, by filtering packets coming from them. The net result would 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 on sending a 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 their interface, but it may also be the case that the NS packet or the NA packet has been lost. From the sending host perspective 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 to send multiple NS. However, current 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 Neighbour 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 which 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 which 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 which perform spanning-tree calculations are either traditional Spanning-Tree Protocol (STP) (see IEEE802.1D-1998) or rapidly converging versions of the same (RSTP/MSTP) (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 which are not connected to rapid protocol switches, it takes a minimum three seconds to perform edge port determination (see IEEE 802.1D-2004). Alternatively completion of 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 behaviour 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 residential gateway 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(DSLAM) that needs to create and maintain binding state will never see the DAD message that is required to create such state.

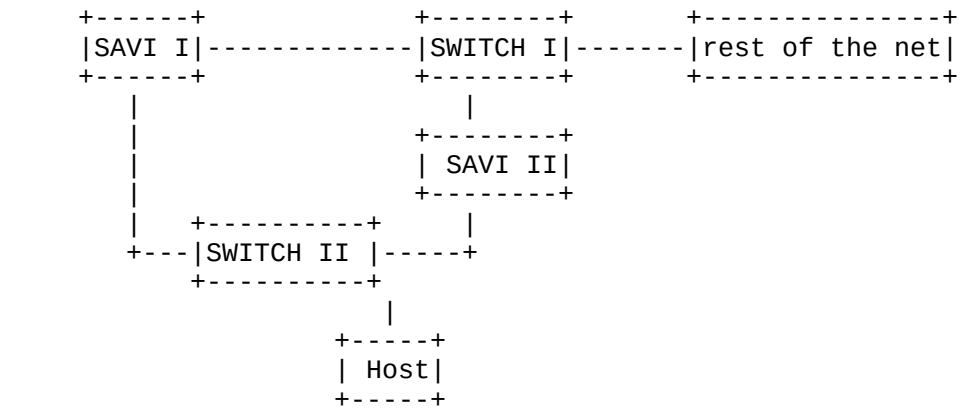
A.1.1.1.1. Special sub-case:SAVI device rate-limiting packets

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 require processing from the SAVI device). This implies that the SAVI device may not process all the ND packets in case 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

In the case SAVI is being 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 for. Consider the following example:



Suppose that after bootstrapping all the elements are working properly and the spanning tree is rooted in the router and it includes one branch that goes SWITCH I-SAVI I- SWITCH II and another branch that goes 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 it 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 why 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 communicate for a while. The SAVI device reboots and lost 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. RFC 1958 reads:

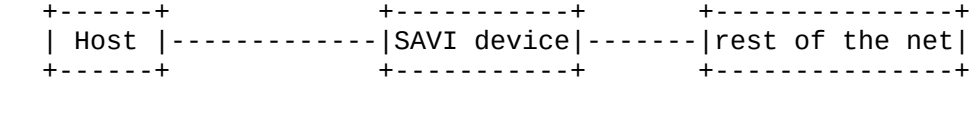
> An end-to-end protocol design should not rely on the maintenance of state (i.e. information about the state of the end-to-end communication) inside the network. Such state should be maintained only in the endpoints, in such a way that the state can only be destroyed when the endpoint itself breaks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



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.

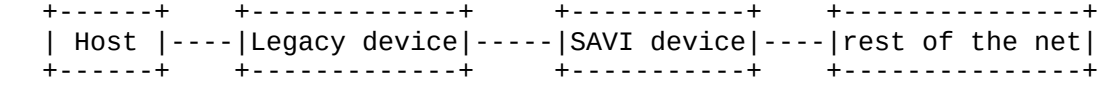
RFC4862 requires that the DAD procedure is performed when the IP address is assigned to the interface, quoting RFC4862 section 5.4. Duplicate Address Detection:

> 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 OSes actually do perform DAD each time the link is up, but further data would be required to take this for granted. Assuming that behaviour, that implies that if the lost of state in the SAVI device also results in the link to the host going down, then the host using the tested OSes 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 though, that it is also possible that the binding state is lost for whatever 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 links goes down each time the SAVI process restarts and improving 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 picture:



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 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, 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 the direct connectivity between the SAVI device and the end host is actually a feature. There are several comments that can be made in this respect:

First, it may well be the case in some scenarios 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.

Second, as a matter of fact, the resulting behaviour 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 would 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 host to the SAVI switch.

Third, it should be noted that topology enforcement is not part of the SAVI problem space and that the SAVI problem by itself is hard enough to add additional requirements.

A.2. Implications of not diacarding 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 to discard non compliant data packets. This is so because forwarding non compliant data packets is essentially allowing packets with spoofed source address to flow throughout the network. However, there are alternative actions that can be taken with respect to this packets. For instance, it would be possible to forward the packets and trigger an alarm to the network administrator to make him aware of the situation. Similarly, it would be possible to log these events and allow to track down the cases where the packets with spoofed

addresses were used for malicious purposes. The reason why a site deploying SAVI may not want to take milder action 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 it has failed to create the appropriate bindings upon the reception of a DAD packets).

Authors' Addresses

Erik Nordmark
Cisco
510 McCarthy Blvd.
Milpitas, California 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

SAVI Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 24, 2014

M. Bagnulo
A. Garcia-Martinez
UC3M

January 20, 2014

SEND-based Source-Address Validation Improvement
draft-ietf-savi-send-11

Abstract

This memo specifies SEND SAVI, a mechanism to provide source address validation using the SEND protocol. The proposed mechanism complements ingress filtering techniques to provide a finer granularity on the control of IPv6 source addresses.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on July 24, 2014.

Table of Contents

1. Introduction

This memo specifies SEND SAVI (SEcure Neighbor Discovery Source Address Validation Improvement), a mechanism to provide source address validation for IPv6 networks using the SEND protocol [RFC3971]. The proposed mechanism complements ingress filtering techniques to provide a finer granularity on the control of the source addresses used.

SEND SAVI uses the DAD_NSOL (Duplicate Address Detection Neighbor SOLicitation) and the DAD_NADV (DAD Neighbor ADVertisement) messages defined in [RFC4862], and the NUD_NSOL (Neighbor Unreachability Detection Neigbor SOLicitation) and NUD_NADV (NUD Neighbor ADVertisement) messages defined in [RFC4861] to validate the address ownership claim of a node. Using the information contained in these messages, host IPv6 addresses are associated to switch ports, so that data packets will be validated by checking for consistency in this binding, as described in [RFC7039]. In addition, SEND SAVI prevents hosts from generating packets containing off-link IPv6 source addresses.

Scalability of a distributed SAVI system comprising multiple SEND SAVI devices is preserved by means of a deployment scenario in which SEND SAVI devices form a "protection perimeter". In this deployment scenario, the distributed SAVI system only validates the packets when they ingress to the protection perimeter, not in every SEND SAVI device traversed.

The SEND SAVI specification, as defined in this document, is limited to links and prefixes in which every IPv6 host and every IPv6 router uses the SEND protocol [RFC3971] to protect the exchange of Neighbor Discovery information. If the SEND protocol is not used, we can deploy other SAVI solutions relying on monitoring different address configuration mechanisms to prove address ownership. For example, FCFS (First-Come, First-Served) SAVI [RFC6620] can be used by nodes locally configuring IPv6 addresses by means of the Stateless Address Autoconfiguration mechanism [RFC4862].

SEND SAVI is designed to be deployed in SEND networks with as few changes to the deployed implementations as possible. In particular, SEND SAVI does not require any changes in the nodes whose source address is to be verified. This is because verification solely relies in the usage of already available protocols. Therefore, SEND SAVI does neither define a new protocol, nor define any new message on existing protocols, nor require that a host or router uses an existing protocol message in a different way.

An overview of the general framework about Source Address Validation

Improvement is presented in [RFC7039].

2. Background to SEND SAVI

2.1. Address Validation Scope

The application scenario of SEND SAVI is limited to the local link. This means that the goal of SEND SAVI is to verify that the source addresses of the packets generated by the nodes attached to the local link have not been spoofed, and that only legitimate routers generate packets with off-link IPv6 source addresses.

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

SEND SAVI allows the validation of the source address of the local traffic, i.e., it allows to verify that the source addresses of the packets generated by the nodes attached to the local link have not been spoofed. SEND SAVI also provides means to prevent hosts from generating packets with source addresses derived from off-link prefixes. However, SEND SAVI does not provide the means to verify if a given router is actually authorized to forward packets containing a particular off-link source address. Other techniques, like ingress filtering [RFC2827], are recommended to validate transit traffic.

2.2. Binding Creation for SEND SAVI

SEND SAVI devices filter packets according to bindings between a layer-2 anchor (the binding anchor) and an IPv6 address. These bindings should allow legitimate nodes to use the bounded IPv6 address as source address, and prevent illegitimate nodes to do so.

