SAVI
Internet Draft
Intended status: Standard Tracks
Expires: January 2012

J. Bi, J. Wu, G. Yao
Tsinghua Univ.
F. Baker
Cisco Systems
July 7, 2011

SAVI Solution for DHCP
draft-ietf-savi-dhcp-10.txt

Status of this Memo

Copyright Notice

Abstract

This document specifies the procedure for creating bindings between a DHCPv4 [RFC2131]/DHCPv6 [RFC3315] assigned source IP address and a binding anchor [I-D.ietf-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

1. Introduction

This document describes the procedure for creating bindings between DHCP addresses and binding anchor on SAVI device [I-D.ietf-savi-framework]. The removal and restoration of the bindings are also specified in this document.

Bindings can be used to filter or identify packets with forged source IP address. Section 9 suggests usage of these bindings for common practice.

The mechanism specified in this document is designed to provide a binding anchor granularity validation, 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 IP addresses assigned by DHCP and corresponding binding anchors. The binding process is inspired by the work of IP Source Guard [IP Source Guard].

This solution is designed for stateful DHCP scenario [RFC2131], [RFC3315]. In stateless DHCP scenarios [RFC3736], a node must have obtained its IPv6 addresses through some other mechanisms and so the address of the client SHOULD be bound based on other SAVI solutions

This solution is primarily designed for a pure DHCP scenario in which only DHCP address is legitimate global address. How to use this mechanism in multiple address assignments scenario is discussed in [I-D.ietf-savi-mix].

2. Conventions used in this document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Terminology

Lease time: Lease time in IPv4 [RFC2131] and valid lifetime in IPv6 [RFC3315]

4. SAVI-DHCP Scenario

Figure 1 shows the main elements in a DHCP 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. Multiple SAVI devices and non-SAVI devices can co-exist on link. A SAVI device can be connected to DHCP client, DHCP relay (even DHCP server), SAVI device and non-SAVI device.

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

Note that in IPv6 environment, DHCP procedure cannot be performed on an interface without a link-local address pre-assigned. Thus, to make this solution work, a SAVI solution for link-local address MUST be enabled. Knowing that (1) in IPv6 environment, DHCPv6 exchanges mainly use link local addresses as source IP address and (2) by default in DHCP SAVI, a packet with an unbound source IP address is dropped, a SAVI solution for link-local addresses, e.g. [I-D.ietf-savi-fcfs], MUST be deployed in addition to DHCP SAVI.

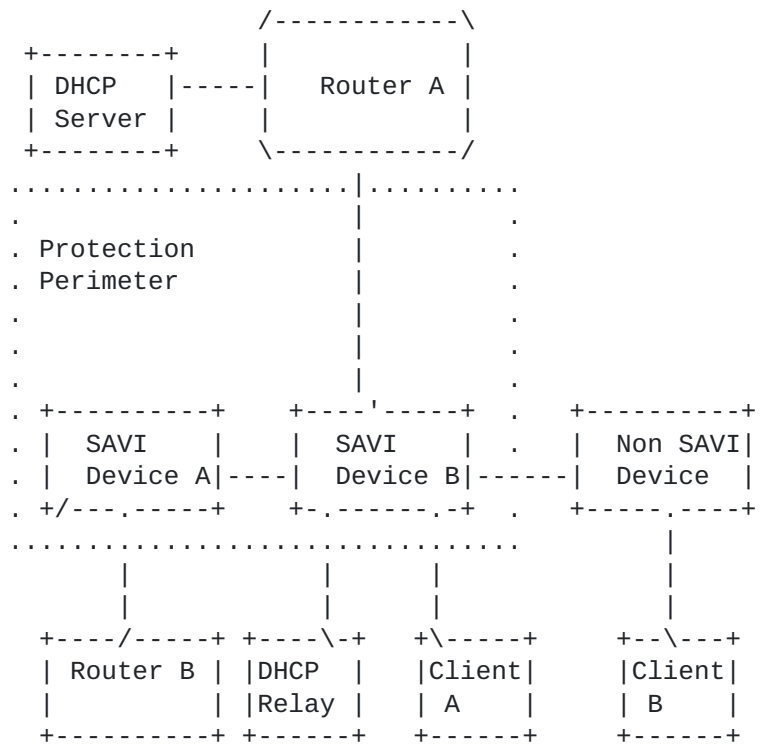

Figure 1 DHCP Scenario

5. Data Structures

This section describes the 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.

