

Source Address Validation
Improvements WG
Internet-Draft
Intended status: Standards Track
Expires: January 6, 2011

Y. Ding
R. Zheng
Y. Li
Huawei Technologies
July 5, 2010

SAVI analysis for PANA with SLACC
draft-ding-savi-pana-with-slacc-00

Abstract

This document analysis the source address vilidation in PANA with slacc,and specifies the procdure for binding assigned address to the UE through PANA related mechanism.

Status of this Memo

This Internet-Draft is submitted to IETF 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 January 6, 2011.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Message Flow	4
3. Security Considerations	8
4. IANA Considerations	8
5. References	8
5.1. Normative Reference	8
5.2. Informative References	8
Authors' Addresses	9

1. Introduction

In IP access network, IP edge device is communated with lots of subscribers, which include home gateways and hosts. In order to keep accurate information of these device, IP edge has to execute source address validation to make the information related to the subscribers processed correctly. In IPv6 access network, subscriber may use LLA (Link Local Address) or ULA (Unique Local Address) to initiate the subscriber authentication. The general approach to obtain a GUA (Global Unique Address) address for a subscriber is to make the home gateway get a delegated prefix and then home gateway advertises the prefix to UEs in home network. Subscriber generates GUA via stateless address configuration. Figure 1 shows a residential IPv6 broadband access network which uses DHCP PD[RFC3633] to get the delegated prefix and SLACC to obtain the GUA address.

HGW gets a delegated prefix, say /56, for its home network use. When UE1 tries to connect to the network, it first gets its own LLA/ULA address and then uses that address as source IP address for the following PANA authentication for subscriber verification. When the authentication succeeds, it sends RS (router solicitation) to HGW and HGW replies with RA (router advertisement) with a /64 prefix option for SLACC configuration. UE1 uses the prefix to generate its GUA address. And it uses GUA for the following data transportation. When another UE2 tries to connect to the network, it repeats the step 2 to 7. However it should be noticed, /64 prefix that HGW sent to UE2 visa RA may not be the same one as that sent to UE1. As IP edge only knows /56 it delegated to HGW, there is no native way for IP edge to know which address/prefix UE1 and UE2 used within the range of the delegated /56 prefix. As IP session terminates on IP edge, IP edge should have the detailed information stored for each session, e.g. prefix, address, layer 2 information, etc. If IP edge wants to treat the connections which terminate on UE1 and UE2 as different sessions, it needs to know the specific information of each to set up correct binding relationship. This contribution tries to analysis the issue and provide some possible ways to let the subscriber's address information get validated and let the IP edge know the SLACC configuration in home network via PANA,.

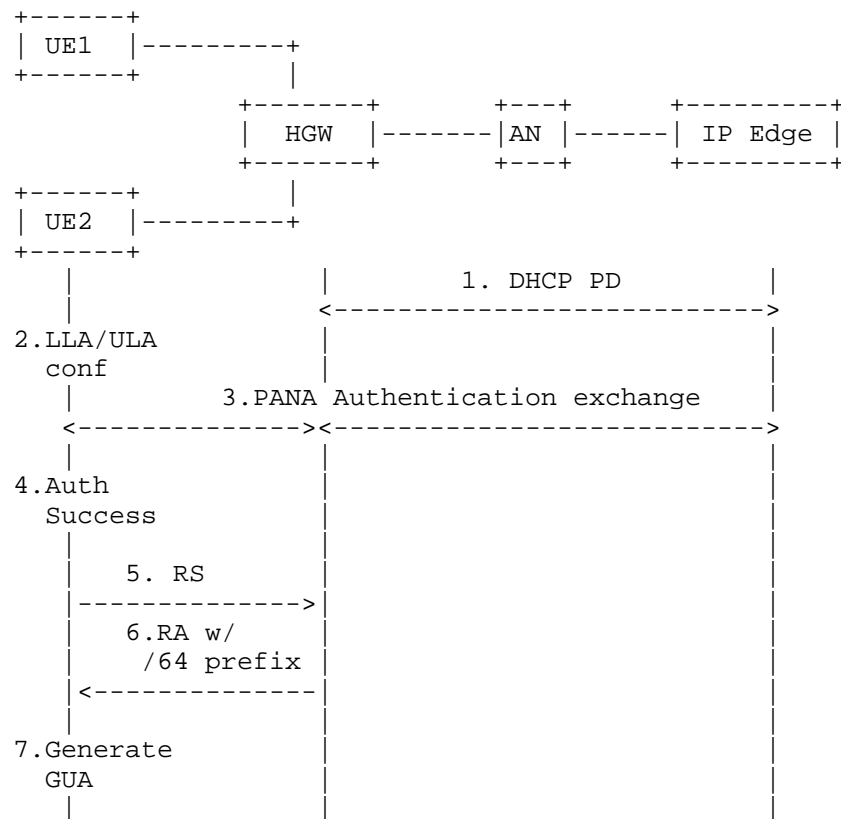


Figure 1: IPv6 Broadband Access Network

2. Message Flow

The problem stated in Introduction section indeed is a special case of IP address/prefix reconfiguration. Section 3 of [RFC5193] briefly described the possible use cases of IP reconfiguration without giving the detailed PANA flow. To implement IPv6 session information binding in broadband access scenario, Figure 2 shows the message flow to support subscriber authentication and SLACC configuration in home network. IP edge device retrieves the relevant information from the process and performs the necessary session bindings.

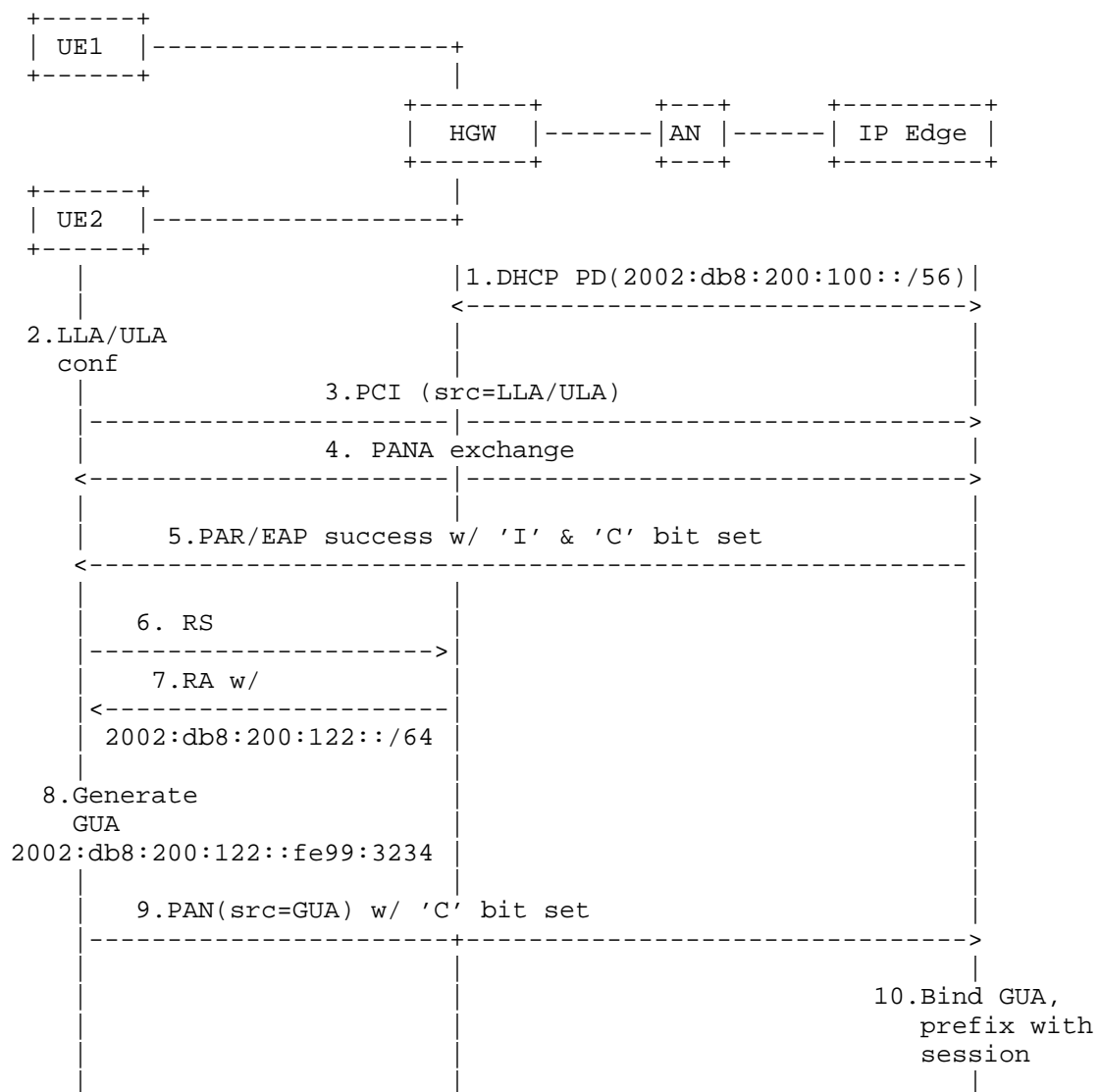


Figure 2: Message flow of IP/prefix reconfiguration in home network - 1

In step 5, PAR(PANA-Auth-request) with EAP success payload is sent to UE. 'I' bit was set to indicate that UE is required to get a GUA and use that GUA for the following message exchange. When receiving the PAR with EAP success, UE starts the SLACC procedures to get the GUA address for data communication. It send the RS(router solicitation) to HGW in step 6. Then in step 7, HGW sends responded RA with

advertised prefix to UE. The advertised prefix is within the range of the delegated /56 prefix in step 1 but is with 64-bit length. UE uses the received /64 prefix to generate a GUA in step 8, which is to be used in the data communication. In step 9 UE sends PAN(PANA-Auth-Answer) with 'C' bit set to IP edge. GUA address generated in step 8 should be used as the source IP address. When IP edge receives PAN, it retrieves the GUA and prefix information and binds them with the session initiated by UE. PANA session ID can be used for the matching of the LLA/ULA and GUA to set up the binding relationship.

With the approach shown in Figure 2, IP edge is able to get the specific address/prefix information of connected UE with embedded mechanism of PANA. It is a light-weighted solution for IPv6 session information retrieval and binding in broadband access network.

There is also another approach to solve this problem in the following figure, the address UE use in data communication may be allocated after authentication process finished, Figure 3 shows the message flow.

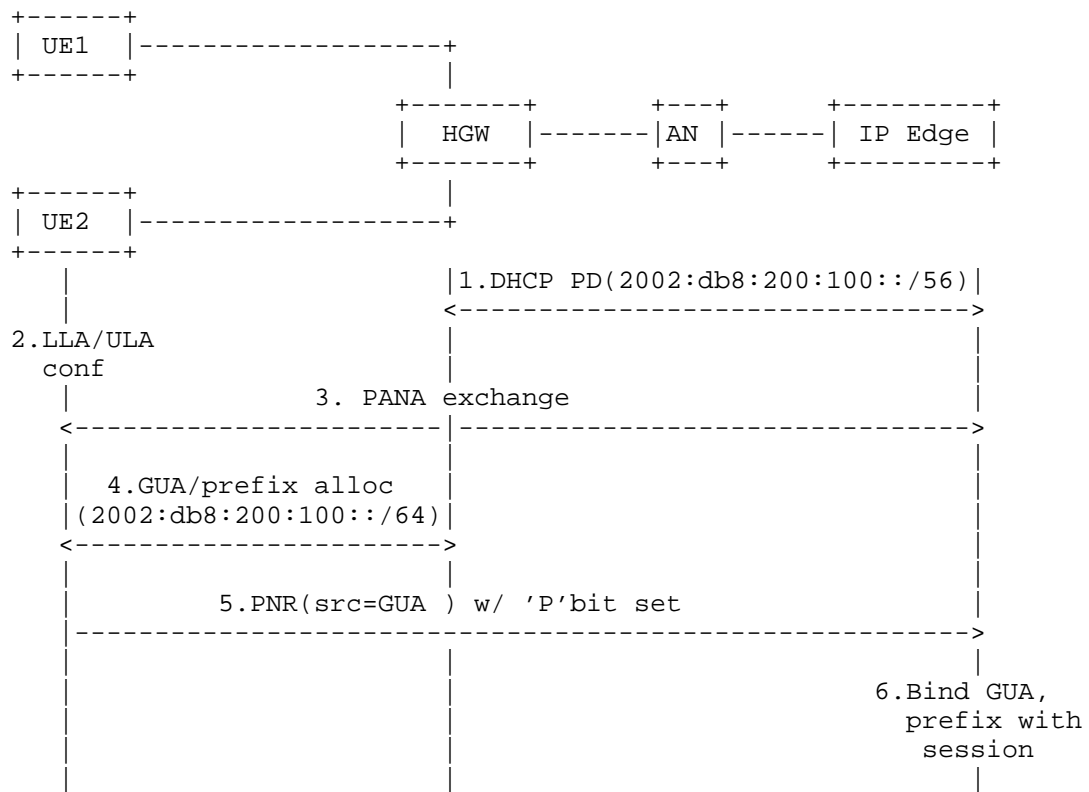


Figure 3: Message flow of IP/prefix reconfiguration in home network -
2

UE use the LLA/ULA as source address to complete the authentication exchange with IP edge in step 3. After this procedure, in step 4, HGW allocate the GUA address to UE, or the prefix within the range of the delegated /56 prefix in step 1 but is with 64-bit length. When received the prefix with 64-bit length, UE use it to generate a GUA. In step 5 UE sends PNR(PANA-Notification-Request) with 'P' bit set to IP edge, 'P' bit was set to indicate doing the Ping operation between PANA peers. GUA address generated in step 4 should be used as the source IP address. When IP edge receives PNR, it retrieves the GUA and prefix information and binds them with the session initiated by UE. PANA session ID can be used for the matching of the LLA/ULA and GUA to set up the binding relationship.

With this approach shown in Figure 3, IP edge is able to get the specific address/prefix information of connected UE with mechanism of PANA. It is another solution for IPv6 session information retrieval and binding in broadband access network.

3. Security Considerations

There is no extra security vulnerability introduced by this contribution. AUTH AVP is used to integrity protect PANA messages when last PAN is sent and source IP address has been switched from LLA/ULA to GUA address.

4. IANA Considerations

There is no new IANA code required to be allocated.

5. References

5.1. Normative Reference

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

5.2. Informative References

- [RFC3633] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, December 2003.
- [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.
- [RFC5191] Forsberg, D., Ohba, Y., Patil, B., Tschofenig, H., and A. Yegin, "Protocol for Carrying Authentication for Network Access (PANA)", RFC 5191, May 2008.
- [RFC5193] Jayaraman, P., Lopez, R., Ohba, Y., Parthasarathy, M., and A. Yegin, "Protocol for Carrying Authentication for Network Access (PANA) Framework", RFC 5193, May 2008.
- [TR-059] DSL Forum, "DSL Evolution - Architecture Requirements for the Support of QoS-Enabled IP Services", TR 059, September 2003.
- [TR-101] DSL Forum, "Migration to Ethernet Based DSL Aggregation", TR 101, April 2006.

Authors' Addresses

Yilan Ding
Huawei Technologies
Huawei Nanjing R&D Center, 101 Software Avenue
Nanjing 210012
China

Phone: +86-25-56622346
Email: denver@huawei.com

Ruobin Zheng
Huawei Technologies
Huawei Industrial Base
Shenzhen 518129
China

Phone: +86-755-28973567
Email: robin@huawei.com

Yizhou Li
Huawei Technologies
Huawei Nanjing R&D Center, 101 Software Avenue
Nanjing 210012
China

Phone: +86-25-56622310
Email: liyizhou@huawei.com

SAVI
Internet Draft
Intended status: Standard Tracks
Expires: March 2011

J. Bi, J. Wu
CERNET
G. Yao
Tsinghua Univ.
F. Baker
Cisco
September 8, 2010

SAVI Solution for DHCP
draft-ietf-savi-dhcp-06.txt

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, except to publish it as an RFC and to translate it into languages other than English.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

This Internet-Draft will expire on January 8, 2010.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Abstract

This document specifies the procedure for creating bindings between a DHCPv4 [RFC2131]/DHCPv6 [RFC3315] assigned source IP address and a binding anchor (refer to [SAVI-framework]) on SAVI (Source Address Validation Improvements) device. The bindings can be used to filter packets generated on the local link with forged source IP address.

Table of Contents

Copyright Notice	2
Abstract	2
1. Introduction	3
2. Conventions used in this document.....	4
3. Mechanism Overview	4
4. Terminology	4
5. Conceptual Data Structures.....	4
5.1. Control Plane Data Structure: Binding State Table(BST) ..	4
5.2. Data Plane Data Structure: Filtering Table(FT).....	5
6. DHCP Scenario	5
7. Binding Anchor Attributes.....	6
7.1. No Attribute	6
7.2. SAVI-Validation Attribute.....	6
7.3. SAVI-DHCP-Trust Attribute.....	7
7.4. SAVI-SAVI Attribute.....	7
7.5. SAVI-BindRecovery Attribute.....	7
7.6. SAVI-ExtSnooping Attribute.....	7
8. Binding Set Up	7
8.1. Rationale	8

8.2. Binding States Description.....	8
8.3. Events	8
8.3.1. Timer expiration event.....	8
8.3.2. Control message arriving event.....	8
8.4. Process of Control Packet Snooping.....	9
8.4.1. From INIT to other states.....	9
8.4.1.1. Trigger Event.....	9
8.4.1.2. Following Actions.....	10
8.4.2. From START to other states.....	11
8.4.2.1. Trigger Event.....	11
8.4.2.2. Following Actions.....	11
8.4.3. From BOUND to other states.....	12
8.4.3.1. Trigger Event.....	12
8.4.3.2. Following Actions.....	12
8.5. State Machine of DHCP Snooping.....	12
9. Supplemental Binding Process: Handling Link Topology Change.	13
9.1. Binding Recovery Process.....	14
9.2. Extended Control Packet Snooping Process.....	15
10. Filtering Specification.....	16
10.1. Data Packet Filtering.....	16
10.2. Control Packet Filtering.....	16
11. Handle Binding Anchor Off-link Event.....	17
12. Binding Number Limitation.....	17
13. State Restoration	17
14. Confirm Triggered Binding.....	18
15. Consideration on Link Layer Routing Complexity.....	18
16. Duplicate Bindings of Same Address.....	19
17. Constants	19
18. Security Considerations.....	19
19. IANA Considerations.....	19
20. References	19
20.1. Normative References.....	19
20.2. Informative References.....	19
21. Acknowledgments	20
22. Change Log	21

1. Introduction

This document describes the procedure for creating bindings between DHCP assigned addresses and a binding anchor (refer to [savi-framework]). Other related details about this procedure are also specified in this document.

These bindings can be used to filter packets with forged IP address. Section 12 suggests usage of these bindings for common practice. [savi-framework] may specify different usages of binding, depending

on the environment and configuration. The definition and examples of binding anchor is specified in [savi-framework].

The binding process is inspired by the work of IP Source Guard [IP Source Guard].

In a stateless DHCP scenario [RFC3736], DHCP is used to configure other parameters but rather IP address. The address of the client SHOULD be bound based on other SAVI solutions, but rather this solution designed for stateful DHCP.

2. Conventions used in this document

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. Mechanism Overview

The mechanism specified in this document is designed to provide an address level source IP address validation granularity, as a supplement to BCP38 [BCP38]. This mechanism is deployed on the access device (including access switch, wireless access point/controller, etc), and performs mainly DHCP snooping to set up bindings between DHCP assigned IP addresses and corresponding binding anchors. The bindings can be used to validate the source address in the packets.

4. Terminology

Main terms used in this document are described in [savi-framework], [RFC2131] and [RFC3315].

5. Conceptual Data Structures

This section describes the possible conceptual data structures used in this mechanism.

Two main data structures are used to record bindings and their states respectively. There is redundancy between the two structures, for the consideration of separation of data plane and control plane.

5.1. Control Plane Data Structure: Binding State Table (BST)

This table contains the state of binding between source address and binding anchor. Entries are keyed on the binding anchor and source IP address. Each entry has a lifetime field recording the remaining lifetime of the entry, a state field recording the state of the

binding and a field recording other information. The lifetime field is used to help remove expired bindings. The state field is used to identify state. The other field is used to keep temporary information, e.g., the transaction ID in DHCP request. Before a binding is finished, the lease time of the address is also kept in this field because it is improper to keep it in the lifetime field which keeps the lifetime of the binding entry but not the address.

Anchor	Address	State	Lifetime	Other
A	IP_1	Bound	65535	
A	IP_2	Bound	10000	
B	IP_3	_Start	1	

Figure 1 Instance of BST

5.2. Data Plane Data Structure: Filtering Table (FT)

This table contains the bindings between binding anchor and address, keyed on binding anchor and address. This table doesn't contain any state of the binding. This table is only used to filter packets. An Access Control List can be regarded as a practical instance of this table.

Anchor	Address
A	IP_1
A	IP_2

Figure 2 Instance of FT

6. DHCP Scenario

Figure 3 shows the main elements in a DHCP enabled network. At least one DHCP server must be deployed in the network, and DHCP relay may be used to relay message between client and server.

Other address assignment mechanisms may be also used in such network. However, this solution is primarily designed for a pure DHCP scenario, in which only DHCP servers can assign valid global address. In a mixed address assignment scenario where multiple address assignment

methods such as DHCPv6 and SLAAC, or DHCPv4 and manually configured assign addresses that share the common prefix, the SAVI device may need additional state in the state machine to detect and avoid address conflict. The SAVI solution for mixed environment is proposed in a separate document [draft-bi-savi-mixed].