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 SEND SAVI device is configured to drop non-compliant packets, then the usage of SEND SAVI may open a new vector of Denial-of-Service (DoS) attacks, based on spoofed binding anchors. For that reason, implementations of this specification use switch ports as their 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.

SEND [RFC3971] provides tools to assure that a ND (Neighbor Discovery) message containing a CGA (Cryptographically Generated Addresses) option and signed by a RSA option has been generated by the legitimate owner of the CGA IPv6 address.

SEND SAVI uses SEND validated messages to create bindings between the CGA and the port of the SEND SAVI device from which it is reasonable to receive packets with the CGA as source addresses. The events that trigger the binding creation process in a SEND SAVI device are:

- The reception of a DAD_NSOL message, indicating the attempt of a node to configure an address. This may occur when a node configures an address for the first time or after being idle for some time, or when the node has changed the physical attachment point to the layer-2 infrastructure.
- The reception of any other packet (including data packets) with a source address for which no binding exists. This may occur if DAD_NSOL messages were lost, a node has changed the physical attachment point to the layer-2 infrastructure without issuing a DAD_NSOL message, a SAVI device loses a binding (for example, due to a restart), or the link topology changed.

When the binding creation process is triggered, the SEND SAVI device has to assure that the node for which the binding is to be created is the legitimate owner of the address. For the case in which the binding creation process initiated by a DAD_NSOL exchange, the SEND SAVI device waits for the reception of a validated DAD_NADV message indicating that other node had configured the address before, or validated DAD_NSOL messages arriving from other locations indicating that another node is trying to configure the same address at the same time. For the case in which other packets than a DAD_NSOL initiate the creation of the binding, the SEND SAVI device explicitly requires the node sending those packets to prove address ownership by issuing a secured NUD_NSOL which has to be answered with a secured NUD_NADV by the probed node.

SEND SAVI devices issue secured NUD_NSOL messages periodically in order to refresh bindings, which had to be answered with a valid NUD_NADV message by the node for which the binding exists.

SEND SAVI devices only forward packets with off-link source addresses if they are received from a port manually configured to connect to a router.

SEND SAVI needs to be protected against replay attacks, i.e., attacks in which a secured SEND message is replayed by another node. As discussed before, the SEND SAVI specification uses SEND messages to

create a binding between the address contained in the message (that must be signed by a node possessing the private key associated to the address) and the port through which the message is received. If an attacker manages to obtain such a message from another node, for example because the message was sent to the all-nodes multicast address or because the attacker has subscribed to the Solicited Node multicast address associated to a remote node, it could replay it preserving the original signature. This may create an illegitimate binding in the SEND SAVI device, or could be used to abort address configuration at other node. While SEND provides some means to limit the impact of the replay of ND messages, the emphasis for SEND anti-replay protection is to limit to a short period of time the validity of the ND information transmitted in the message, for example, the relationship between an IPv6 address and a layer-2 address. Note that the period must be long enough to assure that the information sent by the legitimate sender is considered valid despite the possible differences in clock synchronization between sender and receiver(s). For example, with the values recommended by [RFC3971] for TIMESTAMP_FUZZ and TIMESTAMP_DRIFT, a node receiving a DAD_NSOL message would not discard replays of this message being received within a period of approximately 2 seconds (more precisely, 2/0.99 seconds). The underlying assumption for SEND security is that even if the message is replayed by another node during this period of time, the information disseminated by ND is still the same. However, allowing a node to replay a SEND message do have impact to SEND SAVI operation, regardless the time elapsed since it was generated, since the node can create a new binding in a SEND SAVI device for the port to which an illegitimate node attaches. As can be concluded, the protection provided by SEND is not enough in all cases for SEND SAVI.

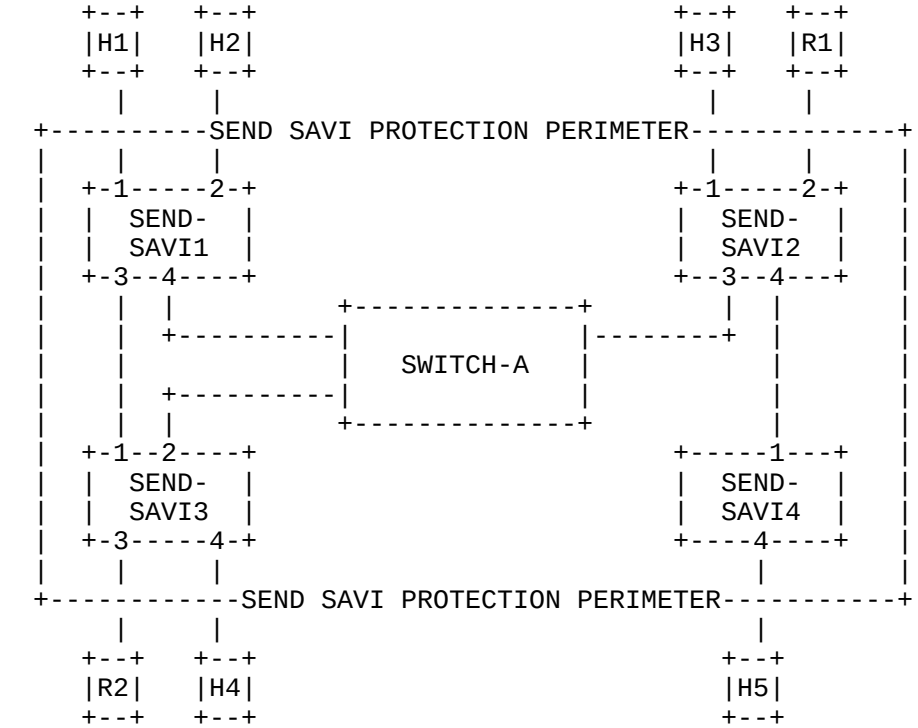
SEND SAVI increases the protection against the replay attacks compared to SEND. First, each node is required to connect to the SEND SAVI topology through a different port to prevent eavesdropping before entering to the SAVI protection perimeter. Then, SEND SAVI bindings are updated only according to messages whose dissemination can be restricted in the SEND SAVI topology without interfering with normal SEND operation. The messages used by SEND SAVI to create bindings are DAD_NSOL messages, for which SEND SAVI limits its propagation to the ports through which a previous binding for the same IPv6 address existed (see Section 3.3.2), and NUD_NADV messages in response to a secured NUD_NSOL sent by the SEND SAVI device only through the tested port. Finally, SEND SAVI filtering rules prevent nodes from replaying messages generated by the SEND SAVI devices themselves. Section 5.1 discusses in more detail the protection provided by SEND SAVI against replay attacks.

2.3. SEND SAVI Protection Perimeter

In order to reduce computing and state requirements in SEND SAVI devices, SEND SAVI devices can be deployed to form a "protection perimeter" [RFC7039]. With this deployment strategy, SEND SAVI devices perform source address validation only when packets enter in the protected realm defined through the protection perimeter. The perimeter is defined by appropriate configuration of the roles of each port, which can be 'Validating' or 'Trusted':

- Validating ports (VPs) are ports in which SEND SAVI filtering and binding creation is performed.
- Trusted ports (TPs) are ports in which limited processing is performed. Only SEND messages related with certificates, prefix information and DAD operation are processed, in order to update the state of the SEND SAVI device or the state related with any of the Validating ports of the switch.

The following figure shows a typical topology involving trusted and untrusted infrastructure.



Trusted ports are used for connections with trusted infrastructure, such as routers and other SEND SAVI devices. Port 2 of SEND-SAVI2 and port 3 of SEND-SAVI3 are Validating ports because they connect to routers. Port 3 of SEND-SAVI1 and port 1 of SEND-SAVI3, and port 4 of SEND-SAVI2 and port 1 of SEND-SAVI4 are trusted because they connect two SAVI devices. Finally, port 4 of SEND-SAVI1, port 3 of SEND-SAVI2 and port 2 of SEND-SAVI3 are trusted because they connect to SWITCH-A to which only trusted nodes are connected.

Validating ports are used for connection with non-trusted infrastructure. Therefore, hosts connect normally to Validating ports. So, in the figure above, ports 1 and 2 of SEND-SAVI1, port 1 of SEND-SAVI2, port 4 of SEND-SAVI3 are Validating ports because they connect to hosts. Port 4 of SEND-SAVI4 is also a Validating port because it is connected to host H5.

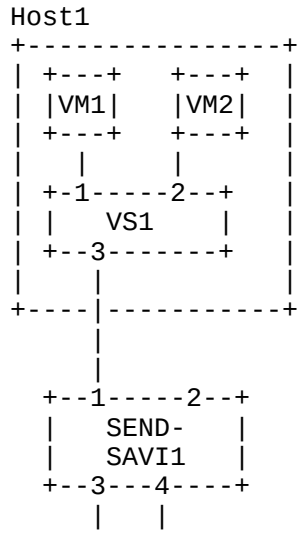
For a more detailed discussion on this, see Section 3.4.