Besides the two main data structures, a mapping table from link layer address to binding anchor may be required, as described in section 7.4.1.

5.1. Control Plane Data Structure: Binding State Table

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 TID field recording transaction ID of DHCP message. The

lifetime field is used to help remove expired bindings. The state field is used to identify state. The TID field is used to keep the Transaction ID (TID) [RFC2131][RFC3315] in DHCP request. The TID field can be cleared after the state is changed to BOUND.

Anchor	Address	State	Lifetime	TID
A	IP_1	BOUND	65535	TID 1
A	IP_2	BOUND	10000	TID 2
B	IP_3	INIT_BIND	1	TID_3

Figure 2 Instance of BST

5.2. Data Plane Data Structure: Filtering Table

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. This table SHOULD be updated by other SAVI mechanisms [I-D.ietf-savi-mix] when DHCP SAVI is deployed in IPv6 environment, as explained in section 4.

Anchor	Address
A	IP_1
A	IP_2

Figure 3 Instance of FT

5.3. Mapping Table from Link Layer Address to Binding Anchor

As described in section 7.4.1, whenever binding anchor must be recovered from DCHP Reply, such a mapping table is required. This table maps link layer address to binding anchor, so as that the SAVI device can determine on which binding anchor to set up a binding only
based on a DHCP Reply message.

Such a table can already exist on SAVI device. For example, if the binding anchor is switch port, the mapping table from MAC address to switch port is required on switch for switching frames. We don't

require SAVI device to set up a different mapping table from the existing one. Instead, SAVI device MUST only use the existing one. If there is not such a mapping table yet, SAVI device MUST NOT set up binding based on EVE_DHCP_REPLY_NULL (cf. 7.4.1).

The set up and update of this table is out of the scope of this document.

6. Binding Anchor Attributes

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

Attribute of each binding anchor is configurable. By default, binding anchor has no attribute. A binding anchor MAY be configured to have one or more compatible attributes.

6.1. No Attribute

Before configuration, by default, a binding anchor has no attribute. To filter bogus DHCP server message by default, server type DHCP message from binding anchor with no attribute MUST be dropped. However, to avoid discarding legitimate traffic, other packets SHOULD NOT be dropped before any binding is setup on such binding anchor. Generally, each binding anchor is configured to have one or more attributes after configuration. However, a binding anchor MAY always have no attribute if it is connected to another link. For example, in Figure 1, on the binding anchor of SAVI Device A connected to Router B, it is unreasonable either to set up bindings for host behind Router B, or filter out traffic from Router B, or allow bogus DHCP message from Router B. Thus, no attribute should be configured on this binding anchor.

6.2. SAVI-Validation Attribute

SAVI-Validation attribute is used on binding anchor on which the source address of data packet and DHCP message is to be validated. The filtering process on binding anchor with such attribute is described in section 9. In Figure 1, the binding anchor between SAVI Device B and Client A, and the binding anchor between SAVI Device B and Non SAVI Device should be configured to have this attribute.

6.3. 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. In Figure 1,

the binding anchor between SAVI Device B and DHCP Relay, and the binding anchor between SAVI Device B and Router A, should be configured to have this attribute.

6.4. SAVI-SAVI Attribute

This attribute is used on binding anchor from which the data traffic is not to be checked. Binding will not be set up on binding anchor with this attribute. All data packets will be allowed directly. In Figure 1, the binding anchor between SAVI Device A and SAVI Device B should be configured with this attribute.

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 [I-D.ietf-savi-framework]. Since no binding entry is needed on such binding anchor, the resource requirement can be reduced greatly.

Though there is no factual difference in packet process between a binding anchor with no attribute and a binding anchor only with SAVI-
SAVI attribute, their connotations are different. SAVI-SAVI attribute
is configured on binding anchor between SAVI devices on the same link
inside the protection perimeter. But only when a binding anchor is on
the protection perimeter and connected to another link, it can have no attribute after configuration.

This attribute is mutually exclusive with SAVI-Validation.

6.5. SAVI-BindRecovery Attribute

This attribute is used on binding anchor that requires data-triggered
binding recovery described in section 8.1. It can be configured on any binding anchor with SAVI-Validation attribute, especially, the binding anchor not directly attached by client. In Figure 1, it is suggested to configure this attribute on binding anchor between SAVI Device B and Non SAVI Device.

This attribute is mutually exclusive with SAVI-SAVI.