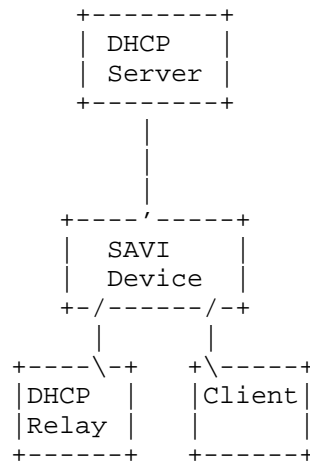


Figure 3 DHCP Scenario

7. Binding Anchor Attributes

This section specifies the binding anchor attributes involved in this mechanism.

Binding anchor is defined in the [savi-framework]. Attribute of each binding anchor is configurable. In default, binding anchor has no attribute. A binding anchor MAY be configured to have one or more compatible attributes. However, a binding anchor MAY have no attribute.

7.1. No Attribute

By default, a binding anchor has no attribute. Server type DHCP message from binding anchor with no attribute MUST be dropped. However, other packets SHOULD NOT be dropped.

7.2. SAVI-Validation Attribute

SAVI-Validation attribute is used on binding anchor on which the source addresses are to be validated. The filtering process on binding anchor with such attribute is described in section 13.

7.3. SAVI-DHCP-Trust Attribute

SAVI-DHCP-Trust Attribute is used on binding anchor on the path to a trustable DHCP server/relay.

DHCP server/relay message coming from binding anchor with this attribute will be forwarded.

7.4. SAVI-SAVI Attribute

This attribute is used on binding anchor from which the traffic is not to be checked. All traffic from binding anchor with this attribute will be forwarded without check. Note that DHCP server message and router message will also be trusted.

Through configuring this attribute on binding anchor that joins two or more SAVI devices, SAVI-Validation and SAVI-SAVI attributes implement the security perimeter concept in [savi-framework]. Since no binding entry is needed on such binding anchor, the binding entry resource requirement can be reduced greatly.

This attribute can also be set on other binding anchors if the administrator decides not to validate the traffic from the binding anchor.

This attribute is mutually exclusive with SAVI-Validation.

7.5. SAVI-BindRecovery Attribute

This attribute is used on binding anchor that requires binding recovery described in section 10.1.

This attribute is mutually exclusive with SAVI-SAVI.

7.6. SAVI-ExtSnooping Attribute

This attribute is used on binding anchor that requires extended control packet snooping described in section 10.2.

This attribute is mutually exclusive with SAVI-SAVI.

8. Binding Set Up

This section specifies the procedure of setting up bindings based on control packet snooping. The binding procedure specified here is exclusively designed for binding anchor with SAVI-Validation attribute.

8.1. Rationale

The rationale of this mechanism is that if a node attached to a binding anchor intends to use a valid DHCP address, the DHCP procedure which assigns the address to the node goes first on the same binding anchor. This basis stands when the link layer routing is stable. However, unstable link layer routing may result in that data packet is received from a different binding anchor with the DHCP messages. Infrequent link layer path change can be handled (but not perfectly) by the mechanism described in section 10. Section 15 discusses the situation that link layer routing is naturedly unstable. To handle this situation is above the scope of this document.

8.2. Binding States Description

This section describes the binding states of this mechanism.

INIT The state before a binding has been set up.

START A DHCP request (or a DHCPv6 Confirm, or a DHCPv6 Solicitation with Rapid Commit option) has been received from host, and it may trigger a new binding.

BOUND The address is authorized to the client.

8.3. Events

8.3.1. Timer expiration event

EVE_ENTRY_EXPIRE: The lifetime of an entry expires

8.3.2. Control message arriving events

EVE_DHCP_REQUEST: A DHCP Request message is received from a binding anchor with SAVI-Validation attribute, and the binding entry limit on the binding anchor has not been reached.

EVE_DHCP_CONFIRM: A DHCPv6 Confirm message is received from a binding anchor with SAVI-Validation attribute, and the binding entry limit on the binding anchor has not been reached.

EVE_DHCP_OPTION_RC: A DHCPv6 Solicitation message with Rapid Commit option is received from a binding anchor with SAVI-Validation attribute, and the binding entry limit on the binding anchor has not been reached.

EVE_DHCP_REPLY: A DHCPv4 Acknowledgement or DHCPv6 Reply message is received from a binding anchor with SAVI-DHCP-Trust attribute, and the message should be forwarded to a binding anchor with SAVI-Validation attribute, which has an entry in the state of START. The TID field in the entry matches the TID in the message.

EVE_DHCP_REPLY_NULL: A DHCPv4 Acknowledgement or DHCPv6 Reply message is received from a binding anchor with SAVI-DHCP-Trust attribute, and the message should be forwarded to a binding anchor with SAVI-Validation attribute, which has no entry in the state of START or matches the TID field.

EVE_DHCP_DECLINE: A DHCP Decline message is received from a binding anchor with SAVI-Validation attribute. The message declines an address bound with the binding anchor in state of LIVE or DETECTION or BOUND.

EVE_DHCP_RELEASE: A DHCP Release message is received from a binding anchor with SAVI-Validation attribute. The message releases an address bound with the binding anchor in state of LIVE or DETECTION or BOUND.

EVE_DHCP_REPLY_RENEW: A DHCPv4 Acknowledgement or DHCPv6 Reply message is received, which suggests a new lease time of address in state of BOUND.

8.4. Process of Control Packet Snooping

8.4.1. From INIT to other states

8.4.1.1. Trigger Event

EVE_DHCP_REQUEST, EVE_DHCP_CONFIRM, EVE_DHCP_OPTION_RC,
EVE_DHCP_REPLY_NULL.

Note that vulnerability may be caused by DHCP Reply triggered initialization. The binding of assigned address and binding anchor may be threatened if the binding mechanism between binding anchor and link layer address is not secure. If one of the following conditions is satisfied, the security can be ensured.

1. Option 82 is used to keep binding anchor in DHCP Request and Reply, or
2. Unspoofable MAC is used as binding anchor(802.11i,802.1ae/af), or
3. The mapping table from MAC to binding anchor is secure.

It is SUGGESTED not to initialize a binding based on DHCP Reply, until the associated mechanism is also implemented.

8.4.1.2. Following Actions

If the triggering event is EVE_DHCP_REQUEST/EVE_DHCP_OPTION_RC:

The SAVI device MUST forward the message.

The SAVI device MUST generate an entry for the binding anchor in the Binding State Table (BST) and set the state field to START. The lifetime of this entry MUST set to be MAX_DHCP_RESPONSE_TIME. The Transaction ID (Refer to Section 2 in [RFC2131] and Section 4.2 in [RFC3315]) field of the request packet MUST be recorded in the entry, except that the mapping from link layer address to binding anchor is secure as specified in section 9.2.1.1.

Anchor	Address	State	Lifetime	Other
A		START	MAX_DHCP_RESPONSE_TIME	TID

Figure 4 Binding entry in BST on client triggered initialization

The TID is kept as a mediator of assigned address and the binding anchor of requesting node, to assure that the assigned address can be bound with binding anchor secure.

If the triggering event is EVE_DHCP_CONFIRM:

Other than the actions above, the address to be confirmed MUST be recorded in the entry.

Anchor	Address	State	Lifetime	Other
A	Addr	START	MAX_DHCP_RESPONSE_TIME	TID

Figure 5 Binding entry in BST on Confirm triggered initialization

If the triggering event is EVE_DHCP_REPLY_NULL:

The SAVI device MUST deliver the message to the destination.

The SAVI device MUST generate a new entry in BST and FT. The binding anchor in entry is looked up based on the destination link layer address, from mapping table from link layer address to

binding anchor (e.g., the MAC-Port mapping table in case that port is used as binding anchor). The state of the corresponding entry is set to be BOUND. The lifetime of the entry MUST be set to be the lease time.

Anchor	Address	State	Lifetime	Other
A	Addr	BOUND	Lease time	

Figure 6 Binding entry in BST on Reply triggered initialization

Anchor	Address
A	Addr

Figure 7 Binding entry in FT on Reply triggered initialization

8.4.2. From START to other states

8.4.2.1. Trigger Event

EVE_DHCP_REPLY, EVE_ENTRY_EXPIRE.

8.4.2.2. Following Actions

If the trigger event is EVE_DHCP_REPLY:

The SAVI device MUST deliver the message to the destination.

The state of the corresponding entry is changed to be BOUND.

If the Address field is null, the lease time in Reply message MUST be recorded in the entry.

If the Address field is not null, the Reply is in response to a Confirm message. If the Reply message is of Status Code Success, perform the procedure in section 19 to fetch the lease time. Otherwise, delete the entry.

Anchor	Address	State	Lifetime	Other
A	Addr	BOUND	Lease time	

Figure 8 From START to BOUND

A corresponding entry MUST also be generated in FT.

If the trigger event is EVE_ENTRY_EXPIRE:

The entry MUST be deleted from BST.

8.4.3. From BOUND to other states

8.4.3.1. Trigger Event

EVE_ENTRY_EXPIRE, EVE_DHCP_RELEASE, EVE_DHCP_DECLINE,
EVE_DHCP_REPLY_RENEW.

8.4.3.2. Following Actions

If the trigger event is EVE_ENTRY_EXPIRE:

Remove the corresponding entry in BST and FT.

If the trigger event is EVE_DHCP_RELEASE or EVE_DHCP_DECLINE:

Remove the corresponding entry in BST and FT. The Release or Decline message MUST be forwarded.

If the trigger event is EVE_DHCP_REPLY_RENEW:

Set the lifetime of the address to be the new lease time.

8.5. State Machine of DHCP Snooping

The main state transits are listed as follows.

State	Event	Action	Next State
INIT	REQ/CFM/RC	Generate entry	START
*INIT	RPL	Generate entry with lease	BOUND
START	RPL	Record lease time	BOUND

START	Timeout	Remove entry	INIT
BOUND	RELEASE/DECLINE	Remove entry	INIT
BOUND	Timeout	Remove entry	INIT
BOUND	RPL_RENEW	Set new lifetime	BOUND

*: optional but NOT SUGGESTED.

REQ: EVE_DHCP_REQUEST

CFM: EVE_DHCP_CONFIRM

RC: EVE_DHCP_OPTION_RC

RPL: EVE_DHCP_REPLY

DECLINE: DHCP DECLINE

RELEASE: DHCP RELEASE

RPL_RENEW: EVE_DHCP_RPL_RENEW

Timeout: EVE_ENTRY_EXPIRE

9. Supplemental Binding Process: Handling Link Topology Change

Supplemental binding process is designed to cover conditions that packet is sent by node without previous DHCP procedure sensed by the SAVI device. A typical situation is that the link topology change after the binding has been set up, and then the node will send packet to a different port with the bound port. Another scenario is that a node moves on the local link without re-configuration process, which can be regarded as a special case of link topology change. In DHCP scenario, till this document is finished, link topology change is the only two events that must be handled through this supplemental binding process.

Supplemental binding process is designed to avoid permanent legitimate traffic blocking. It is not supposed to set up a binding whenever a data packet with unbound source address is received. Generally, longer time and more packets are needed to trigger supplemental binding processes.

For implementations that will face the above problem:

1. Binding Recovery Process is a conditional SHOULD. This function SHOULD be implemented if the vendor has such ability, unless the implementation is known to be directly attached to host. If the mechanism is not implemented and managed nodes are not directly attached, permanent blocking will happen until the node is re-configured.
2. Extended Control Packet Snooping Process is a MUST.

Other techniques may be prudently chosen as alternative if found to have equivalent or even better function to avoid permanently blocking after discussion, implementation and deployment.

9.1. Binding Recovery Process

Refer to [draft-baker-savi-one-implementation-approach] for a detailed implementation suggestion. The process specified here can only be enabled in condition that implementation can meet the specified hardware requirements described in [draft-baker-savi-one-implementation-approach].

If a binding anchor is set to have SAVI-BindRecovery attribute, a FIFO queue or register MUST be used to save recently filtered packets. The SAVI device will fetch packet from the queue/register to check the source address can be used by corresponding client on the local link with limited rate:

1. If the address has a local conflict, meaning the DAD on the address fails, the packet MUST be discarded. If the address is not being used, go to the next step.
- 2.

IPv4 address:

Send a DHCPLEASEQUERY [RFC4388] message querying by IP address to all DHCPv4 servers for IPv4 address or a configured server address. The server addresses may be discovered through DHCPv4 Discovery. If no DHCPLEASEACTIVE message is received, discard the packet; otherwise generate a new binding entry for the address.

IPv6 address:

Send a LEASEQUERY [RFC5007] message querying by IP address to All_DHCP_Relay_Agents_and_Servers multicast address or a configured server address. If no successful LEASEQUERY-REPLY is received, discard the packet; otherwise generate a new binding entry for the

address. The SAVI device may repeat this process if a LEASEQUERY-REPLY with OPTION_CLIENT_LINK is received, in order to set up binding entries for all the address of the client.

This process MUST be rate limited to avoid Denial of Services attack against the SAVI device itself. A constant BIND_RECOVERY_INTERVAL is used to control the frequency. Two data based processes on one binding anchor must have a minimum interval time BIND_RECOVERY_INTERVAL. This constant SHOULD be configured prudently to avoid Denial of Services.

This process is not strict secure. The node with SAVI-BindRecovery binding anchor has the ability to use the address of an inactive node, which doesn't reply to the DAD probe.

In case that the SAVI device is a pure layer-2 device, DHCP Confirm MAY be used to replace the DHCP LEASEQUERY. The security degree may degrade for the address may not be assigned by DHCP server.

This process may fail if any DHCP server doesn't support LEASEQUERY.

9.2. Extended Control Packet Snooping Process

In this snooping process, other than DHCP initialization messages, other types of control packets processed by processor of SAVI device, if the source address is not bound, may trigger the device to perform binding process.

The control messages that MUST be processed include: (1) address resolution Neighbor Solicitation; (2) Neighbor Advertisement; (3) neighbor unreachability detection; (4) Multicast Listener Discovery; (5) Address Resolution Protocol; (6) DHCP Renew/Rebind. Other ICMP messages that may be processed by intermediate device may also trigger the binding process.

The SAVI device MUST first perform DAD to check if the address has a local conflict, and then send DHCP LEASEQUERY or Confirm to recover binding based on DHCP server message.

A minimum time interval EXT_SNOOPING_INTERVAL MUST be set to limit the rate of such triggering process.

Note that this process may not be able to avoid permanent block, in case that only data packets are sent by node. Generally, this mechanism is still practical, because data packet sending without control plane communication is rare and suspicious in reality. Normal

traffic will contain control plane communication packets to help traffic setup and fault diagnosis.

10. Filtering Specification

This section specifies how to use bindings to filter packets.

Filtering policies are different for data packet and control packet. DHCP and ND messages that may cause state transit are classified into control packet. Neighbor Advertisement and ARP Response are also included in control packet, because the Target Address of NA and ARP Response should be checked to prevent spoofing. All other packets are considered to be data packets.

10.1. Data Packet Filtering

Data packets with a binding anchor which has attribute SAVI-Validation MUST be checked.

If the source of a packet associated with its binding anchor is in the FT, this packet SHOULD be forwarded; or else the packet SHOULD be discarded, or alternatively the SAVI SHOULD record this violation.

10.2. Control Packet Filtering

For binding anchors with SAVI-Validation attribute:

Discard/record DHCPv4 Discovery with non-all-zeros source IP address. Discard/record DHCPv4 Request whose source IP address is neither all zeros nor a bound address in FT.

Discard/record DHCPv6 Request whose source is not bound with the corresponding binding anchor in FT. Discard/record DHCPv6 Confirm/Solicit whose source is not a link local address bound with the corresponding binding anchor in FT. The link layer address may be bound based on SAVI-SLAAC solution or other solutions.

Discard/record other types of DHCP messages whose source is not an address bound with the corresponding binding anchor.

Discard/record IPv6 NS and IPv4 gratuitous ARP whose source is not an address bound with the corresponding binding anchor.

Discard/record NA and ARP Replies messages whose target address and source address are not bound with the corresponding binding anchor.

For other binding anchors:

Discard DHCP Reply/Ack messages not from binding anchor with the SAVI-DHCP-Trust attribute or SAVI-SAVI attribute.

11. Handle Binding Anchor Off-link Event

Port DOWN event MUST be handled if switch port is used as binding anchor. In more general case, if a binding anchor turns off-link, this event MUST be handled.

Whenever a binding anchor with attribute SAVI-Validation turns down, the bindings with the binding anchor MUST be kept for a short time.

To handle movement, if receiving DAD NS/Gra ARP request targeting at the address during the period, the entry MAY be removed.

If the binding anchor turns on-link during the period, recover bindings. It may result in some security problem, e.g., a malicious node immediately associates with the binding anchor got off by a previous node, and then it can use the address assigned to the previous node. However, this situation is very rare in reality. Authors decide not to handle this situation.

12. Binding Number Limitation

It is suggested to configure some mechanism in order to prevent a single node from exhausting the binding table entries on the SAVI device. Either of the following mechanism is sufficient to prevent such attack.

1. Set the upper bound of binding number for each binding anchor with SAVI-Validation.
2. Reserve a number of binding entries for each binding anchor with SAVI-Validation attribute and all binding anchors share a pool of the other binding entries.
3. Limit DHCP Request rate per binding anchor, using the bound entry number of each binding anchor as reverse indicator.

13. State Restoration

If a SAVI device reboots accidentally or designedly, the states kept in volatile memory will get lost. This may cause hosts indirectly attached to the SAVI device to be broken away from the network, because they can't recover bindings on the SAVI device of themselves. Thus, binding entries MUST be saved into non-volatile storage whenever a new binding entry changes to BOUND state or a binding with

state BOUND is removed in condition that this function is supported by hardware. Immediately after reboot, the SAVI device MUST restore binding states from the non-volatile storage. The lifetime and the system time of save process MUST be stored. Then the device MUST check whether the saved entries are obsolete when rebooting.

The possible alternatives proposed but not suitable for general cases are:

If the SAVI device is also the DHCP relay, an alternative mechanism is fetching the bindings through bulk DHCP LEASEQUERY [RFC5460].

If the network enables 802.1ag, the bindings can be recovered with the help of the first hop routers through snooping unicast Neighbor Solicitations sent by routers based on the Neighbor Table.

14. Confirm Triggered Binding

If a binding entry is triggered by a CONFIRM message from the client, no lease time will be contained in the REPLY from DHCP server. The SAVI device MUST send LEASEQUERY message to get the lease time of the address to complete the binding entry. If no successful LEASEQUERY-REPLY is received, the binding entry SHOULD be removed. In this scenario, the address is not regarded as assigned by DHCP, and it MAY be bound through other SAVI solution.

If the confirmed address has local conflict, the Client-ID field of Confirm and LEASEQUERY-REPLY MUST be compared. If they are not match, the new binding entry MUST be deleted.

15. Consideration on Link Layer Routing Complexity

An implicit assumption of this solution is that data packet must arrive at the same binding anchor with the binding anchor that the control packets have arrived at. If this assumption is not valid, this control packet based solution will fail or at least discard legitimate packet. Unfortunately, if the link layer routing between host and SAVI device is inconsistent from time to time, this assumption doesn't stand. Time consistency of link layer routing is not assured by link layer routing protocol. For example, TRILL, a recent link layer routing protocol, is flexible and multiple link layer paths are allowed.

To make the basic assumption stand, the best way is enforcing that there should be only one topology path from downstream host to the SAVI device. For example, SAVI device is directly attached by hosts.

If the assumption doesn't stand, a better solution is requiring inter-operation between SAVI protocol and the link layer routing protocol to make SAVI protocol sensitive to the link layer routing change. This solution is above the scope of this document.

16. Duplicate Bindings of Same Address

Note that the same address may be bound with multiple binding anchors, only if the binding processes are finished on each binding anchor successfully respectively.

This mechanism is designed in consideration that a node may move on the local link, and a node may have multiple binding anchors.

Note that the local link movement scenario is not handled perfectly. The former binding may not be removed, unless the node is directly attached to the SAVI device. The nodes sharing the same former binding anchor of the moving node have the ability to use its address.

17. Constants

MAX_DHCP_RESPONSE_TIME 120s

BIND_RECOVERY_INTERVAL Device capacity depended and configurable

18. Security Considerations

There is no security consideration currently.

19. IANA Considerations

There is no IANA consideration currently.

20. References

20.1. Normative References

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

20.2. Informative References

[RFC2131] R. Droms, "Dynamic Host Configuration Protocol", RFC2131, March 1997.

[RFC3307] B. Haberman, "Allocation Guidelines for IPv6 Multicast Addresses", RFC3307, August 2002.

[RFC3315] R. Droms, Ed. "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC3315, July 2003.

[RFC4388] R. Woundy and K. Kinneer, "Dynamic Host Configuration Protocol (DHCP) Leasequery", RFC4388, February 2006.

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

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

[RFC5007] J. Brzozowski, K. Kinneer, B. Volz, S. Zeng, "DHCPv6 Leasequery", RFC5007, September 2007.

[RFC5227] S. Cheshire, "IPv4 Address Conflict Detection", RFC5227, July 2008.

[IP Source Guard] Cisco, "Network Security Technologies and Solutions", chapter 7, Cisco Press, May 20, 2008.

[draft-baker-savi-one-implementation-approach] F. Baker, "An implementation approach to Source Address Validation", draft-baker-savi-one-implementation-approach-00.

[draft-bi-savi-mixed] Jun Bi, "Mixed scenario analysis and best effort solution", draft-bi-savi-mixed-00.