2.4. Special cases

Multi-subnet links: In some cases, a given subnet may have several prefixes. This is supported by SEND SAVI as any port can support multiple prefixes.

Multihomed hosts: A multihomed host is a host with multiple interfaces. The interaction between SEND 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 source address, then from the perspective of a SEND SAVI device, this is equivalent to two hosts with a single interface, each with an IP address. SEND SAVI supports this without 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 SEND 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 configure the SEND SAVI bindings to allow binding the address of the multihomed host to multiple anchors simultaneously.

Virtual switches: A hypervisor or a host Operating System may perform bridging functions between virtual hosts running on the same machine. The hypervisor or host OS may in turn connect to a SEND SAVI system. This scenario is depicted in the next figure, with two virtual machines VM1 and VM2 connected through a virtual switch VS1 to SEND SAVI device SEND-SAVI1. The attachment points of VS1 to VM1 and VM2 are configured as Validating.



In order to provide proper security against replay attacks, it is recommended to perform SEND SAVI filtering as close to untrusted hosts as possible (see Section 3.4 and Section 5.1). In this scenario, this objective can be achieved by enabling SEND SAVI validation in VS1. Ideally, VS1 could be integrated into the SEND SAVI protection perimeter, if the hypervisor or host OS at Host1 can be trusted (even though VM1 and VM2 could not be trusted). To do so, both the attachment to SEND-SAVI1 at VS1, and port 1 at SEND-SAVI1, are configured as Trusted.

If the administrator of the network does not trust on VS1, port 1 of SEND-SAVI1 is configured as Validating, so that every address being used at Host1 is validated at SEND-SAVI1 by SEND SAVI. The attachment point to the physical network at VS1 should be configured as Trusted, if the host administrator knows that it is connected to a SEND SAVI device; in this case, VS1 relies on the infrastructure comprised by the physical SEND SAVI devices, but not the other way. Packets egressing from VM1 are validated twice, first at VS1, and then at SEND-SAVI1; packets going in the reverse direction (from an external host to VM1) are validated once, when they first reach a SEND SAVI device. If the administrator of VS1 does not trust on the physical switch to which it attaches, it can configure the attachment to SEND-SAVI1 as Validating. In the figure above, this means that a packet going from another host to VM1 would be validated twice, once when entering the SEND SAVI perimeter formed by the physical devices, and the other when entering at VS1.

Untrusted routers: One can envision scenarios where routers are dynamically attached to a SEND SAVI network. A typical example would be a mobile phone connecting to a SEND SAVI switch where the mobile phone is acting as a router for other personal devices that are accessing the network through it. Regarding to the validation of the source address performed in a SEND SAVI device, such an untrusted router does not seem to directly fall in the category of Trusted infrastructure (as 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 SEND SAVI switch would discard all the packets containing an off-link source address coming from that device. Although the SEND SAVI device to which this router attaches could be configured to permit the transit of packets with source addresses belonging to the set of prefixes reachable through the untrusted router, such a mechanism is out of the scope of this document. As a result, the default mechanism described in this specification cannot be applied in such a scenario.

3. SEND SAVI Specification

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 [RFC2119].

3.1. SEND SAVI Data Structures

The following three data structures are defined for SEND SAVI operations:

SEND SAVI Data Base. The SEND SAVI function relies on state information binding the source IPv6 address used in data packets to the port through which the legitimate node connects. Such information is stored in the SEND SAVI Data Base. The SEND SAVI Data Base is populated with the contents of validated SEND messages. Each entry contains the following information:

- IPv6 source address
- Binding anchor: port through which the packet was received
- Lifetime
- Status: TENTATIVE_DAD, TENTATIVE_NUD, VALID, TESTING_VP, TESTING_VP'
- Alternative binding anchor: port from which a DAD_NSOL message or any data packet has been received while a different port was stored in the binding anchor for the address.
- Creation time: the value of the local clock when the entry was firstly created

SEND SAVI Prefix list. SEND SAVI devices need to know which are the

link prefixes in order to identify local and off-link traffic. A SEND SAVI device MUST support discovering this information from the Prefix Information option [RFC4861] with the L set bit set of RADV messages coming from Trusted ports, as described in Section 3.3.2. The list of prefixes MAY also be configured manually. This information is not specific to a given port. The SEND SAVI Prefix list contains one entry per prefix in use, as follows:

- Prefix: prefix included in a Prefix Information option
- Prefix lifetime: time in seconds that the prefix is valid. Initially set to the Valid Lifetime value of the Prefix Information option of a valid RADV message, or set to a value of all one bits (0xffffffff), which represents infinity, if configured manually.

When the SEND SAVI device boots, it MUST send a Router Solicitation (RSOL) message, which does not need to be secured if the unspecified address is used (see [RFC3971], sections 5.1.1 and 5.2.1). The SAVI device SHOULD issue a RSOL message in case the prefix entry is about to expire.

3.2. SEND SAVI Device Configuration

In order to perform SEND SAVI operation, some basic parameters of the SEND SAVI device have to be configured. Since a SEND SAVI device operates as a SEND node to generate NUD_NSOL, RSOL or Certification Path Solicitation (CPS) messages,

- the SEND SAVI device MUST be configured with a valid CGA address. When the SEND SAVI device configures this address, it MUST behave as regular SEND node, i.e., using secured NSOL messages to perform DAD, etc., in addition to fulfill the requirements stated for regular IPv6 nodes [RFC6434].
- the SEND SAVI device MAY be configured with at least one trust anchor, if it is configured to validate RADV messages (see Section 3.3.2). In this case the SEND SAVI device MAY be configured with certification paths. The alternative is obtaining them by means of issuing Certification Path Solicitation messages, as detailed in the SEND specification SEND specification [RFC3971].

In addition, the port role for each port of the SEND SAVI device MUST be configured. The guidelines for this configuration are specified in Section 3.4.

3.3. Traffic Processing

In this section we describe how packets are processed by a SEND SAVI device. Behavior varies depending on if the packet belongs to local or transit traffic. This is determined by checking if the prefix of

the source address is included in the SEND SAVI prefix list or the unspecified address (local traffic), or not included in the SEND SAVI prefix list (transit traffic).

3.3.1. Transit Traffic Processing

Transit traffic processing occurs as follows:

- If the SEND SAVI device receives a transit traffic packet through a Trusted port, it forwards it without any SAVI processing.
- If the SEND SAVI device receives a transit traffic packet through a Validating port, it discards the packet.

3.3.2. Local Traffic Processing

If the verification of the source address of a packet shows that it belongs to local traffic, this packet is processed using the state machine described in this section.

For the rest of the section, the following assumptions hold:

- When it is stated that a secured NUD_NSOL message is issued by a SEND SAVI device through a port P, this means the following: the SEND SAVI device generates a NUD_NSOL message according to the Neighbor Unreachability Detection procedure described in [RFC4861], addressed to the IPv6 target address, which is the source address of the packet triggering the procedure. This message is secured by SEND as defined in [RFC3971]. The source address used for issuing the NUD_NSOL message is the source address of the SEND SAVI device. The message is sent only through port P.
- When it is stated that a validated NUD_NADV message is received by a SEND SAVI device, this means that: a SEND secured NUD_NADV message has been received by the same port P through which the corresponding NUD_NSOL message was issued, and the NUD_NADV message has been validated according to [RFC3971] to prove ownership for the IPv6 address under consideration and to prove that it is a response for the previous NUD_NSOL message issued by the SEND SAVI device (containing the same nonce value as the NUD_NSOL message to which it answers).

We use VP to refer to a Validating port, and TP to refer to a Trusted port.

The state machine is defined for a binding of a given source IPv6 address in a given SEND SAVI device. In the transitions considered, packets described as inputs refer to the IPaddr IPv6 address associated to the state machine.

The possible states for a given IPaddr are: NO_BIND, TENTATIVE_DAD,

TENTATIVE_NUD, VALID, TESTING_VP and TESTING_VP'. The NO_BIND state represents that no binding exists for IPaddr; this is the state for all addresses unless a binding is explicitly created.

The states can be classified into 'forwarding' states, i.e., states in which packets received from the port associated to the IPv6 address are forwarded, and 'non-forwarding' states, i.e., states in which packets different to the ones used for signaling are not forwarded. VALID, TENTATIVE_DAD, TESTING_VP and TESTING_VP' are forwarding states, and NO_BIND and TENTATIVE_NUD are non-forwarding states.