7. Binding Set Up

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

7.1. Rationale

The rationale of this mechanism is that if a node attached to a binding anchor is legitimate to use a DHCP address, the DHCP procedure which assigns the address to the node must have been performed on the same binding anchor. This basis stands when the link layer routing is stable. However, layer-2 mobility and unstable link layer routing may result in that data packet is received from a different binding anchor. Infrequent link layer path change can be handled (but not perfectly) by the mechanism described in section 8. Section 13.3 discusses the situation that link layer routing is naturedly unstable. A solution for this issue is outside the scope of this document.

7.2. Binding States Description

This section describes the binding states of this mechanism.

NO_BIND The state before a binding has been set up.

INIT_BIND 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.

7.3. Events

7.3.1. Timer Expiration Event

EVE_ENTRY_EXPIRE: The lifetime of an entry expires

7.3.2. Control Message Arriving Events

Only if a control message can pass the check in section 9.2, the corresponding event is a valid event.

EVE_DHCP_REQUEST: A DHCP Request message is received from a binding anchor with SAVI-Validation attribute, and the binding entry limit (cf. section 13.2) 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.

EVE_DHCP_REPLY_NULL: A DHCPv4 Acknowledgement or DHCPv6 Reply message is received from a binding anchor with SAVI-DHCP-Trust attribute, and there is no entry in state INIT_BIND contains the same TID as the message.

EVE_DHCP_DECLINE: A DHCP Decline message is received from a binding anchor with SAVI-Validation attribute.

EVE_DHCP_RELEASE: A DHCP Release message is received from a binding anchor with SAVI-Validation attribute.

EVE_LEASEQUERY_REPLY: A successful DHCP LEASEQUERY_REPLY is received from a binding anchor with SAVI-DHCP-Trust attribute.

7.4. Process of DHCP Packet Snooping

7.4.1. From NO_BIND to Other States

7.4.1.1. Trigger Event

EVE_DHCP_REQUEST, EVE_DHCP_CONFIRM, EVE_DHCP_OPTION_RC, EVE_DHCP_REPLY_NULL.

7.4.1.2. 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 INIT_BIND.
The lifetime of this entry MUST set to be MAX_DHCP_RESPONSE_TIME.
The TID field of the request packet MUST be recorded in the entry.

Anchor	Address	State	Lifetime	TID
A		INIT_BIND	MAX_DHCP_RESPONSE_TIME	TID

Figure 4 Binding entry in BST on client triggered initialization

The TID is stored because it will be used in a future step (cf. section 7.4.2.2) to correctly associate the assigned address to the binding anchor.

If the triggering event is EVE_DHCP_CONFIRM:

Besides forwarding the message and generating corresponding entry, the address to be confirmed MUST be recorded in the entry. Because no lease time will be contained in the REPLY from DHCP server, the SAVI device MUST send a LEASEQUERY [RFC5007] message querying by IP address to All_DHCP_Relay_Agents_and_Servers multicast address [RFC3315] or a configured server address.

Anchor	Address	State	Lifetime	TID
A	Addr	INIT_BIND	MAX_DHCP_RESPONSE_TIME	TID

Figure 5 Binding entry in BST on Confirm triggered initialization

If the triggering event is EVE_DHCP_REPLY_NULL:

If the binding anchor is not link layer address and there is not a mapping table from link layer address to binding anchor, the message SHOULD be discarded.

Else:

The SAVI device MUST deliver the message to the destination.

The SAVI device MUST generate as many new entries in BST and FT as the number of IADDR found in the message. If the binding anchor type inside the BST is not link layer address, the binding anchor for an entry is recovered from the mapping table from link layer address to binding anchor (cf. section 5.3) based on the destination link layer address inside the DHCP message.

The states of the corresponding entries are set to be BOUND. The lifetime of the entries MUST be set to be the lease time.

The binding entry limit can be exceeded when setting up bindings for all addresses in a REPLY message. If there is enough binding entry resource, corresponding new entries MUST be generated even the binding number limit is exceeded. In case that there is not enough resource left, as many as possible entries SHOULD be set up.

If the binding anchor is switch port, there can be vulnerability in this process which is discussed in section 13.1. Similar problem can happen with other binding anchors.