21. Acknowledgments

Special thanks to Christian Vogt and Joel M. Halpern for careful review and valuation comments on the state machine and text. Thanks to Marcelo Bagnulo Braun, Eric Levy-Abegnoli, Mark Williams, Erik Nordmark, Mikael Abrahamsson, Alberto Garcia, Jari Arkko, David Harrington, Pekka Savola, Xing Li, Lixia Zhang, Robert Raszuk, Greg Daley, John Kaippallimalil and Tao Lin for their valuable contributions.

Authors' Addresses

Jun Bi
CERNET
Beijing, China
Email: junbi@cernet.edu.cn

Jianping Wu
CERNET
Beijing, China
Email: jianping@cernet.edu.cn

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

Fred Baker
Cisco Systems
Email: fred@cisco.com

22. Change Log

From 02 to 03: Section 12, data trigger and counter trigger are combined to binding recovery process. The expression "one of MUST" is changed to "conditional MUST. Conditions related with the implementation are specified. Related constants are changed in section 26."

Main changes from 03 to 04:

- Section "Prefix configuration" is removed.
- Section "Supplemental binding process" is modified in requirement level.
- Sub-section 9.1 "Rationale" is added.
- Section "Filtering during Detection" is removed.

- Section "Handling layer 2 path change" is changed to "Consideration on Link layer routing complexity"
- Section "Background and related protocols" is removed.

Main changes from 04 to 05:

- Trigger events are listed explicitly in section 8.
- Dection and Live states are deleted, together with corresponding sections.

Main change from 05 to 06:

- Section 8.1: reference section 20 is changed to section 15.

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 28, 2011

E. Nordmark
Sun
M. Bagnulo
UC3M
E. Levy-Abegnoli
Cisco Systems
October 25, 2010

FCFS-SAVI: First-Come First-Serve Source-Address Validation for Locally
Assigned Addresses
draft-ietf-savi-fcfs-05

Abstract

This memo describes FCFS SAVI a mechanism to provide source address validation for IPv6 networks using the First-Come First-Serve approach. The proposed mechanism is intended to complement ingress filtering techniques to provide a higher granularity on the control of the source addresses used.

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 April 28, 2011.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	4
2. Non-normative background to FCFS SAVI	4
2.1. Scope of FCFS SAVI	4
2.2. Constraints for FCFS SAVI design	5
2.3. Address ownership proof	5
2.4. Layer-2 Anchor considerations	6
2.5. SAVI enforcement perimeter	6
3. FCFS SAVI normative specification	10
3.1. FCFS SAVI Data structures	10
3.2. FCFS SAVI algorithm	11
3.2.1. Discovering on-link prefixes	11
3.2.2. Processing of transit traffic	11
3.2.3. Processing of local traffic.	12
3.2.4. SAVI port configuration guidelines	17
3.2.5. VLAN support	18
3.3. Protocol Constants	18
4. Security Considerations	18
5. Contributors	21
6. Acknowledgments	21
7. References	22
7.1. Normative References	22
7.2. Informative References	22
Appendix A. Implications of not following the recommended behaviour	22
A.1. Lack of binding state due to packet loss	22
A.1.1. Why initial packets may be (frequently) lost	23
A.2. Lack of binding state due to a change in the topology	25
A.3. Lack of binding state due to state loss	26
A.3.1. The case of a host directly connected to the SAVI device	26
A.3.2. The case of a host connected to the SAVI device through one or more legacy devices.	27
Authors' Addresses	28

1. Introduction

This memo describes FCFS SAVI, a mechanism to provide source address validation for IPv6 networks using the First-Come First-Serve approach. The proposed mechanism is intended to complement ingress filtering techniques to provide a higher granularity on the control of the source addresses used.

2. Non-normative background to FCFS SAVI

2.1. Scope of FCFS SAVI

The application scenario for FCFS SAVI is limited to the local link. This means that 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 any link there usually are hosts and routers attached. Hosts generate packets with their own address as the source address. This is the so-called local traffic. While routers send packets containing a source address other than their own, since they are forwarding packets generated by other hosts (usually located in a different link). This what the so-called transit traffic.

The applicability of FCFS SAVI is limited to the local traffic i.e. to verify if the traffic generated by the hosts attached to the local link contains a valid source address. The verification of the source address of the transit traffic is out of the scope of FCFS SAVI. Other techniques, like ingress filtering [RFC2827], are recommended to validate transit traffic. In that sense, FCFS SAVI complements ingress filtering, since it relies on ingress filtering to validate transit traffic but is 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 addresses, in particular with address configured through Stateless Address AutoConfiguration (SLAAC) [RFC4862]. Manually configured addresses can be supported by FCFS SAVI, but manual configuration of the binding on the SAVI device provides higher security and seems compatible with manual address management. 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 must solely rely in the usage of already available protocols. This means that FCFS SAVI cannot 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 any source address validation solution, we need to define some address ownership proof concept i.e. what it means to be able to proof that a given host owns a given address in the sense that the host is entitled to send packets with that source address.

Since no host changes are acceptable, we need to find the means to proof address ownership without requiring a new protocol. In FCFS SAVI the address ownership proof is based in the First-Come First-Serve approach. This means that the first host that claims a given source address is the owner of the address until further notice. More precisely, 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 the layer-2 information that the FCFS SAVI box has available (e.g. the port in a switched LAN). Following data packets containing that IP source address must use the same layer-2 information in order to be compliant.

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 layer-2 information has 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 layer-2 information. In order to do that FCFS SAVI uses the Neighbour Discovery (ND) protocol. If the host is no longer reachable at the previously recorded layer-2 information, FCFS SAVI assumes that the new location is valid and creates a new binding using the new layer-2 information. In case the host is still reachable using the previously recorded information, the packets coming from the new layer-2 information 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. 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. Layer-2 Anchor considerations

Any SAVI solution is not stronger than the Layer-2 anchor it uses. If the Layer-2 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 Layer-2 anchor is easily spoofable and the SAVI device is configured to drop non compliant packets, then the usage of SAVI may open a new vector of Denial of Service attacks, based on spoofed Layer-2 anchors. For that reason, in this document, we assume that the Layer-2 anchors used by the SAVI solution are not easily spoofable (e.g. ports of a switched network) and that the SAVI device can be configured to drop non-compliant packets. For the rest of the document, we will assume that the Layer-2 anchors are ports of a switched network.

2.5. SAVI enforcement perimeter

SAVI provides perimetrical security. This means that the SAVI devices form what can be called a SAVI enforcement 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 SAVI devices are deployed in the topology and on the configuration of the 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. This means that while packets coming from interfaces connected to the external part of the topology need to be validated by the 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 SAVI device. It also implies that each 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 SAVI device binds the source address to a layer-2 anchor.

One possible approach would be for every SAVI device to store binding information about every source addresses in the subnetwork. This means that every SAVI device would store binding for each source address to the local layer-2 anchor through packets with that source address can be received through. The problem with this approach is that it imposes significant memory burden on the SAVI devices. In order to reduce the memory requirements imposed to each device, the SAVI solution described in this specification distributes the storage of SAVI binding information among the multiple SAVI devices of a subnetwork. The SAVI binding state is distributed across the SAVI devices according to the following criteria: each SAVI device will store binding information about the source addresses bound to layer-2 anchors corresponding to the interfaces that connect to the part of the topology that is outside of the SAVI enforcement perimeter. Since all the untrusted packet sources are by definition in the external part of the perimeter, this means that the packets generated by each of the untrusted sources will reach the perimeter through one interface of a SAVI device. The binding information for that particular source address will be stored in this first SAVI device the packet reaches to.

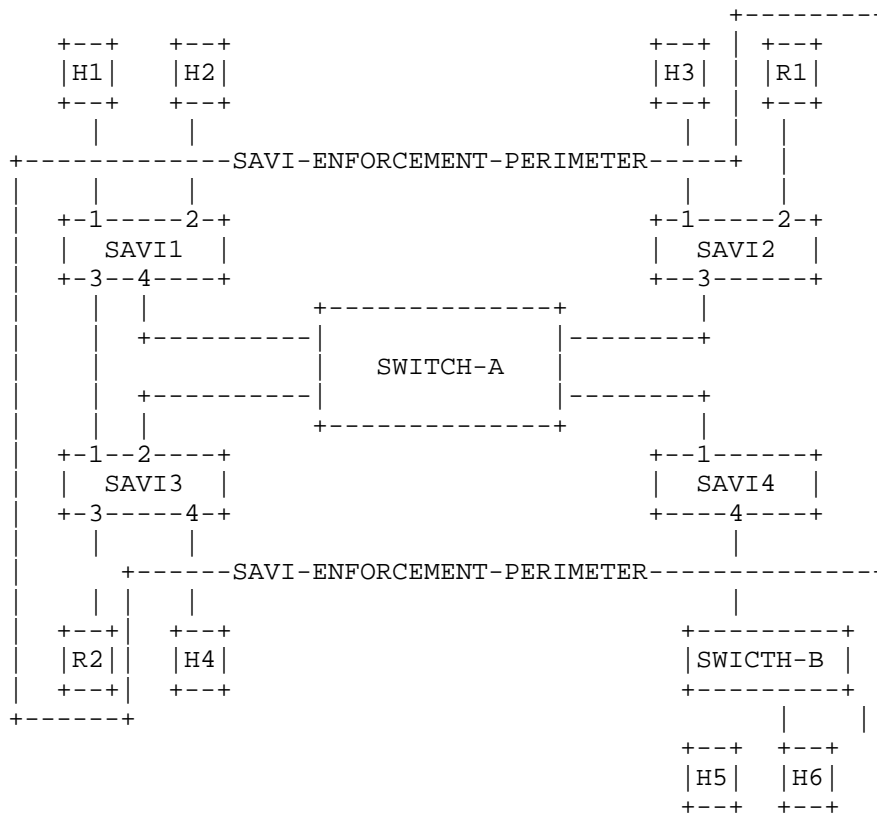
This means the SAVI binding information will be distributed across multiple devices. In order to provide proper source address validation, it is critical that the information distributed among the different SAVI devices is coherent. In particular, it is important to avoid that the same source address is bound to different layer-2 anchors in different 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 we are trying to prevent. In order to preserve the coherency of the SAVI bindings distributed among the SAVI devices within a realm, the Neighbour Discovery (ND) protocol is used, in particular the Neighbour Solicitation (NSOL) and Neighbour Advertisement (NADV) messages. Before creating a SAVI binding in the local SAVI database, the SAVI device will send a NSOL message querying for the address involved. Should any host reply to that message with a NADV message, the SAVI device that sent the NADV will infer that a binding for that address exists in another SAVI device and will not create a local binding for it. If no NADV message is received as a reply to the NSOL, then the local SAVI device will infer that no binding for that address exists in other SAVI device and will create the local SAVI binding for that address. (NOTE that the description included here is overly simplified to illustrate the mechanism used to preserve the coherency of the binding databases of the different SAVI devices. The actual SAVI mechanism normative

specification as described in Section 3 varies in the fact that in some cases it is the SAVI device that generates the NSOL while in other cases it simply forwards the NSOL generated by the end host, and that the SAVI device will send multiple copies of the NSOL in order to improve the reliability of the message exchange, among other things).

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

- o SAVI provides perimetrical security, so some interfaces of a SAVI device will connect to the internal (trusted) part of the topology and other interfaces will connect to the external (untrusted) part of the topology.
- o A SAVI device only verifies packets coming through one interface connected to the untrusted part of the topology.
- o A SAVI device only stores binding information for the source addresses that are bound to layer-2 anchors that correspond to interfaces that connect to the untrusted part of the topology.
- o SAVI uses the NSOL and NADV messages to preserve the coherency of the SAVI binding state distributed among the SAVI devices within a realm.

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



In the figure above, the SAVI enforcement perimeter is provided by 4 SAVI devices, namely SAVI1, SAVI2, SAVI3 and SAVI4. These devices verify information related to the source address both in data and in ND packets and filter packets accordingly.

SAVI devices then have two types of ports: trusted ports and validating ports.

- o Validating ports (VPs) are those in which SAVI processing is performed. This means that when a packet is received through one of the validating ports, the SAVI processing and filtering will be executed.
- o Trusted ports (TPs) are those in which SAVI processing is not performed. So, packets received through trusted ports are not validated and no SAVI processing is performed in them.

Trusted ports are used for connections with trusted infrastructure, including the communication between SAVI devices, the communication with routers and the communication of other switches that while they

are not SAVI devices, they only connect to trusted infrastructure (i.e. other 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 SAVI devices. Port 4 of SAVI1, port 3 of SAVI2, port 2 of SAVI3 and port 1 of SAVI4 are trusted because they connect to SWITCH-A to which only trusted nodes are connected. In 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 SAVI enforcement perimeter also are connected through validating port. 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.

3. FCFS SAVI normative specification

3.1. FCFS SAVI Data structures

FCFS SAVI function relies on state information binding the source address used in data packets to the layer-2 information 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:

- o IP source address
- o Layer-2 information, such as Layer-2 address, port through which the packet was received, etc
- o Lifetime
- o Status: either tentative, valid or testing
- o Creation time: the value of the local clock when the entry was firstly created

In addition to this, FCFS SAVI need to know what are the prefixes that are directly connected, so it maintains a data structure called the FCFS SAVI prefix list, which contains:

- o Prefix
- o 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 SAVI device relies on the FCFS SAVI Prefix list, which contains the set of on-link prefixes. A SAVI device MUST support the following two methods for populating the Prefix List: Manual configuration and Router Advertisement, as detailed next.

Manual configuration: A SAVI device MUST support manual configuration of the on-link prefixes included in the Prefix List.

Router Advertisement: A SAVI device MUST support discovery of on-link prefixes through Router Advertisement messages. The 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 instead than in the ND Prefix List. In addition, when the 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 bridge. 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 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 local prefixes of the receiving interface, this means that the data packet is transit traffic and the packet SHOULD be discarded. The FCFS SAVI function MAY send an ICMP Destination Unreachable Error back to the source address of the data packet. (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 the receiving port, then the SAVI local traffic validation processes is executed as described below.

3.2.3. Processing of local traffic.

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

The state machine described is for the binding of a given source IP address in a given SAVI device. So this means that all the packets described as inputs in the state machine above refer to that given IP address. The key attribute is the IP address. The full state information is:

- o IP ADDRESS: IPAddr
- o LAYER_2 ANCHOR: P
- o LIFETIME: LT

The possible states are:

- o NO_BIND
- o TENTATIVE
- o VALID
- o TESTING_TP
- o TESTING_VP
- o TESTING_LIFETIME

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 Layer-2 anchor.

NO_BIND: The binding for a source IP address entry is in this state when it does not have any binding to a Layer 2 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 DAD_NSOL has been received is in this state during the waiting period during which the DAD procedure is being executed (either by the host itself the SAVI device on its behalf).

VALID: The binding for the source address has been verified, 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 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 entry is about to expire. This is due to the fact that no packets has been seen by the 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.

A simplified figure of the state machine is included below.

NO_BIND

- o Upon the reception through a Validating Port (VP) of a Neighbour Solicitation (NSOL) generated by the Duplicate Address Detection (DAD) procedure (hereafter named DAD_NSOL) containing Target Address IPAddr, the SAVI device MUST 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 250 milliseconds (i.e. the time between two Neighbour Solicitation messages is 250 millisecs). This is equivalent to sending the received DAD_NSOL message twice. The DAD_NSOL messages are not sent through any of the ports configured as Validating Ports. The NSOL messages are sent through the proper Trusted Ports (as defined by the switch behaviour that will depend on whether it performs MLD snooping or not). The state is moved to TENTATIVE. The LIFETIME is set to TENT_LT (i.e. LT==TENT_LT), the LAYER_2 ANCHOR is set to VP (i.e. P==VP) and the Creation time is set to the current value of the local clock.
- o Upon the reception through a Validating Port (VP) of a DATA packet containing IPAddr as the source address, the SAVI device SHOULD execute the process of sending 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 250 milliseconds (i.e. the time between two Neighbour Solicitation messages is 250 millisecs). The implications of not following the recommended behaviour are described in Appendix A. The DAD_NSOL messages are not sent through any of the ports configured as Validating Ports. The NSOL messages are sent through the proper Trusted Ports (as defined by the switch behaviour that will depend on whether it performs MLD snooping or not). The SAVI device MAY discard the data packet while the DAD procedure is being executed. The state is moved to TENTATIVE. The LIFETIME is set to TENT_LT (i.e. LT==TENT_LT), the LAYER_2 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_NSOL packets containing IPAddr as the target address received through a Trusted port are NOT forwarded through any of the Validating ports but they are sent through the proper Trusted Ports (as defined by the switch behaviour that will depend on whether it performs MLD snooping or not)
- o Neighbor Advertisement packets sent to all nodes as a reply to the DAD_NSOL (hereafter called DAD_NADV) 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 (NADV) 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 LAYER_2 ANCHOR and the LIFETIME are cleared. Data packets stored corresponding to this binding are discarded.
- o If a NADV is received through a Validating Port with Target Address set to IPAddr, the NADV packet is discarded
- o If a data packet with source address IPAddr is received with Layer_2 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 (waiting for the NADV).

- 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 NSOL 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 LAYER_2 ANCHOR and LIFETIME are cleared. Data packets stored corresponding to this binding are discarded.
- o If a DAD NSOL 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_DAD_NSOL, but the LAYER_2_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_NSOL is received from a Trusted port, then the DAD_NSOL message is forwarded to port P and it is also forwarded to the proper Trusted Ports (as defined by the switch behaviour that will depend on whether it performs MLD snooping or not). 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_NSOL or DAD_NADV 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_NSOL 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 NSOL messages for that address will be sent by the SAVI device) and RetransTimer set to 250 milliseconds (i.e. the time between two NSOL messages is 250 millisecs). The DAD_NSOL 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.
- o If the LIFETIME expires, then the SAVI device will execute the process of sending DAD_NSOL 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 NSOL messages for that address will be sent by the SAVI device) and RetransTimer set to 250 milliseconds (i.e. the time between two NSOL messages is 250 millisecs). The DAD_NSOL 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_NADV coming from port P and containing IPAddr as the target address are forwarded as usual.

TESTING_TP-LT

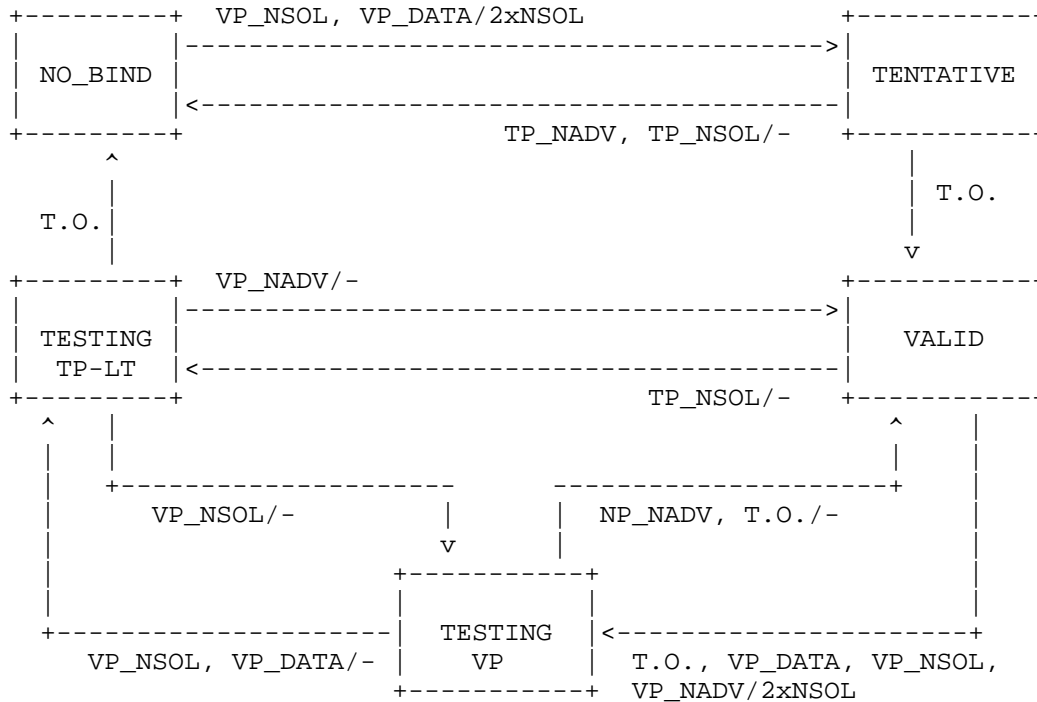
- o If the LIFETIME expires, the LAYER_2 ANCHOR is cleared and the state is changed to NO_BIND
- o If a NADV message containing the IPAddr as target address is received through the Validating port P as a reply to the DAD_NSOL message, then the NADV is forwarded as usual and the state is changed to VALID. 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_NSOL is received from a Trusted port, the DAD NSOL is forwarded as usual.
- o If a DAD_NSOL is received from a Validating Port P' other than P, the DAD_NSOL 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 LAYER_2 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 NADV message containing the IPAddr as target address is received through the Validating port P as a reply to the DAD_NSOL message, then the NADV 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 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 is received through a Trusted Port port that is other than port P, the state is moved to TESTING_TP-LT, and the packet MAY be discarded.
- o If a DAD_NSOL is received through Trusted Port, the packet is forwarded as usual and the state is moved to TESTING_TP-LT.

- o If a DAD_NSOL is received through Validating Port P'' other than P or P', the packet is forwarded as usual and P'==P''

Simplified state machine figure



MLD considerations

The 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 SAVI device will receive the DAD_NSOL for those addresses. Please note that it may not be enough to relay 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 SAVI device. So, the SAVI device SHOULD join the solicited node multicast groups for all the addresses that are in a state other than NO_BIND

3.2.4. SAVI port configuration guidelines

The guidelines for port configuration in SAVI devices are:

- o Ports that are connected to another SAVI device SHOULD be configured as Trusted ports. Not doing so will at least significantly increase the memory consumption in the SAVI devices.