The SEND SAVI device MUST join the Solicited Node Multicast group for all the addresses whose state is other than NO_BIND. This is needed to make sure that the SEND SAVI device receives DAD_NSOL messages issued for those addresses. Note that it may not be enough to relay on the Multicast Listener Discovery (MLD) messages being sent by the node attached to a Validating port for which a binding for the corresponding address exist, since the node may move and packets sent to that particular Solicited Node Multicast group may no longer be forwarded to the SEND SAVI device.

In order to determine which traffic is on-link and off-link, the SEND SAVI device MUST support discovery of this information from the Prefix Information option with the L set bit set of RADV messages. In this case, at least one router SHOULD be configured to advertise RADV messages containing a Prefix Information option with the prefixes that the untrusted nodes can use as source addresses, and the bit L set. An alternative to this is to configure manually the SEND SAVI prefix list, or restrict to the use of link-local addresses.

SEND SAVI devices MUST discard RADV messages received from Validating ports. RADV messages are only accepted and processed when received through Trusted ports.

SEND SAVI devices SHOULD NOT validate RADV messages to update the SEND SAVI Prefix list and forward them to other nodes. These messages can only be received from Trusted ports, and we assume that routers are trusted. Validating RADV messages would be required in any SEND SAVI device the node is traversing. Besides, hosts will validate this message before using the information it contains.

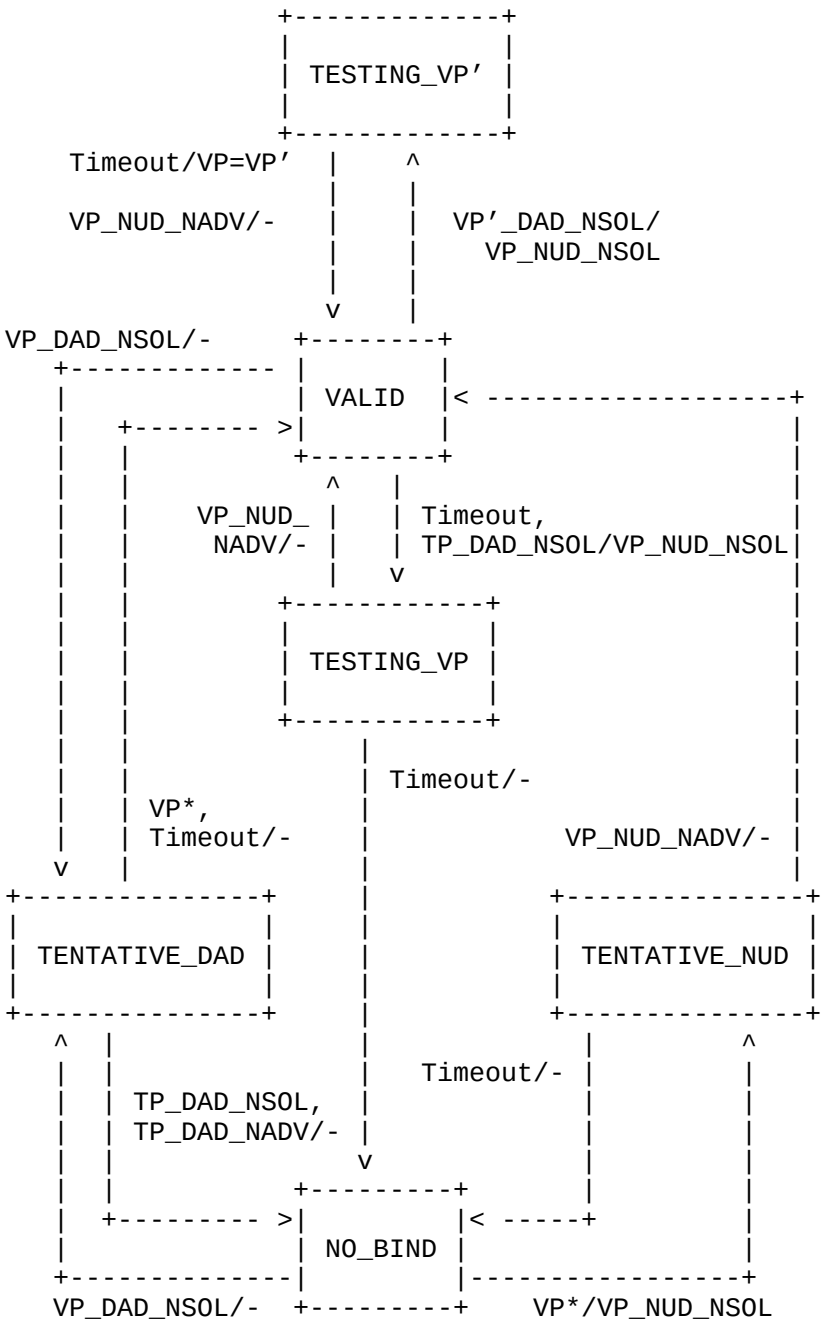
In case SEND SAVI devices are configured to validate RADV message, SEND SAVI devices SHOULD support the processing of validated Certification Path Advertisement (CPA) messages, sent in reply to CPS messages, to acquire certificates used to validate router messages, or alternatively SHOULD be configured with a certification path.

The state machine defined for SEND SAVI operation adheres to the following design guidelines:

- The only events which trigger state changes from forwarding to non-forwarding states and vice versa are the reception of DAD_NSOL, DAD_NADV and NUD_NADV, or the expiration of a timer. The other possible input to consider is 'any other packet', which could generate changes to states belonging to the same forwarding or non-forwarding class as the original state. In other words, when 'any other packet' is received, the state cannot move from being forwarding to non-forwarding and vice versa. The reduced set of messages being able to trigger a change simplifies the processing at SEND SAVI devices.
- DAD_NADV and NUD_NADV are only processed when they are a response to a DAD_NSOL or a NUD_NSOL message.
- SEND SAVI devices MUST only use ND messages received through Validating ports if they are valid, otherwise they discard them. SEND SAVI devices SHOULD assume that such messages received from Trusted ports have been validated by other SEND SAVI devices, or come from a trusted device such a router, so they SHOULD NOT attempt to validate them in order to reduce processing load at the SEND SAVI device.
- The only messages the SEND SAVI device is required to generate specifically per each source IP address are MLD and NUD_NSOL messages. This also keeps the state machine simple.
- Well-behaved nodes are expected to initiate communication by sending secured DAD_NSOL messages. The SEND SAVI state machine is tailored to efficiently process these events. The reception of other packet types without receiving previously validated DAD_NSOL messages is assumed to be consequence of bad-behaving nodes or infrequent events (such as packet loss, a change in the topology connecting the switches, etc.) While a binding will ultimately be created for nodes affected by such events, simplicity of the state machine is prioritized over any possible optimization for these cases.
- If a node has an address configured, and it can prove that it owns this address, the binding is preserved regardless of any indication that a binding for the same source address could be configured in other SEND SAVI device. Bindings for the same source address in two or more SEND SAVI devices may occur due to several reasons, for example when a host moves (the two bindings exist just for a short period of time), or when many nodes generate the same address and the DAD procedure has failed. In these infrequent cases, SEND SAVI preserves connectivity for the resulting bindings.

We next describe how different inputs are processed depending on the state of the binding of the IP address 'IPaddr'. Note that every ND message is assumed to be validated according to SEND specification.

To facilitate the reader understanding the most relevant transitions of the SEND SAVI state machine, a simplified version, which does not contain every possible transition, is depicted in the next figure:



Simplified SEND SAVI state machine

Each state transition is characterized by any of the events which may trigger the change and the message(s) generated as a result of this change. The meaning of some terms are referred next:

- VP_DAD_NSOL as a triggering event means that a validated DAD_NSOL message has been received from the current BINDING_ANCHOR port VP.
- VP* means any packet (data packet) received from the current BINDING_ANCHOR port VP.
- TP_DAD_NSOL as a triggering event means that a DAD_NSOL message was received from a Trusted Port.
- - means that no message is sent. VP=VP' means that the BINDING_ANCHOR is set to VP'.

The notation

Timeout, TP_DAD_NSOL/VP_NUD_NSOL

means that the transition is triggered by either a timeout expiration or the reception of a DAD_NSOL message from a Trusted Port, and in addition to the transition, a NUD_NSOL message is sent through port VP.

For the rest of the description, we assume the following:

- When a validated message is required (i.e., a 'validated DAD_NSOL'), messages are check for validity in the considered switch according to [RFC3971], and messages not fulfilling these conditions are discarded.
- When any SEND message is received from a validated port, the SEND SAVI SHOULD assume that the message has been validated by the SEND SAVI device through which the message accessed to the SEND SAVI protection perimeter (unless the SEND SAVI perimeter has been breached), or the device generating it is trusted. In this case, the SAVI device does not perform any further validation. Performing validation for SEND messages received through a Trusted port may affect performance negatively.