Anchor	Address	State	Lifetime	TID
A	Addr1	BOUND	Lease time 1	TID
A	Addr2	BOUND	Lease time 2	TID

Figure 6 Binding entry in BST on Reply triggered initialization

Anchor	Address
A	Addr1
A	Addr2

Figure 7 Binding entry in FT on Reply triggered initialization

7.4.2. From INIT_BIND to Other States

7.4.2.1. Trigger Event

EVE_DHCP_REPLY, EVE_ENTRY_EXPIRE, EVE_LEASEQUERY_REPLY.

7.4.2.2. Following Actions

If the trigger event is EVE_DHCP_REPLY:

The SAVI device MUST deliver the message to the destination.

If the Address field is null, the lease time in Reply message MUST be recorded in the entry with matched TID. The state of the entry is changed to be BOUND. If more than one IADDR is found in the message, if there is enough binding entry resource, corresponding new entries MUST be generated even the binding number limit is

exceeded. In case that there is not enough resource left, as many as possible entries SHOULD be set up.

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, set the Lifetime of corresponding entry to MAX_LEASEQUERY_DELAY. The state of the entry is changed to be BOUND.

Anchor	Address	State	Lifetime	TID
A	Addr	BOUND	Lease time	TID

Figure 8 From INIT_BIND 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.

If the trigger event is EVE_LEASEQUERY_REPLY:

The Lifetime field of entry with corresponding IP address MUST be set to the lease time in the LEASEQUERY_REPLY. The state of the entry is changed to BOUND.

7.4.3. From BOUND to Other States

7.4.3.1. Trigger Event

EVE_ENTRY_EXPIRE, EVE_DHCP_RELEASE, EVE_DHCP_DECLINE, EVE_DHCP_REPLY_RENEW.

7.4.3.2. 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.

7.5. State Machine of DHCP Snooping

The main state transits are listed as follows.

State	Event	Action	Next State
NO_BIND	REQ/RC	Generate entry	INIT_BIND
NO_BIND	CFM	Generate entry and send Leasequery	INIT_BIND
*NO_BIND	RPL_NULL	Generate entry with lease	BOUND
INIT_BIND	RPL	Record lease time/set LQ_DLY	BOUND
INIT_BIND	Timeout	Remove entry	NO_BIND
INIT_BIND	LQR	Record lease time	BOUND
BOUND	RLS/DCL	Remove entry	NO_BIND
BOUND	Timeout	Remove entry	NO_BIND
BOUND	RNW	Set new lifetime	BOUND

*: optional but NOT RECOMMENDED.

REQ: EVE_DHCP_REQUEST

CFM: EVE_DHCP_CONFIRM

RC: EVE_DHCP_OPTION_RC

RPL: EVE_DHCP REPLY

RPL_NULL: EVE_DHCP REPLY_NULL

DCL: DHCP DECLINE

RLS: DHCP RELEASE

RNW: EVE_DHCP_RPL_RENEW

LQR: EVE_LEASEQUERY_REPLY

Timeout: EVE_ENTRY_EXPIRE

LQ_DLY: MAX_LEASEQUERY_DELAY

8. Supplemental Binding Process

Supplemental binding process is designed to cover scenarios where a packet is sent by node but no previous DHCP exchanges have occurred to update correctly the SAVI device's BST. A typical situation is when the link topology changes after the binding has been set up, and then the node will send packet through a different port than the bound port. Another scenario is that a node moves on the local link without re-configuration process.

Supplemental binding process is designed to avoid blocking permanently legitimate traffic. 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, as explained in the following section.

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 an implementation is directly attached to host, change in link topology will not affect the bindings, and host will always start re-configuration process after interface is re-connected. Thus, there is no need to use additional process to recovery bindings. If the mechanism is not implemented and managed hosts are not directly attached, permanent legitimate traffic blocking can happen until the node is reconfigured.

8.1. Binding Recovery Process

If a binding anchor is set to have SAVI-BindRecovery attribute, packet without matched binding can trigger the SAVI device to check if the source address can be used by corresponding node:

1. Check if the address has a local conflict through:

   IPv4 address: performing Address Resolution Protocol (ARP) [RFC826] or Address Conflict Detection [RFC5227] twice on the address;

   IPv6 address: performing Duplicate Address Detection (DAD) [RFC4862] twice on the address.

   If duplicate detection fails, the packet MUST be discarded. Otherwise, go to the next step.

2.

IPv4 address:

Send a DHCPLEASEQUERY [RFC4388] message querying by IP address to all DHCPv4 servers with IP Address Lease Time option (option 51). The server addresses can be found through DHCPv4 Discovery or from configuration. If no DHCPLEASEACTIVE message with IP Address Lease Time option is received, discard the packet; otherwise generate a new binding entry for the address, with the life time set to the value encoded in IP Address Lease Time option of the DHCPLEASEACTIVE message.