- o Ports connected to hosts SHOULD be configured as Validating ports. Not doing so will allow the host connected to that port to send packets with spoofed source address.
- o Ports connected to routers SHOULD be configured as Trusted ports. Configuring them as Validating ports would increase the signaling due to SAVI. The implication is that a router can generate traffic with any source address as they are assumed to be part of the trusted infrastructure.
- o Ports connected to a chain of one or more legacy switches that have hosts connected SHOULD be configured as Validating ports. Not doing so will allow the host connected to any of these switches to send packets with spoofed source address.
- o Ports connected to a chain of one or more legacy switches that have other SAVI devices and/or routers connected but had no hosts attached to them SHOULD be configured as Trusted ports. Not doing so will at least significantly increase the memory consumption in the SAVI devices and increase the signaling traffic due to SAVI validation.
- o Ports connected to a chain of one or more legacy switches that have a mix of SAVI devices and/or routers with 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. Nevertheless, this topology will result in increased SAVI signaling and memory consumption compared to a topology where SAVI-hosts communications and inter SAVI communications are kept through different legacy switches.

3.2.5. VLAN support

In the case the SAVI device is a switch that supports VLANs, the SAVI implementation will behave as if there was one SAVI process per VLAN. The SAVI process of each VLAN will store the binding information corresponding the the nodes attached to that particular VLAN.

3.3. Protocol Constants

TENT_LT is 500 milliseconds

DEFAULT_LT is 5 minutes

4. Security Considerations

First of all, it should be noted that any SAVI solution will be as strong as the lower layer anchor that it uses. In particular, if the lower layer anchor is forgeable, then the resulting SAVI solution will be weak. For example, if the lower layer anchor is a MAC address that can be easily spoofed, then the resulting SAVI will not

be stronger than that. On the other hand, if we use switch ports as lower layer anchors (and there is only one host connected to each port) it is likely that the resulting SAVI solution will be considerably more secure.

Denial of service attacks

There are two types of DoS attacks that can be envisaged in a SAVI environment. On one hand, we can envision attacks against the SAVI device resources. On the other hand, we can envision DoS attacks against the hosts connected to the network where SAVI is running.

The attacks against the SAVI device basically consist on making the SAVI device to consume its resource until it runs out of them. For instance, a possible attack would be to send packets with different source addresses, making the SAVI device to create state for each of the addresses and waste memory. At some point the 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 SAVI device runs out of the memory allocated for the 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 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 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 SAVI device may attempt to launch a DoS attack towards the 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 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 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 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 an attacks as the above described.

The other type of attack is when an attacker manages to create state in the SAVI device that will result in blocking the data packets sent by the legitimate owner of the address. In IPv6 these attacks are not an issue thanks to the 2^{64} addresses available in each link.

The SAVI device generates 2 DAD_NSOL packets upon the reception of a DAD_NSOL or a data packet. As such, the SAVI device can be used as an amplifier by attackers. In order to limit this type of attack, it is RECOMMENDED that the SAVI device 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 SAVI device to function normally in the rest of the ports.

Residual threats.

SAVI perform its function by binding an IP source address to a Layer_2 anchor. If the attacker manages to send packets using the Layer_2 anchor associated to a given IP address, SAVI validation will be successful and the SAVI device will allow the packet through. This can be achieved by spoofing the Layer_2 anchor or because the Layer_2 anchor is shared among the legitimate owner of the address and the attacker. An example of the latter is the case where the Layer_2 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 tot he legacy switch will share the same Layer_2 anchor (i.e. the switch port). This means that hosts connected to the legacy switch can spoof each others IP address and this will not be detected by the SAVI device. This can be prevented by not sharing Layer_2 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). This means that when a DAD_NSOL is issued for a given IP address for which a binding exists in a SAVI device, the SAVI device expects to see a DAD_NADV coming from the Layer_2 anchor associated to that IP address

in order to preserve the binding. If the SAVI device does not see the DAD_NADV, it may grant the binding to a different Layer_2 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 Layer_2 anchor. An attacker may do so for example by intercepting the DAD_NADV or launching a DoS attack to the host that will prevent it to issue proper DAD replies.

5. Contributors

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

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

Fred Baker
Cisco Systems
Email: fred@cisco.com

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

6. Acknowledgments

This draft benefited from the input from: Joel Halpern, Christian Vogt, Dong Zhang, Frank Xia, Jean-Michel Combes and Lin Tao.

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

7. References

7.1. Normative References

- [RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, May 2000.
- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.

7.2. Informative References

- [I-D.ietf-savi-framework]
Wu, J., Bi, J., Bagnulo, M., Baker, F., and C. Vogt,
"Source Address Validation Improvement Protocol
Framework", draft-ietf-savi-framework-00 (work in
progress), September 2010.
- [RFC4429] Moore, N., "Optimistic Duplicate Address Detection (DAD)
for IPv6", RFC 4429, April 2006.

Appendix A. Implications of not following the recommended behaviour

This specification recommends SAVI implementations to generate a DAD_NSOL 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. Lack of binding state due to packet loss

The DAD procedure is inherently unreliable. It consists on sending a NSOL packet and if no NADV 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 NSOL packet or the NADV 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 coalition 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 NSOL. However, current RFC4862 defines a default value of 1 NSOL 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 packeted 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). This means that 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 fixed access networks

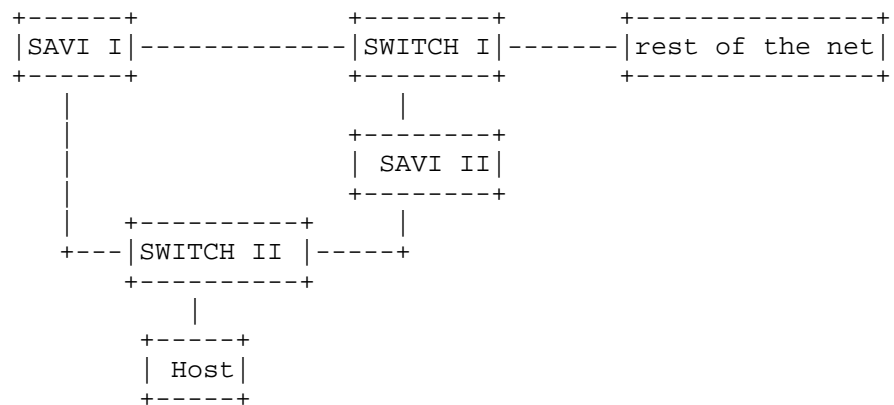
In fixed access networks such as DSL and Cable the end hosts are usually connected to the access network through a residential gateway (RG). If the host interface is initialized prior to the residential gateway getting authenticated and connected to the access network, the access network is not aware 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. 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 NSOL packet, which implies that the data packets coming from the host that sent the DAD NSOL 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.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 NSOL. 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.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:

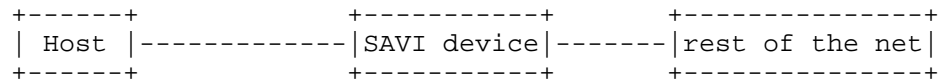
An end-to-end protocol design should not rely on the maintenance of state (i.e. information about the state of the end-to-end communication) inside the network. Such state should be maintained only in the endpoints, in such a way that the state can only be destroyed when the endpoint itself breaks (known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

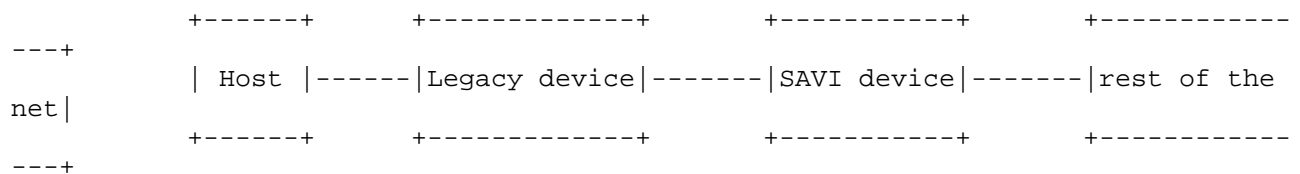
Duplicate Address Detection:

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 go 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.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 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.3.2.1. Enforcing direct connectivity between the SAVI device and the host

Some people have 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.

Authors' Addresses

Erik Nordmark
Sun Microsystems, Inc.
17 Network Circle
Menlo Park, CA 94025
USA

Phone: +1 650 786 2921
Email: Erik.Nordmark@Sun.COM

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

Network Working Group
Internet-Draft
Intended status: Informational
Expires: April 28, 2011

Jianping Wu
Jun Bi
CERNET
Marcelo Bagnulo
UC3M
Fred Baker
Cisco
Christian Vogt, Ed.
Ericsson
October 25, 2010

Source Address Validation Improvement Framework
draft-ietf-savi-framework-01

Abstract

The Source Address Validation Improvement method was developed to complement ingress filtering with finer-grained, standardized IP source address validation. This document describes and motivates the design of the SAVI method.

Status of this Memo

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at <http://www.ietf.org/ietf/lid-abstracts.txt>.

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>.

This Internet-Draft will expire on April 28, 2011.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the

document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	3
2. Model	3
3. Deployment Options	5
4. Scalability Optimizations	6
5. Reliability Optimizations	8
6. Acknowledgment	9
7. References	9
Authors' Addresses	9

1. Introduction

Since IP source addresses are used by hosts and network entities to determine the origin of a packet and as a destination for return data, spoofing of IP source addresses can enable impersonation, concealment, and malicious traffic redirection. Unfortunately, the Internet architecture does not prevent IP source address spoofing. Since the IP source address of a packet generally takes no role in forwarding the packet, it can be selected arbitrarily by the sending host without jeopardizing packet delivery. Extra methods are necessary for IP source address validation, to augment packet forwarding with an explicit check of whether a given packet's IP source address is legitimate.

IP source address validation can happen at different granularity: Ingress filtering [BCP38], a widely deployed standard for IP source address validation, functions at the coarse granularity of networks. It verifies that the prefix of an IP source address routes to the network from which the packet was received. An advantage of ingress filtering is simplicity: The decision of whether to accept or to reject an IP source address can be made solely based on the information available from routing protocols. However, the simplicity comes at the cost of not being able to validate IP source addresses at a finer granularity, due to the aggregated nature of the information available from routing protocols. Finer-grained IP source address validation would be helpful to enable IP-source-address-based authentication, authorization, and host localization, as well as to efficiently identify misbehaving hosts. Partial solutions [BA2007] exist for finer-grained IP source address validation, but are proprietary and hence often unsuitable for corporate procurement.

The Source Address Validation Improvement method was developed to complement ingress filtering with standardized IP source address validation at the maximally fine granularity of individual IP addresses: It prevents hosts attached to the same link from spoofing each other's IP addresses. To facilitate deployment in networks of various kinds, the SAVI method was designed to be modular and extensible. This document describes and motivates the design of the SAVI method.

2. Model

To enable network operators to deploy fine-grained IP source address validation without a dependency on supportive functionality on hosts, the SAVI method was designed to be purely network-based. A SAVI instance is located on the path of hosts' packets, enforcing the

hosts' use of legitimate IP source addresses according to the following three-step model:

1. Identify which IP source addresses are legitimate for a host, based on monitoring packets exchanged by the host.
2. Bind a legitimate IP address to a link layer property of the host's network attachment. This property, called a "binding anchor", must be verifiable in every packet that the host sends, and harder to spoof than the host's IP source address itself.
3. Enforce that the IP source addresses in packets match the binding anchors to which they were bound.

This model allows a SAVI instance to be located anywhere on the link to which the hosts attach, hence enabling different locations for a SAVI instance. One way to locate a SAVI instance is in the hosts' default router. IP source addresses are then validated in packets traversing the default router, yet the IP source addresses in packets exchanged locally on the link may bypass validation. Another way to locate a SAVI instance is in a switch between the hosts and their default router. Thus, packets may undergo IP source address validation even if exchanged locally on the link.

The closer a SAVI instance is located to the hosts, the more effective the SAVI method is. This is because each of the three steps of the SAVI model can best be accomplished in a position close to the host:

- o Identifying a host's legitimate IP source addresses is most efficient close to the host, because the likelihood that the host's packets bypass a SAVI instance, and hence cannot be monitored, increases with the distance between the SAVI instance and the host.
- o Selecting a binding anchor for a host's IP source address is easiest close to the host, because many link layer properties are unique for a given host only on a link segment directly attaching to the host.
- o Enforcing a host's use of a legitimate IP source address is most reliable when pursued close to the host, because the likelihood that the host's packets bypass a SAVI instance, and hence do not undergo IP source address validation, increases with the distance between the SAVI instance and the host.

The preferred location of SAVI instances is therefore close to hosts, such as in switches that directly attach to the hosts whose IP source

addresses are being validated.

3. Deployment Options

The model of the SAVI method, as explained in Section 2, is deployment-specific in two ways:

- o The identification of legitimate IP source addresses is dependent on the IP address assignment method in use on a link, since it is through assignment that a host becomes the legitimate user of an IP source address.
- o Binding anchors are dependent on the technology used to build the link on which they are used, as binding anchors are link layer properties of a host's network attachment.

To facilitate the deployment of the SAVI method in networks of various kinds, the SAVI method is designed to support different IP address assignment methods, and to function with different binding anchors. Naturally, both the IP address assignment methods in use on a link and the available binding anchors have an impact on the functioning and the strength of IP source address validation. The following two sub-sections explain this impact, and describe how the SAVI method accommodates this.

3.1. IP Address Assignment Methods

Since the SAVI method traces IP address assignment packets, it necessarily needs to incorporate logic that is specific to particular IP address assignment methods. However, developing SAVI method variants for each IP address assignment method is alone not sufficient, since multiple IP address assignment methods may co-exist on a given link. The SAVI method hence comes in multiple variants: for links with Stateless Address Autoconfiguration, for links with DHCP, for links with Secure Neighbor Discovery, and for links that use any combination of IP address assignment methods.

The reason to develop SAVI method variants for each single IP address configuration method, in addition to the variant that handles all IP address assignment methods, is to minimize the complexity of the common case: Many link deployments today either are constrained to a single IP address assignment methods or, equivalently from the perspective of the SAVI method, separate IP address assignment methods into different IP address prefixes. The SAVI method for such links can be simpler than the SAVI method for links with multiple IP address assignment methods per IP address prefix.

3.2. Binding Anchors

The SAVI method supports a range of binding anchors:

- o The IEEE extended unique identifier, EUI-48 or EUI-64, of a host's interface.
- o The port on an Ethernet switch to which a host attaches.
- o The security association between a host and the base station on wireless links.
- o The combination of a host interface's link-layer address and a customer relationship in cable modem networks.
- o An ATM virtual channel, a PPPoE session identifier, or an L2TP session identifier in a DSL network.
- o A tunnel that connects to a single host, such as an IP-in-IP tunnel, a GRE tunnel, or an MPLS label-switched path.

The various binding anchors differ significantly in the security they provide. IEEE extended unique identifiers, for example, fail to render a secure binding anchor because they can be spoofed with little effort. And switch ports alone may be insufficient because they may connect to more than a single host, such as in the case of concatenated switches.

Given this diversity in the security provided, one could define a set of possible binding anchors, and leave it up to the administrator to choose one or more of them. Such a selection of binding anchors would, of course, have to be accompanied by an explanation of the pros and cons of the different binding anchors. In addition, SAVI devices may have a default binding anchor depending on the lower layers. Such a default could be to use switch ports when available, and MAC addresses otherwise. Or to use MAC addresses, and switch ports in addition if available.

4. Scalability Optimizations

The preference to locate a SAVI instance close to hosts implies that multiple SAVI instances must be able to co-exist in order to support large links. Although the model of the SAVI method is independent of the number of SAVI instances per link, co-existence of multiple SAVI instances without further measures can lead to higher-than-necessary memory requirements: Since a SAVI instance creates bindings for the IP source addresses of all hosts on a link, bindings are replicated

if multiple SAVI instances co-exist on the link. High memory requirements, in turn, increase the cost of a SAVI instance. This is problematic in particular for SAVI instances that are located on a switch, since it may significantly increase the cost of such a switch.

To reduce memory requirements for SAVI instances that are located on a switch, the SAVI method enables the suppression of binding replication on links with multiple SAVI instances. This requires manual disabling of IP source address validation on switch ports that connect to other switches running a SAVI instance. Each SAVI instance is then responsible for validating IP source addresses only on those ports to which hosts attach either directly, or via switches without a SAVI instance. On ports towards other switches running a SAVI instance, IP source addresses are not validated. The switches running SAVI instances thus form a "protection perimeter". The IP source addresses in packets passing the protection perimeter are validated by the ingress SAVI instance, but no further validation takes place as long as the packets remain within, or leave the protection perimeter.

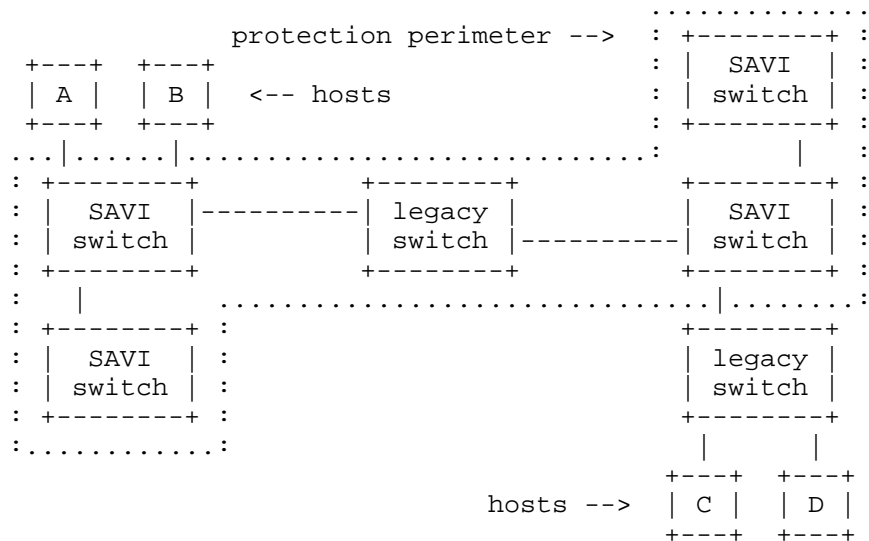


Figure 1: Protection perimeter concept

Figure 1 illustrates the concept of the protection perimeter. The figure shows a link with six switches, of which four, denoted "SAVI

switch", run a SAVI instance. The protection perimeter created by the four SAVI instances is shown as a dotted line in the figure. IP source address validation is enabled on all switch ports on the protection perimeter, and it is disabled on all other switch ports. Four hosts, denoted A through D in the figure, attach to the protection perimeter.

In the example of figure Figure 1, the protection perimeter encompasses one of the legacy switches, located in the middle of the depicted link topology. This enables a single, unpartitioned protection perimeter. A single protection perimeter minimizes memory requirements for the SAVI instances because every binding is kept only once, namely, by the SAVI instance that attaches to the host being validated. Excluding the legacy switch from the protection perimeter would result in two smaller protection perimeters to the left and to the right of the depicted link topology. The memory requirements for the SAVI instances would then be higher: Since IP source address validation would be activated on the two ports connecting to the legacy switch, the SAVI instances adjacent to the legacy switch would replicate all bindings from the respectively other protection perimeter. The reason why it is possible to include the legacy switch in the protection perimeter is because the depicted link topology guarantees that packets cannot enter the protection perimeter via this legacy switch. Without this guarantee, the legacy switch would have to be excluded from the protection perimeter in order to ensure that packets entering the protection perimeter undergo IP source address validation.

5. Reliability Optimizations

The explicit storage of legitimate IP addresses in the form of bindings implies that failure to create a binding, or the premature removal of bindings, can lead to loss of legitimate packets. There are three situations in which this can happen:

- o Legitimate IP address configuration packets, which should trigger the creation of a binding in a SAVI instance, are lost before reaching the SAVI instance.
- o A SAVI instance loses a binding, for example, due to a restart.
- o The link topology changes, resulting in hosts to communicate through SAVI instances that do not have a binding for those hosts' IP addresses.

To limit the disruption that missing bindings for legitimate IP addresses can have, the SAVI method includes a mechanism for reactive

binding creation based on regular packets. This mechanism supplements the proactive binding creation based on IP address configuration packets. Reactive binding creation occurs when a SAVI instance recognizes excessive drops of regular packets originating from the same IP address. The SAVI instance then verifies whether said IP address is unique on the link. How the verification is carried out depends on the IP address configuration method that the SAVI instance supports: The SAVI method variant for Stateless Address Autoconfiguration and for Secure Neighbor Discovery verifies an IP address through the Duplicate Address Detection procedure. The SAVI method variant for DHCP verifies an IP address through a DHCP Lease Query message exchange with the DHCP server. If verification indicates that the IP address is unique on the link, the SAVI instance creates a binding for the IP address. Otherwise, no binding is created, and packets sent from the IP address continue to be dropped.

6. Acknowledgment

The author would like to thank the SAVI working group for a thorough technical discussion on the design and the framework of the SAVI method, as captured in this document, in particular Erik Nordmark, Guang Yao, Eric Levy-Abegnoli, and Alberto Garcia. Thanks also to Torben Melsen for reviewing this document.

This document was generated using the xml2rfc tool.

7. References

- [BA2007] Baker, F., "Cisco IP Version 4 Source Guard", IETF Internet draft (work in progress), November 2007.
- [BCP38] Paul, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", RFC 2827, BCP 38, May 2000.