NO_BIND

When the node is in this state, there are no unresolved NUD_NSOL messages generated by SEND SAVI or DAD_NSOL propagated to any Validating port, so the only relevant inputs are DAD_NSOL messages coming either from a Validating port (VP) or Trusted port (TP), or any packet other than DAD_NSOL coming from VP or TP. There are no timers configured for this state.

Messages received from a Validating port

- If a validated DAD_NSOL message is received from a Validating port VP, the SEND SAVI device forwards this message to all appropriate Trusted ports (the subset of Trusted ports which belong to the forwarding layer-2 topology, with the restrictions imposed by the MLD snooping mechanism, if applied). DAD_NSOL messages are not sent through any of the ports configured as Validating Ports. The SEND SAVI device sets the LIFETIME to TENT_LT, stores all the information required for future validation of the corresponding DAD_NADV message (such as the nonce of the message), creates a new entry in the SEND SAVI Data Base for IPaddr, sets BINDING_ANCHOR to VP, and changes the state to TENTATIVE_DAD. Creation time is set to the current value of the local clock.
  Note that in this case it is not possible to check address ownership by sending a NUD_NSOL because while the node is waiting for a possible DAD_NADV its address is in tentative state and the node cannot respond to NSOL messages [RFC4862].
- If any packet other than a DAD_NSOL is received through a Validating port VP, the SEND SAVI device issues a secured NUD_NSOL through port VP. The SEND SAVI device sets the LIFETIME to TENT_LT. The SEND SAVI device creates a new entry in the SEND SAVI Data Base for IPaddr, sets BINDING_ANCHOR to VP, and the state is changed to TENTATIVE_NUD. Creation time is set to the current value of the local clock. The SAVI device MAY discard the packet while the NUD procedure is being executed, or MAY store it in order to send it if the next transitions are (strictly) TENTATIVE_NUD and then VALID.

Messages received from a Trusted port

- If a DAD_NSOL message containing IPaddr as the target address is received through a Trusted port, it MUST NOT be forwarded through any of the Validating ports but it is sent through the proper Trusted ports. The state is not changed.
- Any packet other than a DAD_NSOL received from a Trusted port is forwarded to its destination. This packet is assumed to come from a SEND SAVI device that has securely validated the binding according to SEND SAVI rules (unless the SEND SAVI perimeter has been breached). The state is not changed.

TENTATIVE_DAD

To arrive to this state, the SEND SAVI device has received a validated DAD_NSOL coming from the BINDING_ANCHOR port and it has forwarded it to the appropriate TPs. The relevant events occurring in this state are: the reception of a DAD_NADV message from a TP, a DAD_NSOL message from the BINDING_ANCHOR port, other Validating port or TP, a data packet from the BINDING_ANCHOR port, and the expiration

of the LIFETIME timer initiated when the DAD_NSOL was received at port the BINDING_ANCHOR port.

Messages received from a trusted port

- The reception of a valid DAD_NADV message from a Trusted port indicates that the binding cannot be configured for the BINDING_ANCHOR port. The state is changed to NO_BIND, and the LIFETIME cleared.
- The reception of a valid DAD_NSOL from a Trusted port indicates that a node connected to another SEND SAVI device may be trying to configure the same address at the same time. The DAD_NSOL message is forwarded to the BINDING_ANCHOR port, so that the node at this port will not configure the address, as stated in [RFC4862]. The DAD_NSOL message is also forwarded to all appropriate Trusted ports. Then, the LIFETIME is cleared, and the state is changed to NO_BIND.
- Any packet other than a validated DAD_NSOL or DAD_NADV received from a Trusted port is forwarded to its destination. This packet is assumed to come from a SEND SAVI device that has securely validated the binding according to SEND SAVI rules (unless the SEND SAVI perimeter has been breached). The state is not changed.

Messages received from a validating port different from the BINDING_ANCHOR

- A validated DAD_NSOL is received from a Validating port VP' different the BINDING_ANCHOR port. The reception of a valid DAD_NSOL from port VP' indicates that a node connected to VP' may be trying to configure the same address at the same time. The DAD_NSOL message is forwarded to the BINDING_ANCHOR port, so that the node at this port will not configure the address, as stated in [RFC4862]. The DAD_NSOL message is also forwarded to all appropriate Trusted ports. Then, the BINDING_ANCHOR is set to VP' (through which the DAD_NSOL message was received), the LIFETIME is set to TENT_LT, and the state remains in TENTATIVE_DAD.
- Any other packet than a validated DAD_NSOL is received from a Validating port VP' different from the BINDING_ANCHOR port is discarded. The state is not changed.

Messages received from the BINDING_ANCHOR port

- If a validated DAD_NSOL is received from the BINDING_ANCHOR port, the LIFETIME is set to TENT_LT, and the state remains in TENTATIVE_DAD.
- If any packet other than a DAD_NSOL is received from the BINDING_ANCHOR port, it is assumed that the node has configured its address, although it has done it in less time than expected by the SEND SAVI device (less than TENT_LT). Since the node proved address ownership by means of the validated DAD_NSOL message, the LIFETIME is set to DEFAULT_LT, and the state is changed to VALID.

LIFETIME expires

- If LIFETIME expires, it is assumed that no other node has configured this address. Therefore, the Validating port VP (currently stored in the BINDING_ANCHOR) could be bound to this IPv6 address. The LIFETIME is set to DEFAULT_LT, and the state is changed to VALID.

VALID

To arrive to this state, the SEND SAVI device has successfully validated address ownership, and has created a binding for IPaddr. Relevant transitions for this state are triggered by the reception of DAD_NSOL from the BINDING_ANCHOR port, other Validating port or a TP, and any packet other than DAD_NSOL from other validating port than the BINDING_ANCHOR or a TP. The expiration of LIFETIME is also relevant to trigger a check for address ownership for the node at the BINDING_ANCHOR port.

Messages received from the BINDING_ANCHOR port

- If a validated DAD_NSOL with IPaddr as source address is received through the BINDING_ANCHOR port, it is forwarded to the appropriate Trusted ports. The LIFETIME is set to TENT_LT and the state is changed to TENTATIVE_DAD.
- Any packet other than a DAD_NSOL containing IPaddr as a source address arriving from the BINDING_ANCHOR port is forwarded appropriately. The state is not changed.

Messages received from a trusted port

- If a DAD_NSOL with IPaddr as source address is received through a Trusted port, the message is forwarded to VP. The LIFETIME is set to TENT_LT, a secured NUD_NSOL message is sent to IPaddr through VP and the state is changed to TESTING_VP.
- If any packet other than a DAD_NSOL with IPaddr as source address is received through a Trusted port, the packet is forwarded to VP and to other appropriate Trusted ports. A secured NUD_NSOL is sent to the BINDING_ANCHOR port, the LIFETIME is set to TENT_LT, and the state is changed to TESTING_VP.

Messages received from a validating port different from the BINDING_ANCHOR

- If a validated DAD_NSOL packet with IPaddr as source address is received through a Validating Port VP' (VP' different from the current BINDING ANCHOR), the message is forwarded to the BINDING_ANCHOR port. In addition, a secured NUD_NSOL is sent to BINDING_ANCHOR port, the ALTERNATIVE BINDING ANCHOR is set to port VP' (for future use if the node at VP' is finally selected), the

LIFETIME is set to TENT_LT, and the state is changed to TESTING_VP'.

- If any packet other than a DAD_NSOL with IPaddr as source address is received from a Validating port VP', different from the current BINDING_ANCHOR for this binding, VP, the packet is discarded. The SEND SAVI device MAY issue a secured NUD_NSOL through the BINDING_ANCHOR port, store VP' in the ALTERNATIVE BINDING ANCHOR for possible future use, set the LIFETIME to TENT_LT, and change the state to TESTING_VP'. An alternative to this behavior is that the SEND SAVI device MAY not do anything (in this case, the state would eventually change after a maximum DEFAULT_LT time, if the node at VP does not respond to a NUD_NSOL at TESTING_VP, the state is moved to NO_BIND). Then a packet arriving from VP' would trigger a process that may end up with binding for the node connecting to VP'.

LIFETIME expires

- If LIFETIME expires, a secured NUD_NSOL message is sent through the BINDING_ANCHOR port to IPaddr, the LIFETIME is set to TENT_LT, and the state is changed to TESTING_VP. In the TESTING_VP state packets are still being forwarded until the timer expires without receiving a NUD_NADV.

TESTING_VP

When the SEND SAVI device enters in the TESTING_VP state, the current Validating port is under check through a secured NUD_NSOL message generated by the SEND SAVI device. While testing, packets from the current Validating port are forwarded. Packets coming from Trusted ports are also forwarded. The relevant events for this state are the reception of a NUD_NADV message from VP, the reception of a DAD_NSOL message from VP, VP' or TP, the reception of any packet other than the previous cases from VP, VP' or TP, and the expiration of the timer associated to the reception of NUD_NADV.