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, with the lifetime set to the valid lifetime extracted from OPTION_CLIENT_DATA option in the LEASEQUERY-REPLY message. 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.

In case that SAVI device is pure layer-2 device without IP address, it is impossible to perform DHCP LEASEQUERY. Besides, this process may fail if any DHCP server doesn't support DHCP LEASEQUERY.

The security issues about this process is discussed is section 13.5.

9. Filtering Specification

This section specifies how to use bindings to filter packets.

Filtering policies are different for data packet and control packet. DHCP and NDP (Neighbor Discovery Protocol) [RFC4861] messages that may cause state transit are classified into control packet. Neighbor Advertisement (NA) 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.

9.1. Data Packet Filtering

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

If the source of a packet is correctly associated with its binding anchor in the FT, this packet MUST be forwarded else the packet MUST be discarded. The SAVI device SHOULD record any violation.

9.2. Control Packet Filtering

For binding anchors with SAVI-Validation attribute:

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

Discard DHCPv6 Request whose source is not bound with the corresponding binding anchor in FT. Discard 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 FCFS SLAAC solution [I-D.ietf-savi-fcfs] or other solutions.

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

Discard IPv6 Neighbor Solicitation (NS) and IPv4 gratuitous ARP
whose source is not an address bound with the corresponding binding
anchor.

Discard 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.

The SAVI device SHOULD record any violation of the previous rules.

10. State Restoration

If a SAVI device reboots accidentally or designedly, the states kept in volatile memory will get lost. This may cause legitimate traffic from hosts indirectly attached to the SAVI device to be blocked until a binding recovery. Purely using the Binding Recovery Process is expansive and results delay to recover a large number of bindings. Thus, recovery from non-volatile storage, as specified below, is recommended.

If this function is supported by hardware, binding entries MUST be saved into non-volatile storage whenever a new binding entry changes to BOUND state or a binding with BOUND state is removed.

Immediately after reboot, the SAVI device MUST restore binding states from the non-volatile storage. The system time of save process MUST be stored. After rebooting, the SAVI device MUST check whether each entry has been obsolete through comparing the saved lifetime and the difference between current system time and saved system time.

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, set a timer with OFFLINK_DELAY. Until the timer becomes zero, the bindings with the binding anchor SHOULD be kept. As an exception, to handle movement, if receiving DAD Neighbor Solicitation/Gratuitous ARP request targeting at the address during OFFLINK_DELAY, the entry MAY be removed.

If the binding anchor turns on-link during OFFLINK_DELAY, turn off the timer and keep corresponding bindings.

12. Constants

MAX_DHCP_RESPONSE_TIME	120s
BIND_RECOVERY_INTERVAL	60s and configurable
MAX_LEASEQUERY_DELAY	10s
OFFLINK_DELAY	2s

13. Security Considerations

13.1. Security Problem about Binding Triggered by EVE_DHCP_REPLY_NULL

When the binding anchor is switch port, binding based on EVE_DHCP_REPLY_NULL can result in security threats. The assigned address could be bound to a wrong switch port if an attack can poison the table mapping a link layer address to switch port (cf. section 5.3).

For example, host A requests address from port 1. When SAVI switch receives a DHCP REPLY with assigned address IP_A and destination link layer address MAC_A, it will check its MAC/port table to find the right binding port. But MAC/port table might be polluted by an attacker host B attached to port 2. Then SAVI switch will find the MAC_A is at port 2 from the polluted MAC/port table and it will result in a wrong binding which binds IP_A and port 2.

If one of the following conditions is satisfied, the security can be ensured.

1. DHCP Option 82 is used to keep binding anchor in DHCP Request and Reply.

   DHCP Option 82 can be used to keep the circuit information of the client and returned by the DHCP server. Thus the binding anchor can be determined from the circuit information in the Option. It can be used whenever an implementation doesn't want to create an entry on the DHCP Request message.

2. Unspoofable MAC is used as binding anchor (802.11i, 802.1ae/af).

3. The mapping table from MAC to binding anchor is secure.

   If the binding anchor is a link layer address, or there are mechanisms preventing the corruption of the table mapping the link layer to a switch port, mapping link layer address to binding anchor may be considered as secure.