Authors' Addresses

Jianping Wu
CERNET
Computer Science, Tsinghua University
Beijing 100084
China

Email: jianping@cernet.edu.cn

Jun Bi
CERNET
Network Research Center, Tsinghua University
Beijing 100084
China

Email: junbi@cernet.edu.cn

Marcelo Bagnulo
Universidad Carlos III de Madrid
Avenida de la Universidad 30
Leganes, Madrid 28911
Spain

Email: marcelo@it.uc3m.es

Fred Baker
Cisco Systems
Santa Barbara, CA 93117
United States

Email: fred@cisco.com

Christian Vogt (editor)
Ericsson
200 Holger Way
San Jose, CA 95134
United States

Email: christian.vogt@ericsson.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 28, 2011

M. Bagnulo
A. Garcia-Martinez
UC3M
October 25, 2010

SEND-based Source-Address Validation Implementation
draft-ietf-savi-send-04

Abstract

This memo describes SEND SAVI, a mechanism to provide source address validation using the SEND protocol. The proposed mechanism is intended to complement ingress filtering techniques to provide a higher granularity on the control of the source addresses used.

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 April 28, 2011.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Non-normative Background to SEND SAVI	3
2.1. Address Validation Scope	3
2.2. SEND SAVI Enforcement Perimeter	4
2.3. Binding Creation for SEND SAVI	5
3. Perimeter Configuration Guidelines for SEND SAVI	6
4. SEND SAVI Specification	9
4.1. SEND SAVI Data Structures	9
4.2. SEND SAVI Device Configuration	11
4.3. SEND SAVI Algorithm	11
4.3.1. Traffic Processing	11
4.4. VLAN Support	22
4.5. Protocol Constants	22
5. Security Considerations	22
6. Acknowledgments	25
7. Normative References	25
Authors' Addresses	26

1. Introduction

This memo describes SEND SAVI (Source-Address Validation Implementation), a mechanism to provide source address validation for IPv6 networks using the SEND protocol [RFC3971]. The proposed mechanism is intended to complement ingress filtering techniques to provide a higher granularity on the control of the source addresses used.

SEND SAVI uses DAD_NSOL (Duplicate Address Detection Neighbor Solicitation), DAD_NADV (DAD Neighbor Advertisement), NUD_NSOL (Neighbor Unreachability Detection NSOL) and NUD_NADV (NUD Neighbor Advertisement) messages to validate the address ownership claim of a node. Ports and other layer-2 binding anchors can be associated to the IPv6 address of the neighbor, so that source address validation could be performed.

Scalability of a distributed SAVI system comprised of multiple SEND SAVI devices is preserved by means of a deployment scenario in which SEND SAVI devices form a "protection perimeter" and validation is only performed when the packet ingress to the protection perimeter.

The SEND SAVI specification, as defined in this document, is limited to links in which every IPv6 host and every IPv6 router uses the SEND protocol [RFC3971] to protect the exchange of Neighbor Discovery information. However, SEND SAVI is designed to be deployed in existing SEND networks requiring a minimum set of changes. In particular, SEND SAVI does not require any changes in the hosts whose source address is to be verified. Any verification must solely rely in the usage of already available protocols. This means that SEND SAVI does neither 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.

An overview of the general framework about Source Address Validation Implementation is presented in [I-D.ietf-savi-framework].

2. Non-normative 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 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 addresses as the source address. This is the so-called local traffic, while routers send packets containing a source address other than their own, since they are forwarding 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 address of the packets generated by the hosts attached to the local link has not been spoofed. In addition, since SEND does provide the means to verify that a node claiming to act as a router is indeed authorized to act as one, SEND SAVI also provides the means to verify that packets containing off-link prefixes in the source address are forwarded by authorized routers. However, SEND SAVI does not provide the means to verify if a given router is actually authorized to forward packets containing a specific off-link source address. Other techniques, like ingress filtering [RFC2827], are recommended to validate transit traffic. Hence, the security level is increased by the use of both SAVI and ingress filtering.

2.2. SEND SAVI Enforcement Perimeter

SAVI devices prevent address spoofing by verifying that a layer-2 anchor is associated to the IPv6 address used as source address for the packets being exchanged. The layer-2 anchor, which must be difficult to spoof can be the port of the layer-2 switch through which a packet containing a given IPv6 address is received, or a layer-2 address. In this document we assume that MAC-specific mechanisms to secure data packets, such as IEEE 802.1AE, are not generally available, so SEND SAVI is defined to operate with ports as the only available layer-2 anchor.

In order to reduce computing and state requirements in SEND SAVI devices, SEND SAVI is designed according to the perimetrical protection deployment model presented in the SAVI framework document [I-D.ietf-savi-framework]. In this model, source address validation is performed only when packets enter in a protected realm defined through the protection perimeter. This perimeter must be deployed in such a way that packets for which validation must be performed can only enter in the protected realm through a port belonging to the border performing the validation. The perimeter is defined by appropriate configuration of the roles of each port, which can be Validating ports and Trusted ports. Validating ports are the ports forming the protection perimeter, so they are the ports in which validation for incoming packets is performed. Trusted ports (TPs) are those in which SEND SAVI filtering is not performed.

2.3. Binding Creation for SEND SAVI

Filtering is performed according to the bindings existing between a layer-2 anchor and an IPv6 address. These bindings should allow legitimate nodes to use the binding IPv6 address as source address, and prevent illegitimate nodes to do so. When a protection perimeter is defined, the binding must be created for a port of the border to which a legitimate node is attached to, and must not be created in other case.

SEND provides tools to assure that a ND message containing a CGA option and signed by a RSA option has been generated by the legitimate owner of the CGA IPv6 address. It also provides tools to verify that a RADV message signed by a RSA option with a key bounded to a CGA or a certificate has been generated by a legitimate router.

SEND SAVI benefits from SEND ability to prove address ownership and router authorization to create SAVI bindings. SEND SAVI assumes that a successfully validated SEND message ingressing to the protection perimeter from a validating port guarantees that the host legitimately issuing the message is connected to that port. In this case, a binding for the host to this layer-2 port is created.

The events that trigger the binding creation process in a Validating port of a SEND SAVI device are:

- o The reception from a Validating port of a DAD_NSOL message, indicating the attempt of a node to configure an address. This may occur when a node configure an address after being idle for sometime, or because the node has changed the physical attachment point to the layer-2 infrastructure.
- o The reception from a Validating port of any other packet (including data packets) with a source address for which no binding exists. This would occur if a DAD_NSOL message was lost before arriving to the Validating port, or if a node has changed the physical attachment point to the layer-2 infrastructure without issuing a DAD_NSOL message.

When the binding creation process is triggered, the SEND SAVI device has to assure that the host for which the binding is to be created is the legitimate owner of the address. For a binding creation process initiated by a DAD_NSOL exchange, the messages to consider for address ownership validation are other DAD_NSOL messages arriving from other locations or a DAD_NADV message indicating that other host has configured the address before. For other packets initiating the creation of the binding, the SEND SAVI device asks the host to prove address ownership by issuing a NUD_NSOL which has to be answered by a NUD_NADV by the probed node. Note that it is not required to ask other SEND SAVI devices, as it is done in the non-SEND FCFS

specification [I-D.ietf-savi-fcfs], since in this case a SEND host can prove authoritatively the ownership of its address.

Bindings are refreshed periodically by means of a NUD_NSOL message issued by the SEND SAVI device through the bounded port which has to be answered by a valid NUD_NADV message by the node for which the binding exist.

SEND SAVI could be sensible to replay attacks, i.e. situations in which a secured SEND message is replayed by a non-legitimate node. For example, a node could immediately re-inject a valid SEND message being received from other node, to force the creation of a binding for which it is not authorized. SEND provides some means to prevent the replaying of ND messages, in particular, the use of nonces to validate advertisements that were previously solicited, and the use of timestamps to validate solicitation messages and unsolicited advertisements. However, the emphasis for SEND anti-replay protection is to assure that confidence in some information (for example, the relationship between an IPv6 address and a layer-2 address) is not hold for more time than reasonable, while in SEND SAVI truthful information (in SEND sense, like the relationship of an IPv6 address and a layer-2 address) can be used to create a SAVI binding in a time span shorter than the time reasonable to consider the information aged. As a consequence, SEND SAVI relies only in messages with a low chance of being replayed from different ports to the legitimate one and still being considered valid by SEND. The messages being used by SEND SAVI to create bindings are:

- o Unsolicited DAD_NSOL messages. According to the SEND SAVI specification Section 4.3.1.1 These messages can only be forwarded to ports through which a previous binding for the same IPv6 address existed.
- o NUD_NADV messages in response to a NUD_NSOL sent by the SEND SAVI device, both exchanged through the same Validating port. In this case, anti-replay protection is assured by nonce exchange. This message exchange is also used to refresh the binding.

Any validated RADV can be used to determine that a node for which a binding exists in a Validating port is a router, since the topological part of the binding has been assured before. In addition, the acquisition of prefix information, required to determine local and transit traffic, is not tied to topological considerations too, so for this case regular SEND validating rules are applied.

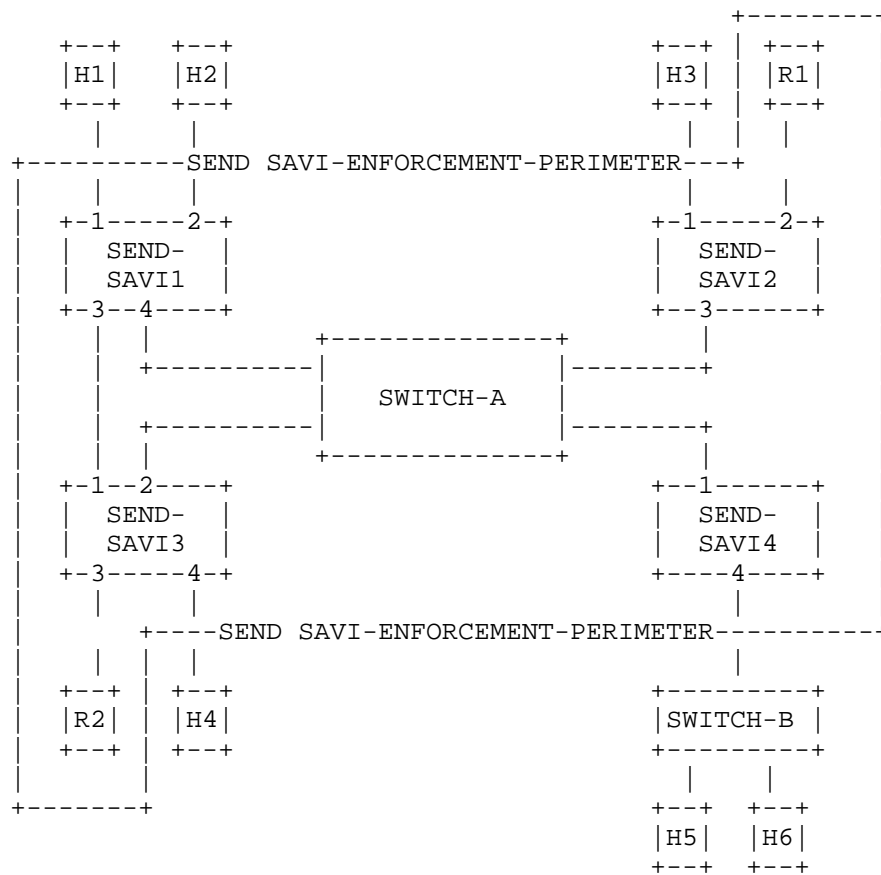
3. Perimeter Configuration Guidelines for SEND SAVI

As it has been discussed before, the perimeter is defined by

appropriate port configuration. Ports in SEND SAVI devices may assume two roles according to its behavior when filtering and validating SEND messages: Validating ports and Trusted ports.

- o Validating ports (VPs) are those in which SEND SAVI filtering and binding creation is performed.
- o Trusted ports (TPs) are those in which neither SEND SAVI filtering nor binding creation are performed. So, packets received through Trusted ports are not filtered by SEND SAVI. The only SEND messages received through a Trusted port which are processed are those related with certificates, prefix information and Neighbor Advertisements for Duplicate Address Detection (DAD_NADV).

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



Trusted ports are used for connections with trusted infrastructure, including the communication between SEND SAVI devices, the communication with other switches which are not SEND SAVI devices, routers or other trusted nodes.

Port 3 of SEND-SAVI1 and port 1 of SEND-SAVI3 are trusted because they connect two SAVI devices. Port 4 of SEND-SAVI1, port 3 of SEND-SAVI2, port 2 of SEND-SAVI3 and port 1 of SEND-SAVI4 are trusted because they connect to SWITCH-A to which only trusted nodes are connected. Port 2 of SEND-SAVI2 and port 3 of SEND-SAVI3 are trusted ports, because they connect to routers.

Validating ports are used for connection with non-trusted infrastructure. Therefore, hosts are normally connected to validating ports. Non-SEND SAVI switches that are outside of the SAVI enforcement perimeter also are connected through validating port. In particular, non-SEND SAVI devices which connect directly to hosts or which have no SEND SAVI capable device between themselves and the hosts are connected through a validating port. 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 SWITCH-B which is a non-SEND SAVI capable switch which is connected to hosts H5 and H6.

SEND SAVI requires all devices performing SAVI function to implement SEND SAVI (for example, coexistence with non-SEND aware FCFS SAVI [I-D.ietf-savi-fcfs] switches is not allowed).

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

- o Ports that are connected to another SEND SAVI device SHOULD be configured as Trusted ports. Not doing so will at least 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.
- o Ports connected to hosts SHOULD be configured as Validating ports. Not doing so will allow the host connected to that port to send packets with spoofed source address.
- o Ports connected to routers SHOULD be configured as Validating ports. However, the SEND SAVI specification also allows the routers to be connected to Trusted ports, as they are assumed to be part of the trusted infrastructure. When connected through a trusted port, a router can generate traffic with any source address, even those belonging to the link, while when connected through a Validating port it can only send traffic using off-link source addresses, or its own source addresses. When routers are connected to Validating, authorization for the routing function is

- bound to the router itself, instead of being bound to a port configured in a switch.
- o Ports connected to a chain of one or more legacy switches that have hosts connected SHOULD be configured as Validating ports. Not doing so will allow the host connected to any of these switches to send packets with spoofed source address.
 - o Ports connected to a chain of one or more legacy switches that have other SEND SAVI devices and/or routers connected but had no hosts attached to them SHOULD be configured as Trusted ports. Not doing so will at least significantly increase the memory consumption in the SEND SAVI devices and increase the signaling traffic due to SEND SAVI validation.
 - o Ports connected to a chain of one or more legacy switches that have a mix of SEND SAVI devices and/or routers with 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.

4. SEND SAVI Specification

4.1. SEND SAVI Data Structures

The following data structures are defined for SEND SAVI operation:

SEND SAVI Port list. This structure defines an entry per port in the SAVI device. Each entry indicates the role configured for the port (Trusted port or Validating port). In addition, only for Validating ports, the entry indicates the presence or absence of a router connected through the port has been stated by successful validation of a RADV message received from this port. This data structure is used to determine the filtering behavior for each port when local-link and off-link traffic is received. If the port is a Trusted Port, both local-link and off-link traffic coming from the port is accepted. If the port is a Validating port but not a Routing port, then only local-link traffic coming from the port for which a binding exists is accepted. If the port is a Validating port and a Routing port, then off-link traffic coming from the port is accepted, but only local-link traffic coming from the port for which a binding exists is accepted.

Each entry of the SEND SAVI Port list contains the following information:

- o Layer-2 Validating port
- o Configured port role (either Trusted port or Validating port). The default configuration is Validating port.

- o Router port bit. This value is only meaningful for ports with a configured port role set to Validating port. It indicates whether a RADV message received from the port has been successfully validated, indicating that a router is connected to the port. For this bit to be set, an entry containing this layer-2 port MUST exist in the SEND SAVI Address table for an IP address of an entry in the SEND SAVI Router table.

SEND SAVI Address list. The SEND SAVI function relies on state information binding the source IPv6 address used in data packets to the port through which the legitimate host connects. Such information is stored in SEND SAVI Address table. The SEND SAVI Address table contains one entry for each of the IPv6 source addresses in use on a Validating port of the SEND SAVI device. The SEND SAVI Address list is populated with the contents of successfully validated SEND messages. Each entry contains the following information:

- o IP source address
- o Layer-2 Validating port to which the host is connected.
- o Lifetime
- o Status: TENTATIVE_DAD, TENTATIVE_NUD, VALID, TESTING_VP, TESTING_VP'.

SEND SAVI Prefix list. In addition to this, a SEND SAVI device needs to know which are the link prefixes in order to identify local and off-link traffic. This information is obtained from validated RADV messages. This information is not specific to a given port. Note that the information in this table is equivalent to the Prefix List conceptual data structure defined in [RFC4861]. The SEND SAVI Prefix list contains one entry per prefix in use, as follows:

- o Prefix
- o Lifetime

SEND SAVI Router list. SEND SAVI keeps a table with one entry for each authorized router in use connected to a Validating port of the SAVI device. In particular, it contains the address for which a successfully validated RADV has been received. The information in this table is used to populate the SEND SAVI port table when at least one router has been validated in a layer-2 Validating port (the layer-2 port can be obtained by looking-up for the IPv6 address of the router in the SEND SAVI Address list). It can also be used to issue a RSOL in case the entry is about to expire, in order to ensure that the node is still performing as a router. Note that the information in this table is equivalent to the Default Router List conceptual data structure defined in [RFC4861]. The information stored in the table is the following:

- o Router IPv6 address
- o Lifetime

4.2. SEND SAVI Device Configuration

In order to perform SEND SAVI operation, some basic parameters of a SEND SAVI device have to be configured.

A SEND SAVI device operates as a full-fledged SEND node in some cases: it may generate NUD_NSOL, RSOL or CPS messages. Therefore, a SEND SAVI device

- o MUST be configured with a valid CGA address. Note that when the SEND SAVI device configures this address, it must follow the same rules as regular SEND hosts (such as using secured NSOL messages to perform DAD, etc.)
- o MUST be configured with at least one Trust anchor to validate the Certification Paths that authorizes route operation.
- o MUST be configured with Certification Paths, either manually or by means of issuing Certification Path Solicitation messages, as detailed in the SEND specification [RFC3971].

In addition, the port role for each port of the SEND SAVI Port list SHOULD be configured. Otherwise, every port would be labeled as Validating port, and performance may be degraded, as discussed in [I-D.ietf-savi-framework].

4.3. SEND SAVI Algorithm

4.3.1. Traffic Processing

In this section we describe how packets are processed.

First, the source address of packet is analysed to determine if it is local or transit traffic, by checking if the prefix of the source address is included in the SEND SAVI Prefix List (local traffic) or not included (transit traffic). A special case of local traffic is the traffic destined to the SEND SAVI device itself, either specifically, or through a multicast address to which the SEND SAVI device is registered (such as the all-nodes address, ff02::1).

Transit traffic processing occurs as follows:

- o If the transit traffic packet is received through a Trusted port, the data packet is forwarded and no SAVI processing performed.
- o If the transit traffic packet is received through a Validating port, the packet is only forwarded if the port appears with the Routing bit set in the SEND SAVI Port list, indicating that a router has been validated through SEND procedures at this port. If transit traffic is received from a Validating port, and the

port does not appear with the Routing bit set in the SEND SAVI Port list, the SAVI SEND device SHOULD send a RSOL message through the considered port.

Processing of traffic addressed to the SEND SAVI device itself occurs as follows:

- o Packets received from Trusted ports are not filtered. In particular, if a successfully validated CPA message is received through a Trusted port, the certificate information is accepted by the SEND SAVI device. If a successfully validated RADV message is received through a Trusted port, the SEND SAVI Prefix list in the SEND SAVI device is updated accordingly.
- o NUD_NADV messages corresponding to SEND SAVI operation are processed according to the specification of Section 4.3.1.1.
- o Packets received from Validating ports are only processed by the SEND SAVI device if a binding exists for the source IPv6 address of the packet, and the state for the binding is VALID or TESTING (see next section). In particular, If a successfully validated RADV message is received through a Trusted port, the SEND SAVI Prefix list in the SEND SAVI device is updated accordingly. The Router bit of the SEND SAVI Port list is set only if the destination address of the RADV message is not a multicast address. If a SEND SAVI device receives a RADV sent to a multicast address, it SHOULD issue a RSOL message to the port through which this message has been received.

We next consider how local traffic is processed.

4.3.1.1. Processing of Local Traffic

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:

- o When it is stated that a secured NUD_NSOL message is issued by a SEND SAVI device through a given port, this means the following: the SEND SAVI device performs a Neighbor Unreachability Detection procedure as described in [RFC4861] with SEND secured messages as defined in [RFC3971] addressed to the IPv6 target address (source address of the packet triggering the procedure). The source address used for issuing the NUD_NSOL is the source address of the SEND SAVI device.
- o When it is stated that a validated NUD_NADV message is received by a SEND SAVI device through a port P, this means that: a SEND secured NUD_NADV message has been received by the same port 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 being 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 for Trusted port.