Messages received from the BINDING_ANCHOR port

- If a validated NUD_NADV is received from VP, the LIFETIME is changed to DEFAULT_LT, and the state is changed to VALID. The message is not forwarded to any other port.
- If a validated DAD_NSOL message is received from VP, it is forwarded to the appropriate Trusted ports, the LIFETIME is set to DEFAULT_LT, and the state is changed to TENTATIVE_DAD.
- Any packet other than DAD_NSOL or NUD_NADV containing IPaddr as a source address arriving from the BINDING_ANCHOR port is forwarded. Neither the LIFETIME nor the state are changed.

Messages received from a trusted port

- If a DAD_NSOL packet is received from a Trusted port, the message is forwarded to VP and the appropriate Trusted ports. Neither the LIFETIME nor the state are changed. The node at the BINDING_ANCHOR port is under check: if it still is at this port, it should answer with a NUD_NADV, and also with a DAD_NADV. If it is not there, neither the NUD_NADV nor the DAD_NADV will be received, the timer will expire and the local state will move to NO_BIND.
- If a packet other than a DAD_NSOL arrives from a Trusted port, the packet is forwarded. Neither the LIFETIME nor the state are changed.

Messages received from a validating port different from the BINDING_ANCHOR

- If a valid DAD_NSOL is received from a Validating port VP' other than the current BINDING_ANCHOR port, the message is forwarded to the BINDING_ANCHOR port and to the appropriate Trusted ports. In addition, a secured NUD_NSOL is sent to the BINDING_ANCHOR port, the ALTERNATIVE BINDING ANCHOR is set to VP' (for future use if the node at VP' is finally selected), the LIFETIME is set to TENT_LT, and the state is changed to TESTING_VP'.
- Any other packet received from a Validating port VP' other than the BINDING_ANCHOR port is discarded. This may occur because the node has moved but have not issued a DAD_NSOL or the DAD_NSOL message has been lost. The state will eventually move to NO_BIND, and then the packets sent from VP' will trigger the creation of the binding for VP'.

LIFETIME expires

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

TESTING_VP'

To arrive to this state, the SEND SAVI device has received an indication that a node at VP' different from the BINDING_ANCHOR port wants to send data with IPaddr as source address occurred while a binding existed for VP. The port VP' which triggered the change of the state to TESTING_VP' was stored at the ALTERNATIVE_BINDING_ANCHOR, so that it can be retrieved if the node at VP' is determined as the legitimate owner of IPaddr. The SEND SAVI device has issued a NUD_NSOL to IPaddr through the BINDING_ANCHOR port. The relevant events that may occur in this case are the reception of a NUD_NADV from port VP (the BINDING_ANCHOR port), the reception of DAD_NSOL from VP, VP', TP and VP" (VP"

different from VP and VP'), the reception of any other packet from VP, VP', TP or VP", and the expiration of the timer.

Messages received from the BINDING_ANCHOR port

- A validated NUD_NADV is received from the BINDING_ANCHOR port. The reception of a valid NUD_NADV indicates that the node at VP is defending its address. The BINDING_ANCHOR in use is kept, the LIFETIME is set to DEFAULT_LT, and the state is changed to VALID.
- If a valid DAD_NSOL is received from the BINDING_ANCHOR port, it is forwarded to VP' (the port stored in the ALTERNATIVE_BINDING_ANCHOR). The BINDING_ANCHOR in use is kept, the LIFETIME is set to TENT_LT and the state is changed to TENTATIVE_DAD. When the DAD_NSOL message is received by the node at VP', this node is expected to unconfigure its address.
- Any packet other than a validated DAD_NSOL, or a validated NUD_NADV coming from the BINDING_ANCHOR port, is forwarded, and the state is not changed.

Messages received from the ALTERNATIVE_BINDING_ANCHOR validating port

- If a valid DAD_NSOL is received from the port stored in the ALTERNATIVE_BINDING_ANCHOR, it is forwarded to the BINDING_ANCHOR port. The BINDING_ANCHOR and the ALTERNATIVE BINDING ANCHOR are kept, the LIFETIME is set to DEFAULT_LT, and the state is not changed.
- Any packet other than a validated DAD_NSOL coming from the ALTERNATIVE_BINDING_ANCHOR port is discarded, and the state is not changed.

Messages received from a validating port different from the BINDING_ANCHOR and the ALTERNATIVE_BINDING_ANCHOR ports

- If a validated DAD_NSOL is received from port VP", different from BINDING_ANCHOR and the ALTERNATIVE_BINDING_ANCHOR ports, it is forwarded to the BINDING_ANCHOR and the ALTERNATIVE_BINDING_ANCHOR ports. The node at ALTERNATIVE BINDING ANCHOR port is expected to unconfigure its address if the message triggering the transition to this state was a DAD_NSOL message received from the ALTERNATIVE_BINDING_ANCHOR port (and not any other packet). The state remains in TESTING_VP' although VP" is stored in the ALTERNATIVE_BINDING_ANCHOR for future use if the node at VP" is finally selected. The LIFETIME is not changed.
- Any packet other than a validated DAD_NSOL received from port VP" is discarded and does not affect to the state.

Messages received from a trusted port

- If a DAD_NSOL is received from a Trusted port, the message is forwarded to the BINDING_ANCHOR and the ALTERNATIVE_BINDING_ANCHOR ports and other appropriate Trusted ports. The LIFETIME is left unchanged and the state is changed to TESTING_VP. The node at the

ALTERNATIVE_BINDING_ANCHOR port is expected to unconfigure its address if the packet triggering the transition to this state was a DAD_NSOL message received from the ALTERNATIVE_BINDING_ANCHOR port.

- Any packet other than a DAD_NSOL coming from a Trusted port is forwarded appropriately, but the state is not changed.

LIFETIME expires

- If LIFETIME expires, it is assumed that the node for which the binding existed is no longer connected through the BINDING_ANCHOR port. Therefore, the BINDING_ANCHOR is set to the ALTERNATIVE_BINDING_ANCHOR port value. The LIFETIME is set to DEFAULT_LT and the state is changed to VALID.

TENTATIVE_NUD

To arrive to this state, a data packet has been received through the BINDING_ANCHOR port without any existing binding in the SEND SAVI device. The SEND SAVI device has sent a NUD_NSOL message to the BINDING_ANCHOR port. The relevant events for this case are the reception of a NUD_NADV from port the BINDING_ANCHOR port; the reception of DAD_NSOL from the BINDING_ANCHOR port, other VP different from the BINDING_ANCHOR port, or a TP; and the reception of any packet other than DAD_NSOL and NUD_NADV from the BINDING_ANCHOR port, and other than DAD_NSOL for other VP different from the BINDING_ANCHOR port, or TP. In addition, the LIFETIME may expire.

Messages received from the BINDING_ANCHOR port

- If a validated NUD_NADV message is received through the BINDING_ANCHOR port, the LIFETIME is set to TENT_LT, and the state is changed to VALID. The message is not forwarded to any port.
- If a validated DAD_NSOL message is received through the BINDING_ANCHOR port, it is forwarded to the appropriate Trusted ports, the LIFETIME is set to TENT_LT and the state is changed to TENTATIVE_DAD.
- Any packet other than NUD_NADV or DAD_NSOL received through the BINDING_ANCHOR port is discarded.

Messages received from a validating port different from the BINDING_ANCHOR

- If a validated DAD_NSOL message is received through port VP' different from the BINDING_ANCHOR port, it is forwarded to the appropriate Trusted ports, the LIFETIME is set to TENT_LT, the BINDING_ANCHOR is set to VP', and the state is changed to TENTATIVE_DAD.

- Any packet other than validated DAD_NSOL received through port VP' MUST NOT be forwarded unless the next state for the binding is VALID. The packets received MAY be discarded or MAY be stored for being sent if the state changes later to VALID. The state is left unchanged.

Messages received from a trusted port

- If a DAD_NSOL message is received through a Trusted port, it is forwarded to the BINDING_ANCHOR port, and the state is left unchanged.
- Any other packet received from a Trusted port is forwarded appropriately. This packet may come from a SEND SAVI device that has securely validated the attachment of the node to its Validating port according to SEND SAVI rules. The state is left unchanged.