It is NOT RECOMMENDED to initialize a binding based on DHCP Reply, unless a mechanism protecting the mapping table from corruption is also implemented.

Similar problem may happen with binding anchors not based on link layer addresses.

13.2. 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.

13.3. Risk from Link Layer Routing Dynamic

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, the link layer routing between host and SAVI device can be inconsistent from time to time. 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.

13.4. Duplicate Bindings of Same Address

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 ink, and a node may have multiple binding anchors. However, the traceability of address is reduced.

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 SAVI device. The nodes sharing the same former binding anchor of the moving node have the ability to use its address.

13.5. Security Problems about Binding Recovery Process

The Binding Recovery Process (cf. section 8.1) 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-triggered recovery processes on one binding anchor MUST have a

minimum interval time BIND_RECOVERY_INTERVAL. This constant SHOULD be configured prudently to avoid Denial of Service attacks.

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 detection probes.

13.6. Residual Threats

As described in [I-D.ietf-savi-framework], this solution cannot strictly prevent spoofing. There are two scenarios in which spoofing can still happen:

1. The binding anchor is spoofable

If the binding anchor is spoofable, e.g., plain MAC address, an attacker can use forged binding anchor to send packet which will not be regarded as spoofing by SAVI device.

Indeed, using binding anchor that can be easily spoofed is dangerous.
An attacker can use the binding anchor of another host to perform a lot of DHCP procedures, and the SAVI device will refuse to set up new binding for the host whenever the binding number limitation has been reached. Thus, it is RECOMMENDED to use strong enough binding anchor, e.g., switch port, secure association in 802.11ae/af and 802.11i.

2. The binding anchor is shared by more than one host

If the binding anchor is shared by more than one host, they can spoof the addresses of each other. For example, a number of hosts can attach to the same switch port of a SAVI device through a hub. The SAVI device cannot distinguish packets from different hosts and thus the spoofing between them will not be detected. This problem can be solved through not sharing binding anchor between hosts.

14. IANA Considerations

This document has no IANA actions.

[RFC Editor: please remove this section prior to publication.]

15. References

15.1. Normative References

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

[I-D.ietf-savi-framework]

Wu, J., Bi, J., Bagnulo, M., Baker, F., and C. Vogt, "Source Address Validation Improvement Framework", draft-ietf-savi-framework-04 (work in progress), March 2011.

[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 IPv6 Addresses", draft-ietf-savi-fcfs-09 (work in progress), April 2011.

[I-D.ietf-savi-mix]

Jun Bi, Guang Yao, J. Halpern and E. Levy-Abegnoli, "SAVI for Mixed Address Assignment Methods Scenario", draft-ietf-savi-mix-00 (work in progress), March 2011.

[RFC826] Plummer, D.C., "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.

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

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

[RFC3736] R. Droms, "Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6", RFC 3736, April 2004.

[RFC4388] R. Woundy and K. Kinnear, "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. Kinnear, B. Volz, S. Zeng, "DHCPv6 Leasequery", RFC5007, September 2007.

15.2. Informative References

[BCP38] 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.

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

[IP Source Guard]

Baker, F., "Cisco IP Version 4 Source Guard", IETF Internet
draft (work in progress), November 2007.

16. Acknowledgments

Special thanks to Jean-Michel Combes, Christian Vogt, Joel M. Halpern, Eric Levy-Abegnoli and Alberto Garcia for careful review and valuation comments on the state machine and text.
Thanks to Marcelo Bagnulo Braun, Mark Williams, Erik Nordmark, Mikael Abrahamsson, 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
Tsinghua University
Network Research Center, Tsinghua University
Beijing 100084
China
Email: junbi@tsinghua.edu.cn

Jianping Wu
Tsinghua University
Computer Science, Tsinghua University
Beijing 100084
China
Email: jianping@cernet.edu.cn

Guang Yao
Tsinghua University
Computer Science, Tsinghua University
Beijing 100084
China
Email: yaog@netarchlab.tsinghua.edu.cn

Fred Baker
Cisco Systems
Santa Barbara, CA 93117
United States
Email: fred@cisco.com

17. 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.
- Detection and Live states are deleted, together with corresponding sections.

Main change from 05 to 06:

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

Main changes from 06 to 07:

- So many changes in this modification. We suggest to track http://www.ietf.org/mailarchive/web/savi/current/msg01543.html. Changes are made according to the comments.

Main changes from 07 to 08,09:

- The modifications are made according to the comments from Jean-Michel Combes.