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

The possible states are

- o NO_BIND. This state represents that no binding exists for the address. This is the state for all addresses unless a binding is explicitly created.
- o TENTATIVE_DAD. This state is reached when the SEND SAVI device has received a validated DAD_NSOL message. The SEND SAVI device waits for a possible DAD_NADV. Packets with the source address of the binding are not forwarded.
- o TENTATIVE_NUD. A packet different from a valid DAD_NSOL message has been received from port VP and the SEND SAVI device has sent a NUD_NSOL message to the port. Packets with the source address of the binding are not forwarded.
- o VALID. The binding for the source address has been verified. Packets with the source address of the binding are forwarded.
- o TESTING_VP. The lifetime of the binding has expired so SEND SAVI device has sent a NUD_NSOL message to the port, or a DAD_NSOL coming from other SEND SAVI device has been received. The SEND SAVI device waits for a validated NADV. Packets with the source address of the binding are allowed to be forwarded.
- o TESTING_VP'. A validated DAD_NSOL message has been received from a Validating port of the SEND SAVI device. The device waits for a DAD_NADV coming from port VP, or changes the binding to port VP' if no response is received after TENT_LT milliseconds. Packets coming from port VP with the source address of the binding are allowed to be forwarded.

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

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

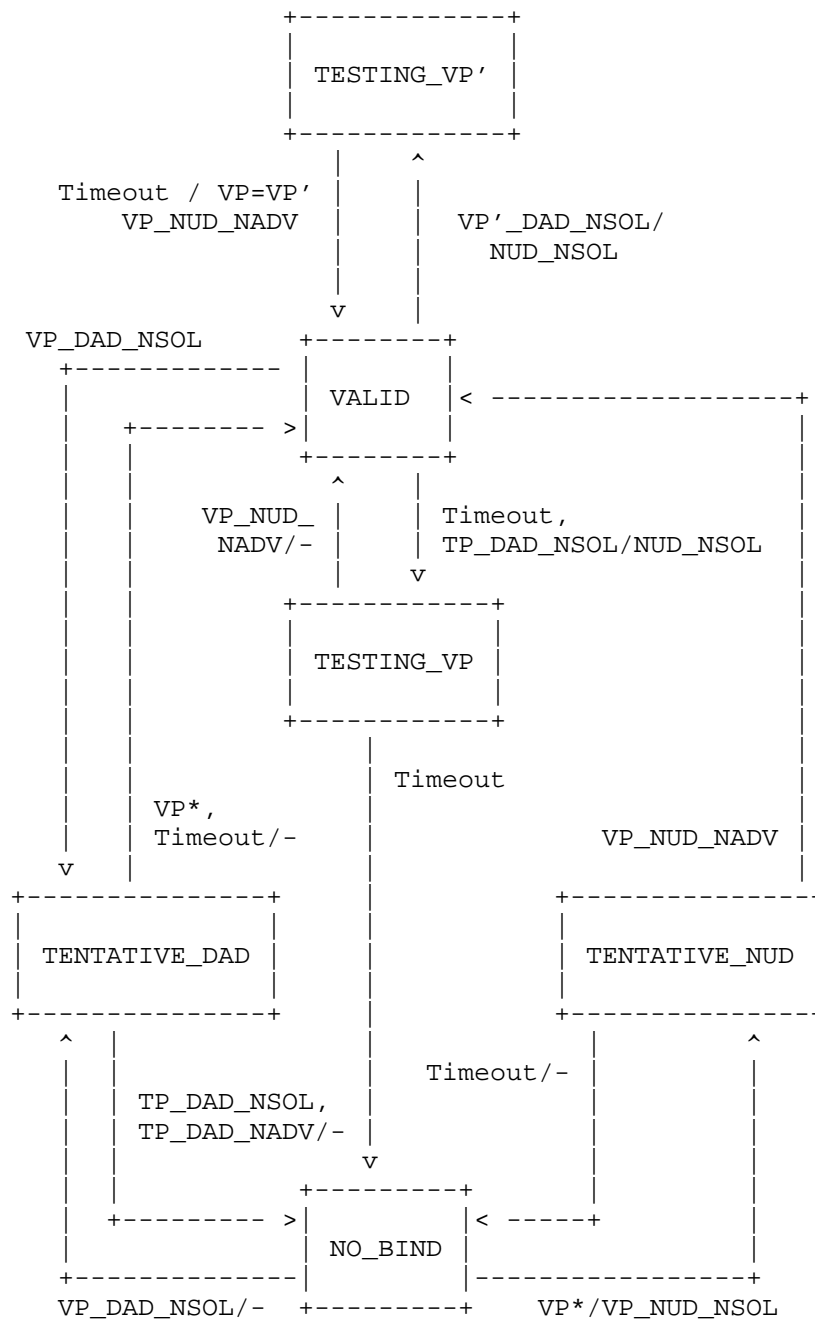
- o The only events which triggers state changes from forwarding to non-forwarding states and vice versa are the reception of DAD_NSOL, DAD_NADV and NUD_NADV, or the expiration of a timer. Besides, DAD_NADV and NUD_NADV are only processed when expected as a response to a DAD_NSOL or a NUD_NSOL message. The other possible input to consider is 'any other packet', which could generate changes to states belonging to the same class as the original state (i.e. when 'any other packet' is received, the state cannot move from being forwarding to non-forwarding and vice versa). Note that non-validated SEND messages always belong to the 'any other packet' category. The reduced set of messages being able to trigger a change simplifies the processing at SEND SAVI devices. It is also convenient for defining a comprehensive model regarding to anti-replay protection.
- o The SEND SAVI device is only required to generate NUD_NSOL messages for SEND SAVI operation. This also simplifies the state machine.
- o Well-behaved hosts are expected to initiate communication by sending secured DAD_NSOL messages. The SEND SAVI state machine is designed to process these events in an optimal way. The reception of other packet types without receiving previously validated DAD_NSOL messages is assumed to be the result of either bad-behaving hosts or the lost of packets. While these events may occur and a binding will ultimately be created for such hosts, the case in which data packets are received without receiving previously a DAD_NSOL message is not always optimized, for the sake of simplicity of the state machine. It is also worth to note that a validated DAD_NSOL provides a reliable hint about the address ownership of a host attached to a given port, while this is not the case for data packets, for example.
- o If a host has an address configured, and it can prove the ownership of this address, the state 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), if accidentally two hosts generate the same address and the DAD procedure has failed. In these unfrequent cases, connectivity is honored over security.

The SEND 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 SEND SAVI device will receive the DAD_NSOL for those addresses. Please note that it may not be enough to relay 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 SEND SAVI device. So, the SAVI device SHOULD join the solicited node multicast groups

for all the addresses that are in a state other than NO_BIND.

We next describe how different inputs are processed depending on the state of the binding of the IP address 'IPaddr'. A Waiting_lifetime timer is associated to each binding.

A simplified version is depicted in the next figure:



NO_BIND

Relevant inputs for this state: When the node is in this state, there are no unresolved DAD_NSOL or NUD_NSOL messages (generated by SEND SAVI), so the only relevant inputs are DAD_NSOL messages coming either from VP or TP, or any packet other than DAD_NSOL coming from VP or TP. There are no timers too.

- o 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, and with the restrictions imposed by the MLD snooping mechanism, if applied). The DAD_NSOL messages are not sent through any of the ports configured as Validating Ports. The SEND SAVI device sets the Waiting_timer to TENT_LT, stores all the information required for future validation of the corresponding DAD_NADV message (such as the nonce of the message) and changes the state to TENTATIVE_DAD. Note that in this case it is not possible to check address ownership by sending a NUD_NSOL because while the host is waiting for a possible DAD_NADV its address is in tentative state and it cannot respond to NSOL messages ([RFC4862]).
- o 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 Waiting_timer to TENT_LT. The state is changed to TENTATIVE_NUD.
- o Validated DAD_NSOL message containing IPAddr as the target address received through a Trusted port are NOT forwarded through any of the Validating ports but they are sent through the proper Trusted Ports (as defined by the switch behavior that will depend on whether it performs MLD snooping or not). The SEND SAVI device MAY assume that any DAD_NSOL message received from a Trusted port has been successfully validated by other SEND SAVI device, so that no additional validation is required. The state is not changed.
- o 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 attachment of the host to its Validating port 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 port VP and forwarded to TP. The possible events occurring in this state are: the reception of a DAD_NADV message from a TP (the corresponding DAD_NSOL was forwarded to this port), a DAD_NSOL message from VP, other validating port VP' or TP, and the expiration of the timer initiated when the DAD_NSOL was received at port VP.

- o If a validated DAD_NADV is received from a Trusted port, the binding cannot be configured for port VP. The state is changed to NO_BIND, and the Waiting_timer cleared.
- o If a validated DAD_NSOL is received from a Trusted port, a host 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 port VP, so that the host at port VP will not configure the address, as stated in [RFC4862]. The DAD_NSOL message is also forwarded to all appropriate Trusted ports. Then, the Waiting_timer is cleared, and the state is changed to NO_BIND.
- o 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 attachment of the host to its Validating port according to SEND SAVI rules (unless the SEND SAVI perimeter has been breached). The state is not changed.
- o If a validated DAD_NSOL is received from a Validating port VP' different to VP, a host connected to VP' may be trying to configure the same address at the same time. The DAD_NSOL message is forwarded to port VP, so that the host at port VP will not configure the address, as stated in [RFC4862]. The DAD_NSOL message is also forwarded to all appropriate Trusted ports. Then, the Waiting_timer is set to TENT_LT, and the state remains in TENTATIVE_DAD, although in this case with VP=VP'.
- o Any other packet than a validated DAD_NSOL is received from a Validating port VP' different from VP is discarded. The state is not changed.
- o If a validated DAD_NSOL is received from port VP, the Waiting_timer is set to TENT_LT, and the state remains in TENTATIVE_DAD.
- o If any packet other than a validated DAD_NSOL is received from VP, it is assumed that the host 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 host proved address ownership by means of the validated DAD_NSOL message, the binding is created. The Waiting_timer is set to DEFAULT_LT, and the state is changed to VALID.
- o If Waiting_timer expires, it is assumed that no other host has configured this address. Therefore, the Validating port VP could be bound to this IPv6 address. The Waiting_timer is set to DEFAULT_LT, and the state is changed to VALID.

VALID

To arrive to this state, successful validation of address ownership has been completed. Relevant transitions for this state are triggered by the reception of DAD_NSOL from ports VP, VP' or TP, and any packet other than DAD_NSOL from VP' or TP. The expiration of

Waiting_timer is also relevant to check again for address ownership.

- o If a validated DAD_NSOL with IPAddr as source address is received through Validating port VP, this message is forwarded to the appropriate trusted ports. The Waiting_timer is set to TENT_LT and the state is changed to TENTATIVE_DAD.
- o Any packet other than a validated DAD_NSOL containing IPAddr as a source address arriving from Validating port VP is forwarded appropriately. The state is not changed. Note that in the SEND SAVI case Timeout_valid for the entry MUST NOT be set to DEFAULT_LT (as occurs for FCFS SAVI), since regular sending of packets does not provide the required security, which is achieved by performing secured NUD periodically with the sending host.
- o If a validated DAD_NSOL with IPAddr as source address is received through a Trusted port, the message is forwarded to VP. The Waiting_timer is set to TENT_LT, a secured NUD_NSOL message is sent to IPAddr through VP and the state is changed to TESTING_VP.
- o 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 VP, the Waiting_timer is set to TENT_LT, and the state is changed to TESTING_VP.
- o If a DAD_NSOL packet with IPAddr as source address is received through a Validating Port VP' (VP' different from the current Validating port for this binding), the message is forwarded to VP. In addition, a secured NUD_NSOL is sent to VP, the Waiting_timer is set to TENT_LT, and the state is changed to TESTING_VP'.
- o If any packet other than a DAD_NSOL with IPAddr as source address is received from a Validating Port VP', different from the current Validating port for this binding, VP, the packet is discarded. The SEND SAVI device MAY issue a secured NUD_NSOL through port VP, set the Waiting_timer 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, and a packet arrives from VP').
- o If Waiting_timer expires, a secured NUD_NSOL message is sent through port VP to the IPv6 address, the Waiting_timer 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 secured NUD_NADV message from VP, the reception of a secured 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 waiting for NUD_NADV.

- o If a validated NUD_NADV message is received from VP, the message is discarded, the Waiting_timer is changed to DEFAULT_LT, and the state is changed to VALID.
- o If a validated DAD_NSOL message is received from VP, the message is forwarded to the appropriate Trusted ports, the Waiting_timer is set to DEFAULT_LT, and the state is changed to TENTATIVE_DAD.
- o Any packet other than DAD_NSOL or NUD_NADV containing IPAddr as a source address arriving from Validating port VP is forwarded. Neither the Waiting_timer nor the state are changed.
- o If a DAD_NSOL message is received from a Trusted port, the message is forwarded to VP and the appropriate Trusted ports. Neither the Waiting_timer nor the state are changed. The host at VP port is under check: if it still is at port VP, 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, the local state will move to NO_BIND, and the state at the remote node will change to VALID.
- o If a packet other than a DAD_NSOL arrives from a Trusted port, the packet is forwarded. Neither the Waiting_timer nor the state are changed.
- o If a DAD_NSOL is received from a validating port VP', the message is forwarded to VP and the appropriate Trusted ports. In addition, a secured NUD_NSOL is sent to VP, the Waiting_timer is set to TENT_LT, and the state is changed to TESTING_VP'.
- o Any other packet received from a validating port VP' is discarded. This may occur because the host 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'.
- o If the Waiting_timer expires, the Waiting_timer is cleared and the state is changed to NO_BIND.

TESTING_VP'

To arrive to this state an indication that a host at VP' wants to send data with IPAddr as source address while a binding existed for VP. The SEND SAVI device has issued a NUD_NSOL to the host through port VP. The possible events that may occur in this case are the reception of a NUD_NADV from port VP, 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.

- o If a validated NUD_NADV is received from port VP, then the host at VP is defending its address. VP is kept as the Validating port, the Waiting_timer is set to DEFAULT_LT, and the state is changed to VALID.
- o If a validated DAD_NSOL is received from port VP, the message is forwarded to VP'. The Waiting_timer is set to TENT_LT and the state is changed to TENTATIVE_DAD. If the host at VP is reconfiguring its address; when forwarding the DAD_NSOL message, the node at VP' is expected to unconfigure its address.
- o Any packet other than a validated DAD_NSOL or a validated NUD_NADV coming from port VP is forwarded, and the state is not changed.
- o If a validated DAD_NSOL is received from port VP', the message is forwarded to VP. The Waiting_timer is set to DEFAULT_LT, and the state is not changed.
- o Any packet other than a validated DAD_NSOL coming from port VP is discarded, and the state is not changed.
- o If a validated DAD_NSOL is received from port VP", different from VP and VP', the message is forwarded to VP and VP'. VP' is expected to unconfigure its address if it was a VP'_DAD_NSOL message (and not any other packet) the message triggering the transition to this state. The state remains in TESTING_VP' although with VP'=VP". The Waiting_timer is not changed.
- o Any packet other than a validated DAD_NSOL received from port VP" is discarded and does not affect to the state.
- o If a validated DAD_NSOL is received from a Trusted port, the message is forwarded to ports VP, VP' and other appropriate Trusted ports. The Waiting_timer is left unchanged and the state is changed to TESTING_VP. VP' is expected to unconfigure its address if it was a VP'_DAD_NSOL message (and not any other packet) the message triggering the transition to this state.
- o Any packet other than a validated DAD_NSOL coming from a Trusted port is forwarded appropriately, but the state is not changed.
- o If Waiting_timer expires, it is assumed that the host for which the binding existed is no longer connected through port VP. Therefore, the Validating port VP' could be bound to this IPv6 address. The Waiting_timer 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 port VP without any existing binding in the SEND SAVI device. The SEND SAVI device has sent a NUD_NSOL message to VP. The relevant events for this case are the reception of a NUD_NADV from port VP, the reception of DAD_NSOL from port VP, VP' or TP, and the reception of any packet other than DAD_NSOL and NUD_NADV for port VP, and different from DAD_NSOL for VP' or TP. In addition, the Waiting_timer may expire.

- o If a validated NUD_NADV message is received through port VP, the message is discarded, the Waiting_timer is set to TENT_LT, and the state is changed to VALID.
- o If a validated DAD_NSOL message is received through port VP, the message is forwarded to the appropriate Trusted ports, the Waiting_timer is set to TENT_LT and the state is changed to TENTATIVE_DAD.
- o Any packet other than NUD_NADV or DAD_NSOL received through port VP is discarded.
- o If a validated DAD_NSOL message is received through port VP' different from port VP, the message is forwarded to the appropriate Trusted ports, the Waiting_timer is set to TENT_LT with VP=VP', and the state is changed to TENTATIVE_DAD.
- o Any packets other than DAD_NSOL received through port VP' are discarded, and the state is left unchanged.
- o If a validated DAD_NSOL message is received through a Trusted port, the message is forwarded to port VP, and the state is left unchanged.
- o Any other packet received from a Trusted port are forwarded appropriately. These packets may come from a SEND SAVI device that has securely validated the attachment of the host to its Validating port according to SEND SAVI rules. The state is left unchanged.
- o If Waiting_timer expires, the Waiting_timer is cleared and the state is changed to NO_BIND.

4.4. VLAN Support

In the case the SAVI device is a switch that supports VLANs, the SAVI implementation will behave as if there was one SAVI process per VLAN. The SAVI process of each VLAN will store the binding information corresponding the nodes attached to that particular VLAN.

4.5. Protocol Constants

TENT_LT is 500 milliseconds.

DEFAULT_LT is 5 minutes.

5. Security Considerations

It should be noted that any SAVI solution is as strong as the lower layer anchor that it uses. In particular, if the lower layer anchor is forgeable, then the resulting SAVI solution will be weak. For example, if the lower layer anchor is a MAC address that can be easily spoofed, then the resulting SAVI will not be stronger than that. On the other hand, if we use switch ports as lower layer

anchors (and there is only one host connected to each port) it is likely that the resulting SAVI solution will be considerably more secure.

SEND SAVI improves protection compared to conventional SAVI, as a result of the increased ability of SEND hosts 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. Not doing so will generate security breaches through which an attacker could send packets using any source address, regardless of the bindings established in other SEND SAVI devices. However, SEND SAVI is designed to allow even in this case communication for legitimate users. The worst case for the misconfiguration of the perimeter is then that two hosts may use the same source IPv6 address. The reasons for having a misconfigured perimeter, apart from initial misconfiguration, are the dynamic operations performed by layer-2 routing mechanisms, for example, as a result of a failure in a link or switching device. To prevent the security risks associated, in the case of changes in the topology of the SEND SAVI devices, all ports of a SEND SAVI device MAY be changed automatically to Validating. Note that neither connectivity nor the protection offered are compromised by operating in a mode in which all ports of the SEND SAVI devices operate in Validating mode (only performance is affected by this setting).

SEND SAVI does not protect against spoofers being attached to the same port as a legitimate host. For this reason it is RECOMMENDED that only one host attaches at the same time to a given port.

One possible concern about SEND SAVI is its behavior when an attacker tries to forge the identity of a legitimate host by replaying messages. Note that information that can be valid for SEND a short period after being generated (the binding between an IPv6 address and a layer-2 MAC address) is not valid for SEND SAVI if it arrives from an non-legitimate port. We now perform a security analysis of such a replay attack for SEND SAVI. On one hand, there is some information for which the security risks are equivalent to those of SEND operation, which are situations in which the information received is not tied to port-related state in SEND SAVI operation. Such situations are the reception of CPA messages containing certificates, or the processing of an unsolicited RADV message, which can be used in SEND SAVI to associate the router condition to the IPv6 address of an existing binding in the SEND SAVI Port list. On the other hand, all the messages which can be create a SEND SAVI binding may be

sensible for the replaying of valid SEND messages. 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.

- o To prevent DAD_NSOL replay attacks, DAD_NSOL messages are not forwarded to ports through which an existing binding existed. Therefore, to capture a message that could be used to launch a replay attack, an attacker must be located either in the port through which the legitimate host is (in which case the attack is useless), or in a port in which a legitimate host was before and for which a binding still exists. For this latter case, an attacker can prevent the configuration of binding for a legitimate host in other port (which could have moved from the initial location), and the binding would be available for the attacker for DEFAULT_LT ms. The attacker can do this either in the port for which a binding existed, or in other port to which it is connected or to which it can convey this information for a third node to perform this attack. This risk is inherent to allowing layer-2 host mobility in an scenario in which many hosts can attach to the same port (either at the same time or in instants very close one to the other). Another consideration is that this situation reflect the fact that it is impossible to determine the legitimacy of a node with a more secure NUD_NSOL/NUD_NADV exchange when the nodes claim to be configuring the address.
- o When a NUD_NSOL/NUD_NADV exchange is used to create or maintain a state, the messages are only forwarded to the port in which the host claiming to be legitimate is located. In this case, an attacker must be connected to the same port of the legitimate host to be able to capture a message which could be replayed. The replay of NUD_NSOL is useless, since it is not used to trigger the creation of a binding. The replay of a NUD_NADV message through the same port is useless, since SEND SAVI does not protect against spoofers attached to the same port. The replay of a NUD_NADV message through a different port does result neither in the creation of a binding in other SEND SAVI device, nor in the binding created in the SEND SAVI device originating the NUD_NSOL message, since this SEND SAVI device only considers NUD_NADV message received from the same port through which the NUD_NSOL message was sent.

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 hosts. 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 generate a NUD_NSOL and create a state in which a NUD_NADV is waited for. 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, a rate-limiting mechanism

SHOULD be enforced in a per-port basis.

6. Acknowledgments

Thanks to Ana Kukec for her review and comments on this document. The text has also benefited from feedback provided by Tony Cheneau.

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

Alberto Garcia-Martinez is partly funded by T2C2, a Spanish R&D project.