LIFETIME expires

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

3.4. SEND SAVI Port Configuration Guidelines

The detailed guidelines for port configuration in SEND SAVI devices are:

- Ports connected to another SEND SAVI devices MUST be configured as Trusted ports. Not doing so will prevent off-link traffic from being forwarded, along with the following effects for on-link traffic: increase significantly the CPU time, memory consumption and signaling traffic due to SEND SAVI validation, in both the SEND SAVI devices and the node whose address is being validated.
- 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 address.
- No more than one host SHOULD be connected to each port. Connecting more than one host to a port will allow hosts to generate packets with the same source address as the other hosts connected to the same port, and will allow performing replaying attacks as described in Section 5.1.
- Ports connected to routers MUST be configured as Trusted ports. Not doing so results in SEND SAVI devices discarding off-link traffic. Note that this means that, since routers are connected through Trusted ports, they can generate traffic with any source address, even those belonging to the link.
- Ports connected to a chain of one or more legacy switches that have other SEND SAVI devices but had no routers or hosts attached to them SHOULD be configured as Trusted ports. Not doing so will significantly increase the memory consumption in the SEND SAVI devices and increase the signaling traffic due to SEND SAVI

validation.

3.5. VLAN Support

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

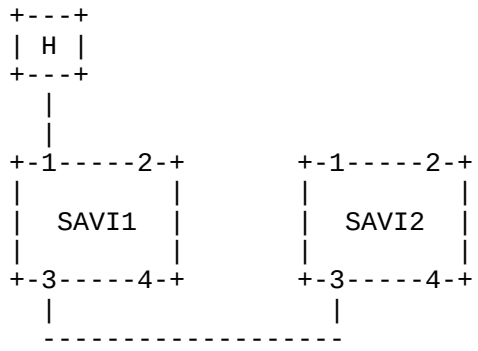
3.6. Protocol Constants

TENT_LT is 500 milliseconds.

DEFAULT_LT is 5 minutes.

4. Protocol Walkthrough

In this section we include two cases which illustrate the behavior of SEND SAVI, the change of the attachment port of a host, and the attack of a malicious host. We use the topology depicted in the following figure.



4.1. Change of the attachment point of a host

There are two cases, depending on whether the host H moves to a different port on the same switch, or to a different switch.

4.1.1. Moving to a port of the same switch

Host H is connected to port 1 of SAVI1 and moves to port 2 of the same switch. Before moving, the SEND SAVI state associated to IPH,

the IP address of H is

SAVI1=VALID, BINDING_ANCHOR=1 / SAVI2=NO_BIND

In the general case, H issues a DAD_NSOL message for IPH when it is connected to a different port. When SAVI1 receives this message, it validates it and changes its state to

SAVI1=TESTING_VP', BINDING_ANCHOR=1, ALTERNATIVE_BINDING_ANCHOR=2, TIMER=TENT_LT / SAVI2=NO_BIND

The DAD_NSOL message is propagated to port 1, because it is the current BINDING_ANCHOR, and the trusted port 3; but it is not propagated to Validating port 4. SAVI1 configures a timer for TENT_LT seconds. In addition, SAVI1 generates a NUD_NSOL and sends it through port 1. When SAVI2 receives this message through it trusted port, it discards it and remains in the NO_BIND state.

SAVI1 waits for a NUD_NADV message being received from port 1. Since there is no node attached to 1, there is no response for neither of these messages. When TENT_LT expires at SAVI1, the state changes to

SAVI1=VALID, BINDING_ANCHOR=2 / SAVI2=NO_BIND

If the node moving does not issue a DAD_NSOL when it attaches to port 2, then SAVI1 will receive a data packet through this port. The data packet is discarded, SAVI1 issues a secured NUD_NSOL through port 1, and changes the state to TESTING_VP'.

SAVI1=TESTING_VP', BINDING_ANCHOR=1, ALTERNATIVE_BINDING_ANCHOR=2 TIMER=TENT_LT / SAVI2=NO_BIND

SAVI1 waits for a NUD_NADV message being received from port 1. Since there is no node attached to 1, there is no response for neither of these messages. When TENT_LT expires at SAVI1, the state changes to

SAVI1=VALID, BINDING_ANCHOR=2 / SAVI2=NO_BIND

An alternative behavior allowed by the specification for the case in which the host does not issue a DAD_NSOL is that SAVI1 does nothing. In this case, after some time (bounded by DEFAULT_LT), the switch will change the state for IPH to TESTING_VP, check if H is still at port 1 (which is not), and move the state to NO_BIND. Then, a packet arriving from port 2 would trigger a process that finishes with a VALID stated with BINDING_ANCHOR=2.

4.1.2. Moving to a port of a different switch

Host H, connected to port 1 of SAVI1, moves to port 4 of SAVI2. Before moving, the SEND SAVI state associated to IPH, the IP address of H is

SAVI1=VALID, BINDING_ANCHOR=1 / SAVI2=NO_BIND

If H issues a DAD_NSOL message for IPH when it connects to port 4 of SAVI2, the state is changed to

SAVI1=VALID, BINDING_ANCHOR=1 / SAVI2=TENTATIVE_DAD, BINDING_ANCHOR=4, TIMER=TENT_LT

The DAD_NSOL message is propagated only through the trusted port of SAVI2. Then, SAVI1 changes its state as follows:

SAVI1=TESTING_VP, BINDING_ANCHOR=1, TIMER=TENT_LT / SAVI2=TENTATIVE_DAD, BINDING_ANCHOR=4, TIMER=TENT_LT

SAVI1 propagates the DAD_NSOL message to port 1. Since he only node which can answer with a secured DAD_NUD has moved, the timer at SAVI2 expires, and SAVI2 changes its state to VALID:

SAVI1=TESTING_VP, BINDING_ANCHOR=1, TIMER=TENT_LT / SAVI2=VALID, BINDING_ANCHOR=4

Just a very short time after, the timer at SAVI1 expires, and the state changes to NO_BIND.

SAVI1=NO_BIND / SAVI2=VALID, BINDING_ANCHOR=4

If host H does not send a DAD_NSOL when it moves to SAVI2, but a data packet, SAVI2 changes its state to TENTATIVE_NUD.

SAVI1=VALID, BINDING_ANCHOR=1 / SAVI2=TENTATIVE_NUD, BINDING_ANCHOR=4, TIMER=TENT_LT

SAVI2 issues a secured NUD_NSOL through port 4. H is assumed to have the address configured (otherwise it should not have generated a data packet), so it can respond with a NUD_NADV. When SAVI1 receives the NUD_NADV and validates it, the state is changed to VALID

SAVI1=VALID, BINDING_ANCHOR=1 / SAVI2=VALID, BINDING_ANCHOR=4

After some time (bounded by DEFAULT_LT), the state in SAVI1 will expire, and SAVI1 will perform a check for host H.

SAVI1=TESTING_VP, BINDING_ANCHOR=1, TIMER=TENT_LT / SAVI2=VALID, BINDING_ANCHOR=4

SAVI1 issues a NUD_NSOL through port 1 for IPH. No response is received in this case, so SAVI1 changes its state to NO_BIND

SAVI1=NO_BIND / SAVI2=VALID, BINDING_ANCHOR=4

4.2. Attack of a malicious host

Host H is attached to the SEND SAVI infrastructure through port 1 of SAVII1. We consider that host M starts sending data packets using IPH (the IP address of H) as source address, without issuing a DAD_NSOL (a similar analysis can be done for this case).

4.2.1. M attaches to the same switch as the victim's switch

The initial state before the attack of M is:

SAVI1=VALID, BINDING_ANCHOR=1 / SAVI2=NO_BIND

M attaches to port 2 of SAVI1, and starts sending data packets. When SAVI1 receives the data packet, the packet is discarded. SEND SAVI may issue a secured NUD_NSOL through port 1, and changes the state to

SAVI1=TESTING_VP', BINDING_ANCHOR=1, ALTERNATIVE_BINDING_ANCHOR=2, TIMER=TENT_LT / SAVI2=NO_BIND

Host H is still attached to port 1, so it receives the NUD_NSOL and responds with a secured NUD_NADV. SAVI1 receives this message, validates it and changes its state again to

SAVI1=VALID, BINDING_ANCHOR=1 / SAVI2=NO_BIND

To prevent the drain of CPU resources in SAVI1, the processing of further packets received from port 2 may be rate-limited, as discussed in Section 5.2.

An alternative to the previous behavior is that SAVI1 does nothing when node M starts sending packets from port 2. In this case, when the timer to renew the state triggers (this time is bounded by DEFAULT_LT), SAVI1 moves the state to TESTING_VP, sends a NUD_NSOL through port 1, host H responds, and the state remains in VALID for BINDING_ANCHOR=1. In this way, communication of host H is also defended.

4.2.2. M attaches to a different switch to the victim's switch

The initial state before the attack of M is:

SAVI1=VALID, BINDING_ANCHOR=1 / SAVI2=NO_BIND

M attaches to port 2 of SAVI2, and starts sending data packets. When SAVI2 receives the data packet, it changes the state to

SAVI1=VALID, BINDING_ANCHOR=1 / SAVI2=TENTATIVE_DAD, BINDING_ANCHOR=2, TIMER=TENT_LT

SAVI2 issues a secured NUD_NSOL through port 2. Since M does not own the IPH CGA, it cannot respond to the message. When the timer expires, the state is moved back to:

SAVI1=VALID, BINDING_ANCHOR=1 / SAVI2=NO_BIND

To prevent the drain of CPU resources in SAVI2, the processing of further packets received from port 2 may be rate-limited, as discussed in Section 5.2.