7. Normative 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.
- [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.
- [I-D.ietf-savi-framework] Wu, J., Bi, J., Bagnulo, M., Baker, F., and C. Vogt, "Source Address Validation Improvement Protocol Framework", draft-ietf-savi-framework-00 (work in progress), September 2010.
- [I-D.ietf-savi-fcfs] Nordmark, E., Bagnulo, M., and E. Levy-Abegnoli, "FCFS-SAVI: First-Come First-Serve Source-Address Validation for Locally Assigned Addresses", draft-ietf-savi-fcfs-05 (work in progress), October 2010.

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>

SAVI
Internet-Draft
Intended status: Informational
Expires: March 12, 2011

D. McPherson
VeriSign, Inc.
F. Baker
Cisco Systems
J. Halpern
Ericsson
September 8, 2010

SAVI Threat Scope
draft-ietf-savi-threat-scope-03

Abstract

This memo discusses threats enabled by IP source address spoofing and discusses a number of techniques aimed at mitigating those threats.

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 March 12, 2011.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Overview	4
2. Glossary of Terms	5
3. Spoofed-based Attack Vectors	6
3.1. Blind Attacks	6
3.1.1. Single Packet Attacks	6
3.1.2. Flood-Based DoS	6
3.1.3. Poisoning Attacks	7
3.1.4. Spoof-based Worm/Malware Propagation	7
3.1.5. Reflective Attacks	8
3.1.6. Accounting Subversion	8
3.1.7. Other Blind Spoofing Attacks	8
3.2. Non-Blind Attacks	9
3.2.1. Man in the Middle (MITM)	9
3.2.2. Third Party Recon	9
4. Current Anti-Spoofing Solutions	9
4.1. Host to link layer neighbor via switch	11
4.2. Upstream routers	11
4.3. ISP Edge PE Router	12
4.4. ISP NNI Router to ISP NNI Router	12
4.5. Spoofing In Local Area Network Segments	12
4.6. Cable Modem Subscriber Access	13
4.7. DSL Subscriber Access	13
4.8. BCP 38	13
4.9. Unicast RPF	13
4.10. Port-based Address Binding	13
4.10.1. Manual Binding	14
4.10.2. Automated Binding	14
4.10.3. IEEE 802.1X	14
4.11. Cryptographic Techniques	14
4.12. Residual Attacks	15
5. Topological Considerations	15
5.1. Address Provisioning Mechanisms	15
5.2. LAN devices with Multiple Addresses	15
5.2.1. Routers	15
5.2.2. NATs	16
5.2.3. Multi-Instance Hosts	16
5.2.4. Multi-LAN Hosts	16
5.2.5. Firewalls	16
5.2.6. Mobile IP	17
5.2.7. Other Topologies	17
5.3. IPv6 Considerations	17
6. Applicability of Anti-Spoofing Solutions	18
6.1. Analysis of Host Granularity Anti-Spoofing	18
7. Existing Techniques for IP Source Address Validation	19
8. Deployment Considerations	20
9. IANA Considerations	21

10. Security Considerations	21
11. Acknowledgements	22
12. References	22
12.1. Normative References	22
12.2. Informative References	22
Authors' Addresses	22

1. Overview

The Internet Protocol, specifically IPv4 [RFC0791] and IPv6 [RFC2460], employ a connectionless hop-by-hop packet forwarding paradigm. A host connected to an IP network that wishes to communicate with another host on the network generates an IP packet with source and destination IP addressing information, among other options.

At the IP Network Layer, or Internet Layer, there is typically no required transactional state when communicating with other hosts on the network. Hosts generating packets for transmission have the opportunity to spoof (forge) the source address of packets which they transmit.

Source address verification is necessary in order to detect and reject spoofed packets and contribute to the overall security of IP networks. In particular, source address verification techniques enable detection and rejection of spoofed packets, and also implicitly provide some assurances that the source address in an IP packet is legitimately assigned to the system that generated the packet.

Solutions such as BCP 38 [RFC2827] provide guidelines for one such technique for network ingress filtering. However, if these techniques are not implemented at the ingress point of the IP network, then the validity of the source address cannot be positively ascertained. Furthermore, BCP 38 only implies source address verification at the Internet Layer, and is most often implemented on IP subnetwork address boundaries. One of the difficulties in encouraging the deployment of BCP 38 is that there is relatively little benefit until it is very widely deployed, which is not yet the case. The local application of the principle of BCP 38 fortunately has local benefit, even before BCP 38 is fully deployed. It also helps get the Internet towards a state where BCP 38 is more widely followed.

It should be noted that while BCP 38 directs providers to provide protection from spoofed prefixes, it is clearly desirable for enterprise operators to provide that protection more locally, and with better traceability. This allows the enterprise to be a better Internet participant, and to quickly detect and remedy problems when they occur.

Also, there is a possibility that in a LAN environment where multiple hosts share a single LAN or IP port on a switch or router, one of those hosts may spoof the source addresses of other hosts within the local subnet. Understanding these threats and the relevant

topologies in which they're introduced is critical when assessing the threats that exist with source address spoofing.

The aim of this document is to provide some additional details regarding spoofed-based threat vectors, and discuss implications of various network topologies.

2. Glossary of Terms

The following acronyms and terms are used throughout this memo.

BGP: The Border Gateway Protocol, used to manage routing policy between large networks.

CPE Router: Customer Premises Equipment Router. The router on the customer premises, whether owned by the customer or the provider. Also called the Customer Edge, or CE, Router.

IP Address: An Internet Protocol Address, whether IPv4 or IPv6.

ISP: Internet Service Provider. Any person or company that delivers Internet service to another.

MAC Address: An Ethernet Address or comparable IEEE 802 series address.

NNI Router: Network to Network Interface Router. This router interface faces a similar system operated by another ISP or other large network.

PE Router: Provider Edge Router. This router faces a customer of an ISP.

Spoofing: The act of forging datagram header contents at the Link or Network Layer

TCP: The Transmission Control Protocol, used on end systems to manage data exchange.

uRPF: Unicast Reverse Path Forwarding. A procedure in which the route table, which is usually used to look up destination addresses and route towards them, is used to look up the source address and ensure that one is routing away from it. When this test fails, the event may be logged, and the traffic is commonly dropped.

3. Spoofed-based Attack Vectors

Spoofing is employed on the Internet for a number of reasons, most of which are in some manner associated with malicious or otherwise nefarious activities. In general, two classes of spoofed-based attack vectors exist: blind attacks and non-blind attacks. The following sections provide some information of blind and non-blind attacks.

3.1. Blind Attacks

Blind attacks typically occur when an attacker isn't on the same local area network as a source or target, or when an attacker has no access to the datapath between the attack source(s) and the target systems. The result is that they have no access to legitimate source and target systems.

3.1.1. Single Packet Attacks

One type of blind attacks, which we'll refer to here as "single packet DoS attacks", involves an attacking system injecting spoofed information into the network which results in either a complete crash of the target system, or in some manner poisons some network configuration or other information on a target system so as to impact network or other services.

An example of an attack that can cause a receiving system to crash is a LAND attack. A LAND attack packet would consist of an attacking system forwarding a packet (e.g., TCP SYN) to a target system that contains both a source and destination address of that target system. The target system would "lock up" when creating connection state associated with the packet, or would get stuck in a state where it continuously replies to itself.

Another class of a single packet attacks involves an attacker poisoning network or DNS cache information, perhaps to simply break a given host's connection, enable MITM or other attacks. Network level attacks that could involve single packet DoS include ARP cache poisoning and ICMP redirects. The most obvious example which depends upon falsifying an IP source address is an on-link attacker poisoning a router's ARP or ND cache. The ability to forge a source address can also be helpful in causing a DNS cache to accept and use incorrect information.

3.1.2. Flood-Based DoS

Flooding-based DoS attack vectors are particularly effective attacks on the Internet today. They usually entail flooding a large number

of packets towards a target system, with the hopes of either exhausting connection state on the target system, consuming all packet processing capabilities of the target or intermediate systems, or consuming a great deal of bandwidth available to the target system such that they are essentially inaccessible.

Because these attacks require no reply from the target system and require no legitimate transaction state, attackers often attempt to obfuscate the identity of the systems that are generating the attack traffic by spoofing the source IP address of the attacking traffic flows. Because ingress filtering isn't applied ubiquitously on the Internet today, spoof-based flooding attack vectors are typically very difficult to traceback. In particular, there may be one or more attacking sources beyond your network border, and the attacking sources may or may not be legitimate sources, it's difficult to discriminate if the sources are not directly connected to the local routing system.

Common flood-based DoS attack vectors today include SYN floods, ICMP floods, and IP fragmentation attacks. Attackers may use a single legitimate or spoofed fixed attacking source address, although frequently they cycle through large swaths of address space. As a result, mitigating these attacks on the receiving end with source-based policies is extremely difficult.

Furthermore, the motivator for spoof-based DoS attacks may actually be to encourage the target to filter explicitly on a given set of source addresses, or order to disrupt the legitimate owner(s) access to the target system.

3.1.3. Poisoning Attacks

While poison attacks can often be done with single packets, it is also true that a stream of packets can be used to find a window where the target will accept the incorrect information. In general, this can be used to perform broadly the same kinds of poisonings as above, with more versatility.

3.1.4. Spoof-based Worm/Malware Propagation

Self-propagating malware has been observed that spoofs its source address when attempting to propagate to other systems. Presumably, this was done to obfuscate the actual source address of the infected system.

3.1.5. Reflective Attacks

DNS reflective amplification attacks are a particularly potent DoS attack vector on the Internet today. Like other amplification attacks, an attack source generates a packet with a source-spoofed address mapping to that of the target system. The packet, upon receipt by the victim or some intermediate node, typically elicits a large reply message, which is directed to the target of the attack. The amplification factor observed for attacks targeting DNS root and other top level domain name infrastructure in early 2006 was on the order of 76:1. The result is that just 15 attacking sources with 512Kbps of upstream attack bandwidth could generate one Gbps of response attack traffic towards a target system.

Smurf attacks employ a similar reflective amplification attack vector, exploiting traditional default IP subnet directed broadcast address behaviors that would result in all the active hosts on a given subnet responding to (spoofed) ICMP echo request from an attacker, and generating a large amount of ICMP echo response traffic directed towards a target system. They were particularly effective in large campus LAN environments where 50k or more hosts might reside on a single subnet.

3.1.6. Accounting Subversion

If an attacker wishes to distribute content or other material in a manner that employs protocols that require only uni-directional flooding and generate no end-end transactional state, they may desire to spoof the source IP address of that content in order to avoid detection or accounting functions enabled at the IP layer. While this particular attack has not been observed, it is included here to reflect the range of power that spoofed addresses may have even without the ability to receive responses.

3.1.7. Other Blind Spoofing Attacks

Other Blind spoofing attacks might include spoofing in order to exploit source routing or other policy based routing implemented in a network. It may also be possible in some environments to use spoofing techniques to perform blind or non-blind attacks on the routers in a site or in the Internet. There are many techniques to mitigate these attacks, but it is well known that there are vulnerabilities in this area. Among other attacks, if there are multiple routers on-link with hosts, a host may be able to cause problems for the routing system by replaying modified or unmodified routing packets as if they came from another router.

3.2. Non-Blind Attacks

Non-blind attacks often involve mechanisms such as eavesdropping on connection, resetting state so that new connections may be hijacked, and an array of other attack vectors. Perhaps the most common of these attack vectors is known as man in the middle attacks.

3.2.1. Man in the Middle (MITM)

Connection Hijacking is one of the more common man in the middle attack vectors. In order to hijack a connection an attacker usually needs to be in the forwarding path and often times employs TCP RST or other attacks in order to reset a transaction. The attacker may have already compromised a system that's in the forwarding path, or they may wish to insert themselves in the forwarding path.

For example, an attacker with access to a host on LAN segment may wish to redirect all the traffic on the local segment destined for a default gateway address (or all addresses) to itself in order to perform man-in-the-middle attacks. In order to accomplish this the attacker might transmit gratuitous ARP [RFC0826] messages or ARP replies to the Ethernet broadcast address ff:ff:ff:ff:ff:ff, notifying all the hosts on the segment that the IP address(es) of the target(s) now map to it's own MAC address. If the port to which the attacker is connected were to implement policy that binds a single Link Layer and IP address tuple to that port upon initial provisioning, spoofed packets, at the Link Layer and/or Network Layer, would be discarded on ingress.

3.2.2. Third Party Recon

Another example of sighted attack is third party reconnaissance. The use of spoofed addresses, while not necessary for this, can often provide additional information, and helps mask the traceability of the activity. The attack involves sending packets towards a given target system and observing either target or intermediate system responses. For example, if an attacker were to source spoof TCP SYN packets towards a target system from a large set of source addresses, and observe responses from that target system or some intermediate firewall or other middle box, they would be able to identify what IP layer filtering policies may be in place to protect that system.

4. Current Anti-Spoofing Solutions

The first requirement is to eliminate datagrams with spoofed addresses from the Internet. Identifying and dropping datagrams whose source address is incompatible with the Internet topology at

sites where the relationship between the source address and topology can be checked can eliminate such datagrams. For example, Internet devices can confirm that:

- o The IP source address is appropriate for the lower layer address (they both identify the same system)
- o The IP source address is appropriate for the device at the layer 2 switch port (the address was assigned to a, and perhaps the, system that uses that port)
- o The prefix to which the IP source address belongs is appropriate for the part of the network topology from which the IP datagram was received (while the individual system may be unknown, it is reasonable to believe that the system is located in that part of the network).

Filtering points farther away from the source of the datagram can make decreasingly authoritative assertions about the validity of the source address in the datagram. Nonetheless, there is value in dropping traffic that is clearly inappropriate, and in maintaining knowledge of the level of trust one can place in an address.

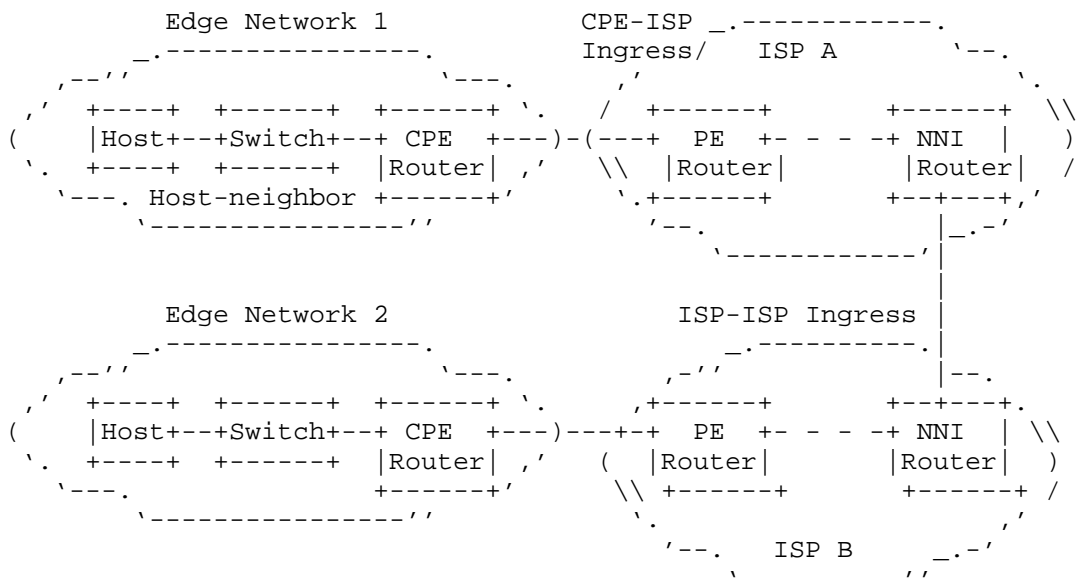


Figure 1: Points where an address can be validated

Figure 1 illustrates five related paths where a source address can be validated:

- o host to switch, including host to host via the switch
- o Host to enterprise CPE Router
- o Enterprise CPE Router to ISP edge PE Router, and the reverse
- o ISP NNI Router to ISP NNI Router

In general, datagrams with spoofed addresses can be detected and discarded by devices that have an authoritative mapping between IP addresses and the network topology. For example, a device that has an authoritative table between Link Layer and IP addresses on a link can discard any datagrams in which the IP address is not associated with the Link Layer address in the datagram. The degree of confidence in the source address depends on where the spoofing detection is performed and the prefix aggregation in place between the spoofing detection and the source of the datagram.

4.1. Host to link layer neighbor via switch

The first point at which a datagram with a spoofed address can be detected is on the link to which the source of the datagram is connected. At this point in the network, the source Link Layer and IP addresses are both available, and can be verified against each other. A datagram in which the IP source address does not match the corresponding Link Layer address can be discarded. Of course, the trust in the filtering depends on the trust in the method through which the mappings are developed. This mechanism can be applied by a first hop router, or switch on the link. The first hop switch has the most precise information for this.

On a truly shared medium link, such as classic Ethernet, the best that can be done is to verify the Link Layer and IP addresses against the mappings. When the link is not shared, such as when the hosts are connected through a switch, the source host can be identified precisely based on the port through which the datagram is received or the MAC address if it is known to the switch. Port identification prevents transmission of malicious datagrams whose Link Layer and IP addresses are both spoofed to mimic another host.

4.2. Upstream routers

Beyond the first hop router, subsequent routers may additionally filter traffic from downstream networks. Because these routers do not have access to the Link Layer address of the device from which the datagram was sent, they are limited to confirming that the source IP address is within a prefix that is appropriate for downstream router from which the datagram was received.

Options include the use of simple access lists or the use of unicast reverse path filtering (uRPF). Access lists are generally appropriate only for the simplest cases, as management can be difficult. Strict Unicast RPF accepts the source address on a datagram if and only if it comes from a direction that would be rational to send a datagram directed to the address, which means that the filter is derived from routing information. These filtering procedures are discussed in more detail in [RFC3704].

4.3. ISP Edge PE Router

An obvious special case of the discussion is with an ISP PE router, where it provides its customer with access. BCP 38 specifically encourages ISPs to use ingress filtering to limit the incidence of spoofed addresses in the network.

The question that the ISP must answer for itself is the degree to which it trusts its downstream network. A contract might be written between an ISP and its customer requiring that the customer apply the procedures of network ingress filtering to the customer's own network, although there's no way upstream networks would be able to verify this.

4.4. ISP NNI Router to ISP NNI Router

The considerations explicitly related to customer networks can also be applied to neighboring ISPs. An interconnection agreement might be written between two companies requiring network ingress filtering policy be implemented on all customers connections. ISPs might, for example, mark datagrams from neighboring ISPs that do not sign such a contract or demonstrably do not behave in accordance with it as 'untrusted'. Alternatively, the ISP might place untrusted prefixes into a separate BGP community and use that to advertise the level of trust to its BGP peers.

In this case, uRPF is less effective as a verification tool, due to asymmetric routing. However, when it can be shown that spoofed addresses are present, the procedure can be applied.

4.5. Spoofing In Local Area Network Segments

On Link Layer segments where multiple hosts reside, or a single MAC address can be associated with a port or interface, if a binding between a hardware address (e.g., MAC address) and corresponding IP address(es) are not provisioned via some means (either manual or dynamic mechanisms), then an attacker may exploit attack vectors that enable MITM or other spoof-based attacks.

4.6. Cable Modem Subscriber Access

Cable Modem Termination Systems (CMTS) employ DOCSIS Media Access Control (MAC) domains, which are similar to Ethernet LAN environments.

4.7. DSL Subscriber Access

While DSL subscriber access can be bridged or routed, as seen by the service provider's device, it is generally the case that the protocols carry enough information to verify which subscriber is sending packets. Thus, for ensuring that one DSL subscriber does not spoof another, enforcement can generally be done at the aggregation router. This is true even when there is a bridged infrastructure among the subscribers, as DSL access generally requires all subscriber traffic to go through the access aggregation router.

If it is desirable to provide spoofing protection among the devices within a residence, that would need to be provided by the CPE device, as the ISPs router does not have enough visibility to do that. It is not clear at this time that this problem is seen as a relevant threat.

4.8. BCP 38

If BCP 38 [RFC2827] is implemented in LAN segments, it is typically done so on subnetwork boundaries and traditionally relates only to Network Layer ingress filtering policies. The result is that hosts within the segment cannot spoof packets from address space outside of the local segment itself, however, they may still spoof packets using sources addresses that exist within the local network segment.

4.9. Unicast RPF

Unicast RPF is a crude mechanism to automate definition of BCP 38 style filters based on routing table information. Its applicability parallels that of BCP 38, although deployment caveats exist, as outlined in [RFC3704].

4.10. Port-based Address Binding

Much of the work SAVI appears to be initially targeting is aimed at minimizing source address spoofing in the LAN. In particular, if mechanisms can be defined to accommodate configuration of port binding information for IP and MAC layer addresses in LAN environments, a large portion of the spoofing threat space in the LAN can be marginalized.

However, establishing this binding is not trivial, and varying across both topologies type and address allocation mechanisms.

4.10.1. Manual Binding

Binding of a single Link Layer and Network Layer address to a port may initially seem trivial. However, two primary areas exist that can complicate such techniques. In particular, these areas involve topologies where more than a single IP layer address may be associated with a MAC address on a given port, or where multiple hosts are connected to a single IP port. Furthermore, if one or more dynamic address allocation mechanisms such as DHCP are employed, then some mechanism must exist to associate those IP layer addresses with the appropriate Link layer ports, as addresses are allocated or reclaimed.

4.10.2. Automated Binding

For IPv4 the primary and very widely used automated binding technique is DHCP based address assignment. Controlling where authoritative information can be sourced, coupled with sniffing and enforcing the assignments is an effective technique.

For IPv6, there are two common automated address binding techniques. While there are many variations and details, for purposes of understanding the threats and basic responses, these are Stateless Address AutoConfiguration (SLAAC) and DHCPv6 based address assignment. In both cases, binding establishment needs to be tied to the state machines for these protocols, and appropriate message sniffing and enforcement. For DHCPv6 based techniques, it is also necessary to use classification techniques to ensure that responses come from authoritative sources.

4.10.3. IEEE 802.1X

IEEE 802.1x is an authentication protocol that permits a network to determine the identity of a system seeking to join it and apply authorization rules to permit or deny the action.

4.11. Cryptographic Techniques

Needless to say, MITM and replay attacks can typically be mitigated with cryptographic techniques. However, many of the applications today either don't or can't employ cryptographic authentication and protection mechanisms. ARP for IPv4 does not use such protection. While SEND provides such protection for IPv6 ND, SEND is not widely used to date. While DNSsec will significantly help protect DNS from spoof based poisoning attacks, it will probably be sufficiently long

for truly widespread use that other protections can be usefully deployed as well.

4.12. Residual Attacks

It should be understood that not all combinations of network, service and enforcement choices will result in a protectable network. For example, if one uses conventional SLAAC, in a switched network, but tries to only provide address enforcement on the routers on the network, then the ability to provide protection is severely limited.

5. Topological Considerations

As noted previously, topological components and address allocation mechanisms have significant implications on what is feasible with regard to Link layer address and IP address port bindings. The following sections discuss some of the various topologies and address allocation mechanisms that proposed SAVI solutions should attempt to address.

5.1. Address Provisioning Mechanisms

In a strictly static environment, configuration management for access filters that map Link Layer and Network Layer addresses on a specific switch port might be a viable option. However, most networks, certainly those that accommodate actual human users, are much more dynamic in nature. As such, mechanisms that provide port-MAC-IP bindings need to accommodate dynamic address allocation schemes enabled by protocols such as DHCP, DHCPv6 for address allocation, and IPv6 Stateless Address Autoconfiguration.

5.2. LAN devices with Multiple Addresses

From a topology considerations perspective, when attempting port-MAC-IP bindings, host connected to switch ports that may have one or more IP addresses, or certainly, devices that forward packets from other network segments, are problematic.

5.2.1. Routers

The most obvious example of devices that are problematic when attempting to implement port-MAC-IP bindings is that of routers. Routers not only originate packets themselves and often have multiple interfaces, but also forward packets from other network segments. As a result, it's difficult for port-MAC-IP binding rules to be established a priori, because it's likely that many addresses and IP subnets should be associated with the port-MAC in question.

5.2.2. NATs

Validating traffic from Prefix-based and multi-address NATs also becomes problematic, for the same reasons as routers. Because they may forward traffic from an array of address, a priori knowledge must exist providing what IPs should be associated with a given port-MAC pair.

5.2.3. Multi-Instance Hosts

Another example that introduces complexities is that of multi-instance hosts attached to a switch port. These are single physical devices, which internally run multiple physical or logical hosts. When the device is a blade server, with internal blades each hosting a machine, there is essentially a physical switch inside the blade server. While tractable, this creates some complexity for determining where enforcement logic can or should live.

Logically distinct hosts such as are provided by many varieties of virtualization logic result in a single physical host, connect to a single port on the Ethernet switch in the topology, actually having multiple internal IP and MAC addresses, and essentially an internal switch. While it may be possible for this internal switch to help control threats among the virtual hosts, or between virtual hosts and other parts of the network, such enforcement cannot be counted on at this time.

5.2.4. Multi-LAN Hosts

Multi-interface hosts, in particular those that are multi-homed and may forward packets from any of a number of source addresses, can be problematic as well. In particular, if a port-MAC-IP binding is made on each of the interfaces, and then either a loopback IP or the address of third interface is used as the source address of a packet forwarded through an interface for which the port-MAC-IP binding doesn't map, the traffic may be discarded. Static configuration of port-MAC-IP bindings may accommodate this scenario, although some a priori knowledge on address assignment and topology is required.

While the use of loopback addressing or sending packets out one interface with the source address from another are rare, they do legitimately occur. Some servers, particularly ones that have underlying virtualization, use loopback techniques for management.

5.2.5. Firewalls

Firewalls that forward packets from other network segments, or serve as a source for locally originated packets, suffer from the same

issues as routers.

5.2.6. Mobile IP

Mobile IP hosts in both IPv4 and IPv6 are proper members of the site where they are currently located. Their care-of-address is a properly assigned address that is on the link they are using. And their packets are sent and received using that address. Thus, they do not introduce any additional complications. (There was at one time consideration of allowing mobile hosts to use their home address when away from home. This was not done, precisely to ensure that mobile hosts comply with source address validity requirements.) Mobile Hosts with multiple physical interfaces fall into the cases above.

Mobile IP home agents are somewhat more interesting. Although they are (typically) fixed devices, they are required to send and receive packets addressed from or to any currently properly registered mobile node. From an analysis point of view, even though the packets that a Home Agent handles are actually addressed to or from the link the HA is on, it is probably best to think of them as routers, with a virtual interface to the actual hosts they are serving.

5.2.7. Other Topologies

Any topology that results in the possibility that a device connected to a switch port may forward packets with more than a single source address for packet which it originated may be problematic. Additionally, address allocation schemas introduce additional considerations when examining a given SAVI solutions space.

5.3. IPv6 Considerations

IPv6 introduces additional capabilities which indirectly complicate the spoofing analysis. IPv6 introduces and recommends the use of stateless address autoconfiguration (often referred to as SLAAC). This allows hosts to determine their IP prefix, select an ID, and then start communicating. While there are many advantages to this, the absence of control interactions complicates the process of behavioral enforcement.

An additional complication is the very large ID space. Again, this 64 bit ID space provided by IPv6 has many advantages. It provides the opportunity for many useful behaviors. However, it also means that in the absence of controls, hosts can mint anonymous addresses as often as they like, modulo the idiosyncrasies of the duplicate address procedure. Like many behaviors, this is a feature for some purposes, and a problem for others. But it does have implications

for switch cost; the switch needs to store more addresses and so needs more memory.

6. Applicability of Anti-Spoofing Solutions

The above sections covered a number of security threats. Not all these threats can be prevented by anti-spoofing techniques. However, all of these threats can be ameliorated to some degree. We can look at three categories of effect we can achieve:

Prevention: Some of the threats described above explicitly require that a host send packets using some other active hosts IP address as a source. Anti-Spoofing measures can prevent these attacks.

Impediment: Many of the attacks above, such as some kinds of DoS attacks, can be conducted more easily if the attacking host can use multiple different IP addresses. Depending upon the kind of anti-spoofing available, the scope of such false addresses, or even their use, may be prevented, hindering the attacker even if the attack is not completely prevented.

Traceability: Even when attacks cannot be prevented, the ability to reliably trace an attack allows appropriate responses, and thereby also creates an environment which discourages attacks instead of encouraging them. Thus, ensuring that even attacks which are not dependent upon spoofing cannot use source address spoofing to hide their origin is extremely important.

For example, sites which deploy BCP 38 cannot be the source of attacks which rely on spoofing the source site from which an attack was launched. Wide deployment of BCP 38 would also simplify the task of tracking attacks back to their actual origin.

6.1. Analysis of Host Granularity Anti-Spoofing

Applying anti-spoofing techniques at the host level enables a site to achieve several valuable objectives. While it is likely the case that for many site topologies and policies, full source spoofing protection is not possible, it is also true that for many sites there are steps that can be taken that provide benefit.

One important class of benefit is masquerade prevention. Security threats involving one machine masquerading as another, for example in order to hijack an apparently secure session, can occur within a site with significant impact. Having mechanisms such that host facing devices prevent this is a significant intra-site security improvement. Given that security experts report that most security

breaches are internal, this can be valuable. One example of this is that such techniques should mitigate internal attacks on the site routing system.

A second class of benefit is related to the traceability described above. When a security incident is detected, either within a site, or externally (and traced to the site) it can be critical to determine what the actual source of the incident was. If address usage can be tied to the kinds of anchors described earlier, then it is possible to respond to security incidents.

In addition to these local observable benefits, there can be more global benefits. For example, if address usage is tied to anchors, it may be possible to prevent or control the use of large numbers of anonymous IPv6 addresses for attacks, or at least to track even those attacks back to their source.

7. Existing Techniques for IP Source Address Validation

Existing techniques for IP source address validation are insufficient. While each technique has its own shortcomings, the following list of general categories of reasons include some of the deficiencies of all existing technique:

False negatives: Techniques may yield false negatives, thus enabling an attacker to select an IP source address that is spoofed, but still passes IP source address validation.

False positives: Techniques may yield false positives, thereby causing interruption or denial of service to hosts that use legitimate IP source addresses.

Non-trivial configuration: Requirements for non-trivial configuration imply expenditures and pose a risk for misconfiguration, which may again lead to false positives or false negatives. Both may discourage operators from employing a given technique.

Proprietary: Procurement policies of organizations oftentimes require that devices purchased use techniques that are standardized rather than proprietary. Such policies, for good or ill, hinder or prevent the deployment of proprietary techniques.

The only standardized technique for IP source address validation is ingress filtering [RFC2827]. This calls for routers to check whether the prefix of a to-be-forwarded packet's IP source address is amongst a list of prefixes considered legitimate for the interface through

which the packet arrives. Packets that fail this check are discarded.

Ingress filtering may yield false negatives in a deterministic manner. Packets with a legitimate IP source address prefix, but a spoofed interface identifier, pass ingress filtering checks. Also, packets with an illegitimate IP source address prefix pass the checks as long as the prefix is from the list of prefixes considered legitimate, if more than one prefix is considered as legitimate on the ingress interface..

Ingress filtering implementations that require manual establishment of the list of legitimate prefixes cause considerable configuration overhead. Unicast Reverse Path Forwarding (uRPF) mitigates this issue by automatically deriving the list of legitimate prefixes from a router's forwarding table: A to-be-forwarded packet's IP source address prefix is considered legitimate if the packet is coming through an interface via which return traffic would be routed. On the other hand, Unicast Reverse Path Forwarding may yield false positives, in particular for hosts and networks with multiple, topologically separate Internet attachments [RFC3704].

Consequently, there is a need for an IP source address validation technique that avoids false negatives and false positives, that can be set up with no or only trivial configuration, and that has been standardized. The development of such a technique is the goal of the proposed Source Address Validation Improvements (SAVI) working group in the Internet Engineering Task Force.

8. Deployment Considerations

From a global Internet perspective, deployment of anti-spoofing techniques tends to suffer from a "tragedy of the commons" situation. That is, there is a general consensus that everyone should implement anti-spoofing measures, yet individual organizations don't want to bear the cost of deployment themselves, often because demonstrating direct tangible return on investment is not possible. Upon analysis, it often seems apparent that local implementation of anti-spoofing measures is of more benefit to the "greater Internet" than the local network domain itself. A similar situation occurs with de-aggregation of Internet routing information for multi-homing and traffic engineering purposes, as well as the lack of explicit inter-domain routing prefix filters on the Internet.

Until network operators begin to accept that their local design choices have global implications, and act upon this responsibility, the problem is not going to go away.

Ideally, with additional work in the areas of SAVI to ease deployment and management burdens, the deployment cost to operators will decrease and more wide-scale deployment will continue. Furthermore, application of SAVI-like techniques provides more obvious benefits to network administrators that are concerned with MITM and similar attacks.

As mentioned earlier, there are local security benefits to the deployment of SAVI security mechanisms. This may help motivate the deployment of tools with widespread benefit.

9. IANA Considerations

This memo asks the IANA for no new parameters.

Note to RFC Editor: This section will have served its purpose if it correctly tells IANA that no new assignments or registries are required, or if those assignments or registries are created during the RFC publication process. From the authors' perspective, it may therefore be removed upon publication as an RFC at the RFC Editor's discretion.

10. Security Considerations

This document provides limited discussion of some security threats source address validation improvements will help to mitigate. It is not meant to be all-inclusive, either from a threat analysis perspective, or from the source address verification application side.

It is seductive to think of SAVI solutions as providing the ability to use this technology to trace a datagram to the person, or at least end system, that originated it. For several reasons, the technology can be used to derive circumstantial evidence, but does not actually solve that problem.

In the Internet Layer, the source address of a datagram should be the address of the system that originated it and to which any reply is expected to come. But systems fall into several broad categories. Many are single user systems, such as laptops and PDAs. Multi-user systems are commonly used in industry, and a wide variety of middleware systems and application servers have no user at all, but by design relay messages or perform services on behalf of users of other systems (e.g., SMTP and peer-to-peer file sharing).

Until every Internet-connected network implements source address

validation at the ultimate network ingress, and assurances exist that intermediate devices are to never modify datagram source addresses, source addresses must not be used as an authentication mechanism. The only technique to unquestionably verify source addresses of a received datagram are cryptographic authentication mechanisms such as IPsec.

11. Acknowledgements

A portion of the primer text in this document came directly from [I-D.baker-sava-operational], authored by Fred Baker and Ralph Droms. Many thanks to Christian Vogt and Suresh Bhogavilli for contributing text and a careful review of this document.

12. References

12.1. Normative References

- [RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.
- [RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

12.2. Informative References

- [I-D.baker-sava-operational]
Baker, F. and R. Droms, "IPv4/IPv6 Source Address Verification", draft-baker-sava-operational-00 (work in progress), June 2007.
- [RFC0826] Plummer, D., "Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware", STD 37, RFC 826, November 1982.
- [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.
- [RFC3704] Baker, F. and P. Savola, "Ingress Filtering for Multihomed Networks", BCP 84, RFC 3704, March 2004.

Authors' Addresses

Danny McPherson
VeriSign, Inc.

Email: dmcpherson@verisign.com

Fred Baker
Cisco Systems

Email: fred@cisco.com

Joel M. Halpern
Ericsson

Email: joel.halpern@ericsson.com

SAVI
Internet Draft
Intended status: Standard Tracks
Expires: April 15, 2011

T. Lin, Ed.
Hangzhou H3C Tech. Co., Ltd.

October 15, 2010

Roaming over SAVI devices
draft-lin-savi-roaming-nd-00

Abstract

This document specifies the procedure for roaming over devices that implemented SAVI (Source Address Validation Improvements) device. The binding entry creating by NDP snooping can be synchronized by some mechanism.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, except to publish it as an RFC and to translate it into languages other than English.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before October 10, 2010. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/ietf/lid-abstracts.txt>

The list of Internet-Draft Shadow Directories can be accessed at <http://www.ietf.org/shadow.html>

This Internet-Draft will expire on April 15, 2011.

Copyright Notice

Copyright (c) 2010 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

Abstract	1
Copyright Notice	2
1. Introduction	3
2. Conventions used in this document	3
3. Terminology	3
4. Problem Statement	3
5. Solution for roaming over devices	5
5.1. Binding entry establishing	5
5.2. Binding entry moving	5
5.2. Source address selection	6
6. Security Considerations	6
7. IANA Considerations	6
8. References	6
8.1. Normative References	6
8.2. Informative References	6
9. Acknowledgments	7
Author's Address	7

1. Introduction

This document specifies the procedure for roaming over devices that implemented SAVI (Source Address Validation Improvements) device. The binding entry creating by NDP snooping can be synchronized by some mechanism.

There is a mechanism which designed to provide a host level source IP address validation granularity([I-D.bi-savi-stateless]), which includes NDP/ARP snooping to set up bindings between stateless IP addresses and corresponding anchors. The bindings can be used to validate the source address in the packets. This mechanism is deployed on access device(we mainly talk about access switch), named SAVI devices.

The NDP/ARP snooping mechanism in SAVI devices snoop the NDP/ARP packets to establish the corresponding binding entry of the host, which includes IP address, MAC address, VLAN, port, etc.

The binding entry can filter packets with forged IP addresses, including NDP/ARP packet and common data packet. [I-D.ietf-savi-framework-00] describes this filtering procedure.

This mechanism can nicely filter packets on a SAVI device in one access device LAN. If there is two or more SAVI devices, if the host roams from one SAVI device to another SAVI device, the following issues must be considered: How to aging the binding entry on the first SAVI device, and how to defend the forged host while the real host can roaming correctly.

2. Conventions used in this document

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. Terminology

Uplink port: The up link port of the access switch. SAVI device is a access device, so the snooping and filtering should aim at the port connected to user. NDP/ARP snooping can omit the learning procedure of binding entry.

4. Problem Statement

As shown in figure 1, two switches are connected one aggregation device (AGG). If one host connects to SWB(Switch B), the

corresponding binding entry will be established in SWB. When this host roams to SWA(Switch A), all involved ports's link state will changed, so the corresponding binding entry can established in SWA. But SWB's entry won't be removed immediately.

The following is the processes:

- 1) When the host is removed from SWB, SWB will keep the binding entry for a while.
- 2) When the host is connected SWA, the link state of the host's net card interface will turn up, so the host will send DAD NS.
- 3) When SWA receive this packet, it find this is a new host connected to it. SWA will establish a new binding entry, and forward this packet.
- 4) When SWB receives this packet, because the received port is uplink port, so it only forward it.
- 5) Because there isn't any host connect the SWB's port which the host connected formerly, so this packet will be discarded finally.

Now, there are two binding entries of one host in two SAVI devices, this is a very confusable condition.

From the above, the key element of the roaming over SAVI devices is uplink port. If the binding entries can be established in this port, the roaming can successfully process, but all the SAVI devices will have all the host corresponding binding entries. These is a huge resource waste.

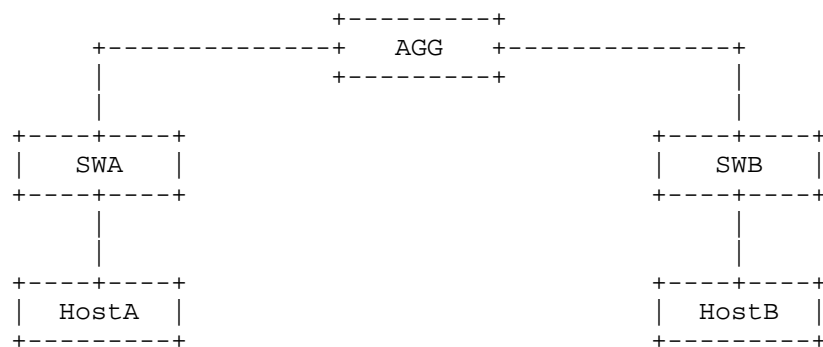


Figure 1 Roaming over devices

5. Solution for roaming over devices

Considering the uplink port snooping, the SAVI device can know the host connected others SAVI device, this is a key to promote the roaming.

Based on the uplink port, the following procedure is proposed to deal with the issue in section 4, the host can roam over SAVI devices.

5.1. Binding entry establishing

When the host connects to SWB, the normal binding entry establishing procedure([I-D.bi-savi-stateless]) is process in SWB. The DAD NS also forward to SWA transit AGG, SWA's uplink port will received this packet.

But SWA must ignore this packet, not to established any binding entry to avoid the huge binding table.

5.2. Binding entry moving

When the host is roaming to SWA, the following processes happen:

- 1) When the host is removed from SWB, SWB will keep the binding entry for a while.
- 2) When the host is connected SWA, the link state of the host's net card interface will turn up, so the host will send DAD NS.
- 3) When SWA receive this packet, it find this is a new host connected to it. SWA will establish a new binding entry, and forward this packet.
- 4) When SWB receives this packet, it will analysis this packet as normal NDP snooping, so as to find there is a binding entry correspond to that host, only the input port is uplink port.
- 5) According this binding entry, SWB sending a normal NS through the original port the host connected to probe if or not the host is still connected here. Because the host is removed from SWB, so SWB can't received any response.
- 6) Then SWB sends a normal NS through the uplink port to probe if there is a new host connected other switch. SWA can receive this packet and forward it to the host, and the host will respond to SWB.

- 7) When SWB receives this responding packet, it will remove that corresponding binding entry. Because this packet is received in uplink port from other switch, SWB can know that other switch have established a binding entry of this packet.

Now the host have successfully roaming from SWB to SWA, and SWB have not the host's corresponding binding entry.

5.3. Source address selection

In the process 5) and 6), SWB needs to send a NS packet which source IP address must be SWB's IP address. Otherwise, the response packet may not be received by SWB, and this mechanism will not take effect.

6. Security Considerations

There is no security consideration currently.

7. IANA Considerations

There is no IANA consideration currently.

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.

8.2. Informative References

[RFC2131] R. Droms, "Dynamic Host Configuration Protocol", RFC2131, March 1997.

[RFC3307] B. Haberman, "Allocation Guidelines for IPv6 Multicast Addresses", RFC3307, August 2002.

[RFC3315] R. Droms, Ed. "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC3315, July 2003.

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

[RFC4862] Thomson, S., Narten, T. and Jinmei, T., "IPv6 Stateless Autoconfiguration", RFC4862, October, 2007.

[RFC5227] S. Cheshire, "IPv4 Address Conflict Detection", RFC5227, July 2008.

[I-D.ietf-savi-framework-00]

J. Wu, J. Bi, M. Bagnulo, F. Baker, C. Vogt, "Source Address Validation Improvement Protocol Framework", I-D.ietf-savi-framework-00(work in progress), October 2010.

[I-D.bi-savi-stateless]

J. Bi, G. Yao, J. Wu, and F. Baker, "SAVI Solution for Stateless Address", I-D.bi-savi-stateless-00 (work in progress), April 2010.

9. Acknowledgments

Thanks to Zhenglin Qi, Daogui Liu, Cong Xue, Lei Cao, Li Hou, Jun Bi, and Guang Yao for their valuable contributions.

Authors' Addresses

Tao Lin
Hangzhou H3C Tech. Co., Ltd.
Beijing R&D Center of H3C, Digital Technology Plaza
NO. 9 Shangdi 9th Street, Haidian District
Beijing 100085
China

Phone: +86 010 82774704
EMail: lintaog@gmail.com