5. Security Considerations

SEND SAVI operates only with validated SEND messages to create bindings. Note that IPv6 packets generated by non-SEND nodes will be discarded by the first SEND SAVI device receiving it. Therefore, attackers cannot obtain any benefit by not using SEND. In order to perform address validation in a mixed scenario comprising SEND and non-SEND devices, a different solution is required, which should be addressed in other document.

Nodes MUST NOT assume that all SEND messages received from a SEND SAVI device are validated, since these devices only validate the messages strictly required for SEND SAVI operation. Among the number of messages which are not validated by SEND SAVI, we can name NUD_NSOL messages generated by other nodes and its corresponding NUD_NADV responses, or RSOL messages.

SEND SAVI improves protection compared to conventional SAVI, as a result of the increased ability of SEND nodes to prove address ownership.

A critical security consideration regarding to SEND SAVI deals with the need of proper configuration of the roles of the ports in a SEND SAVI deployment scenario. Regarding to security, the main requirement is that ports defining the protected perimeter SHOULD be

configured as Validating ports. Not doing so will allow an attacker sending packets using any source address, regardless of the bindings established in other SEND SAVI devices.

5.1. Protection Against Replay Attacks

One possible concern about SEND SAVI is its behavior when an attacker tries to forge the identity of a legitimate node by replaying SEND messages used by the SEND SAVI specification. An attacker could replay any of these messages to interfere with SEND SAVI operation. For example, it could replay a DAD_NSOL message to abort the configuration of an address for a legitimate node and to gain the right to use the address for DEFAULT_LT seconds.

We can analyse two different cases when considering SEND SAVI replay attacks:

- When the SEND message replayed is used to create or update binding information for SEND SAVI, since the port through which this message is received is key to SEND SAVI operation. SEND SAVI creates and maintains bindings as a result of the reception of DAD_NSOL messages and of the exchange of NUD_NSOL/NUD_NADV messages.
- When the SEND message replayed does not result in the update of binding information for SEND SAVI, and thus it is not related to the specific port through which it was received. Such situations are the reception of CPA messages containing certificates, and the processing of an RADV message coming from a Trusted port, which can be used in SEND SAVI to populate the SEND SAVI Prefix list. In these two cases, the security risks are equivalent to those of SEND operation, i.e., we can consider that the information will not be changed by its legitimate sender for the time during which the SEND specification allows replaying (which depends on the values of TIMESTAMP_FUZZ and TIMESTAMP_DRIFT, [RFC3971]).

For replay of messages belonging to the second case, i.e., messages which do not result in changes in the SEND SAVI binding information, the security provided by SEND is sufficient. For the replay of messages belonging to the first case, DAD_NSOL and NUD_NSOL/NUD_NADV messages, protection results from the behavior of SEND SAVI, specified in Section 3.3.2, which restricts the ports to which the messages involved in SEND SAVI binding updates are disseminated. SEND SAVI devices only forward these messages to ports for which a binding to the address being tested by the DAD_NSOL message existed. Therefore, it is not enough for an attacker to subscribe to a Solicited Node address to receive DAD_NSOL messages sent to that address, but the attacker needs to generate a valid DAD_NSOL message associated to the address for which the binding is being tested, which is deemed unfeasible [RFC3971].

5.2. Protection Against Denial of Service Attacks

The attacks against the SEND SAVI device basically consist of making the SEND 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 SEND SAVI device create state for each of the addresses and waste memory. At some point, the SEND 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 SEND SAVI device runs out of the memory allocated for the SEND SAVI Data Base, 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 DEFAULT_LT, which is updated by NUD_NSOL/NUD_NADV exchange. The result is that in order for an attacker to actually fill the SEND SAVI Data Base with false source addresses, it needs to continuously answer to NUD_NSOL 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 also RECOMMENDED that a SEND 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 REQUIRED minimum is the memory needed to store four bindings associated to the port, although it SHOULD be raised if the ratio between the maximum number of bindings allowed in the device and the number of ports is high. The motivation for setting a minimum number of bindings per port is as follows. An attacker attached to a given port of a SEND SAVI device may attempt to launch a DoS attack towards the SEND SAVI device by creating many bindings for different addresses. It can do so, by sending DAD_NSOL for different addresses. The result is that the attack will consume all the memory available in the SEND SAVI device. The above recommendation aims to reserve a minimum amount of memory per port, so that nodes 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.

The SEND SAVI device may store data packets while the address is being verified, for example, when a DAD_NSOL was lost before arriving to the SEND SAVI device to which the host attaches, and the host sends data packets, these data packets may be stored until the SEND SAVI device verifies the binding by means of a NUD packet exchange. In this case, the memory for data packet storage may also be a target

of DoS attacks. A SEND SAVI device MUST limit the amount of memory used to store data packets, allowing the other functions (such as being able to store new bindings) to have available memory even in the case of an attack such those described above.

It is worth to note that the potential of Denial of Service attacks against the SEND SAVI network is increased due to the use of costly cryptographic operations in order to validate the address of the nodes. An attacker could generate packets using new source addresses in order to make the closest SEND SAVI device spend CPU time to validate DAD_NSOL messages or to generate a secure NUD_NSOL. This attack can be used to drain CPU resources of SEND SAVI devices with a very low cost for the attacker. In order to solve this problem, rate-limiting the processing of packets which trigger SEND SAVI events SHOULD be enforced in a per-port basis.

5.3. Considerations on the deployment model for trust anchors

The SEND specification [RFC3971] proposes two deployment models for trust anchors, either a centralized model relaying on a globally rooted public key infrastructure, or a more local, decentralized deployment model, in which end hosts are configured with a collection of public keys which are trusted only on a domain.

The appeal of a centralized model is the possibility for hosts to use SEND to validate routers as they move through links belonging to different organizations without additional configuration. However, without any further protection, it also enables routers authorized with a certificate path rooted on a global trust anchor to appear as legitimate routers in a link in which they were not intended to act as such. This threat already existed for SEND deployments, for which links configured to accept centralized trust anchors may send outgoing traffic and use prefix information from alien routers. In a SEND SAVI deployment, such routers may be able to deliver off-link traffic to any node of the link.

In order to cope with this threat, SEND SAVI specifies that nodes are only allowed to behave as routers if they connect through Trusted ports. In particular, RADV messages and traffic with off-link source addresses are discarded when received through Validating ports, which are the ports intended for non-trusted infrastructure, as moving nodes. The protection provided by filtering RADV messages prevents SEND nodes from identifying alien routers as legitimate routers, even though the trust anchor of these routers is valid.

Besides, it is worth to say that SEND SAVI supports a decentralized deployment model.

5.4. Residual threats

SEND SAVI assumes that a host will be able to defend its address when the DAD procedure is executed for its addresses, and that it will answer to a NUD_NSOL with a NUD_NADV when required. 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). If the SEND SAVI device does not see the DAD_NADV or the NUD_NADV, 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 launching a DoS attack to the host that will prevent it to issue proper replies.

5.5. Privacy considerations

A SEND SAVI device MUST delete binding anchor information as soon as possible (i.e., as soon as the state for a given address is back to NO_BIND), 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 about the majority of hosts that never spoof SHOULD NOT be logged.

6. IANA Considerations

This document has no actions for IANA.

7. Acknowledgments

Thanks to Jean-Michel Combes, Ana Kukec, Ted Lemon, Adrian Farrel, Barry Leiba, Brian Haberman, Vicent Roca and Benoit Claise for their review and comments on this document. The text has also benefited from feedback provided by Tony Cheneau and Greg Daley.

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

8. References

8.1. Normative References

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

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

[RFC3972] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, March 2005.

[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.

8.2. Informative References

[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.

[RFC6434] Jankiewicz, E., Loughney, J., and T. Narten, "IPv6 Node Requirements", RFC 6434, December 2011.

[RFC6620] Nordmark, E., Bagnulo, M., and E. Levy-Abegnoli, "FCFS SAVI: First-Come, First-Served Source Address Validation Improvement for Locally Assigned IPv6 Addresses", RFC 6620, May 2012.

[RFC7039] Wu, J., Bi, J., Bagnulo, M., Baker, F., and C. Vogt, "Source Address Validation Improvement (SAVI) Framework", RFC 7039, October 2013.

[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.

Authors' Addresses

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

Alberto Garcia-Martinez
Universidad Carlos III de Madrid
Av. Universidad 30
Leganes, Madrid 28911
SPAIN

Phone: 34 91 6248782
Email: alberto@it.uc3m.es
URI: http://www.it.uc3m.es