Internet Engineering Task Force (IETF)
Request for Comments: 7513
Category: Standards Track
ISSN: 2070-1721

J. Bi
J. Wu
G. Yao
Tsinghua Univ.
F. Baker
Cisco

May 2015

# Source Address Validation Improvement (SAVI) Solution for DHCP

## Abstract

This document specifies the procedure for creating a binding between a DHCPv4/DHCPv6-assigned IP address and a binding anchor on a Source Address Validation Improvement (SAVI) device. The bindings set up by this procedure are used to filter packets with forged source IP addresses. This mechanism complements BCP 38 (RFC 2827) ingress filtering, providing finer-grained source IP address validation.

## Status of This Memo

## Copyright Notice

Table of Contents

# 1. Introduction

This document describes a fine-grained source address validation mechanism for IPv4 and IPv6 packets. This mechanism creates bindings between IP addresses assigned to network interfaces by DHCP and suitable binding anchors (Section 4.3.5). As discussed in Section 3 and [RFC7039], a "binding anchor" is an attribute that is immutable or difficult to change that may be used to identify the system an IP address has been assigned to; common examples include a Media Access Control (MAC) address found on an Ethernet switch port or Wi-Fi security association. The bindings are used to identify and filter packets originated by these interfaces using forged source IP addresses. In this way, this mechanism can prevent hosts from using IP addresses assigned to any other attachment point in or not associated with the network. This behavior is referred to as "spoofing" and is key to amplification attacks, in which a set of systems send messages to another set of systems claiming to be from a third set of systems, and sending the replies to systems that don’t expect them. Whereas BCP 38 [RFC2827] protects a network from a neighboring network by providing prefix granularity source IP address validity, this mechanism protects a network, including a Local Area Network, from itself by providing address granularity source IP validity when DHCP/DHCPv6 is used to assign IPv4/IPv6 addresses. Both provide a certain level of traceability, in that packet drops indicate the presence of a system that is producing packets with spoofed IP addresses.

SAVI-DHCP snoops DHCP address assignments to set up bindings between IP addresses assigned by DHCP and corresponding binding anchors. It includes the DHCPv4 and DHCPv6 Snooping Process (Section 6) and the Data Snooping Process (Section 7), as well as a number of other technical details. The Data Snooping Process is a data-triggered procedure that snoops the IP header of data packets to set up bindings. It is designed to avoid a permanent blockage of valid addresses in the case that DHCP snooping is insufficient to set up all the valid bindings.

This mechanism is designed for the stateful DHCP scenario [RFC2131] [RFC3315]. Stateless DHCP [RFC3736] is out of scope for this document, as it has nothing to do with IP address allocation. An alternative SAVI method would have be used in those cases. For hosts using Stateless Address Autoconfiguration (SLAAC) to allocate addresses, First-Come, First-Served Source Address Validation Improvement (FCFS SAVI) [RFC6620] should be enabled. SAVI-DHCP is primarily designed for pure DHCP scenarios in which only addresses assigned through DHCP are allowed. However, it does not block link-local addresses, as they are not assigned using DHCP. It is RECOMMENDED that the administration deploy a SAVI solution for link-local addresses, e.g., FCFS SAVI [RFC6620].

This mechanism works for networks that use DHCPv4 only, DHCPv6 only, or both DHCPv4 and DHCPv6. However, the DHCP address assignment mechanism in IPv4/IPv6 transition scenarios, e.g., [RFC7341], are beyond the scope of this document.

## 2. Requirements Language

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

Binding anchor: A "binding anchor" is defined to be a physical and/or link-layer property of an attached device, as in [RFC7039]. A list of sample binding anchors can be found in Section 3.2 of that document. To the degree possible, a binding anchor associates an IP address with something unspoofable that identifies a single-client system or one of its interfaces. See Section 4.3.5 for more detail.

Attribute: A configurable property of each binding anchor (port, MAC address, or other information) that indicates the actions to be performed on packets received from the attached network device.

DHCP address: An IP address assigned via DHCP.

SAVI-DHCP: The name of this SAVI function for DHCP-assigned addresses.

SAVI device: A network device on which SAVI-DHCP is enabled.

Non-SAVI device: A network device on which SAVI-DHCP is not enabled.

DHCP Client-to-Server message: A message that is sent from a DHCP client to a DHCP server or DHCP servers and is one of the following types:

- DHCPv4 Discover: DHCPDISCOVER [RFC2131].
- DHCPv4 Request: DHCPREQUEST generated during SELECTING state [RFC2131].
- DHCPv4 Renew: DHCPREQUEST generated during RENEWING state [RFC2131].

- DHCPv4 Rebind: DHCPREQUEST generated during REBINDING state [RFC2131].
- DHCPv4 Reboot: DHCPREQUEST generated during INIT-REBOOT state [RFC2131].
- Note: DHCPv4 Request/Renew/Rebind/Reboot messages can be identified based on Table 4 of [RFC2131].
- DHCPv4 Decline: DHCPDECLINE [RFC2131].
- DHCPv4 Release: DHCPRELEASE [RFC2131].
- DHCPv4 Inform: DHCPINFORM [RFC2131].
- DHCPv4 DHCPLEASEQUERY: A message sent to inquire about the lease that might exist for an IPv4 address [RFC4388].
- DHCPv6 Request: REQUEST [RFC3315].
- DHCPv6 Solicit: SOLICIT [RFC3315].
- DHCPv6 Confirm: CONFIRM [RFC3315].
- DHCPv6 Decline: DECLINE [RFC3315].
- DHCPv6 Release: RELEASE [RFC3315].
- DHCPv6 Rebind: REBIND [RFC3315].
- DHCPv6 Renew: RENEW [RFC3315].
- DHCPv6 Information-Request: INFORMATION-REQUEST [RFC3315].
- DHCPv6 LEASEQUERY: A message sent to inquire about the lease that might exist for an IPv6 address [RFC5007].

DHCP Server-to-Client message: A message that is sent from a DHCP server to a DHCP client and is one of the following types:

- DHCPv4 ACK: DHCPACK [RFC2131].
- DHCPv4 NAK: DHCPNAK [RFC2131].
- DHCPv4 Offer: DHCPOFFER [RFC2131].
- DHCPv4 DHCPLEASEACTIVE: A response to a DHCPLEASEQUERY request containing lease information [RFC4388].

- DHCPv4 DHCPLEASEUNKNOWN: A response to a DHCPLEASEQUERY request indicating that the server does not manage the address [RFC4388].
- DHCPv4 DHCPLEASEUNASSIGNED: A response to a DHCPLEASEQUERY request indicating that the server manages the address and there is no current lease [RFC4388].
- DHCPv6 Reply: REPLY [RFC3315].
- DHCPv6 Advertise: ADVERTISE [RFC3315].
- DHCPv6 Reconfigure: RECONFIGURE [RFC3315].
- DHCPv6 LEASEQUERY-REPLY: A response to a LEASEQUERY request [RFC5007].

Lease time: The lease time in IPv4 [RFC2131] or the valid lifetime in IPv6 [RFC3315].

Binding entry: A rule that associates an IP address with a binding anchor.

Binding State Table (BST): The data structure that contains the binding entries.

Binding entry limit: The maximum number of binding entries that may be associated with a binding anchor. Limiting the number of binding entries per binding anchor prevents a malicious or malfunctioning node from overloading the binding table on a SAVI device.

Direct attachment: Ideally, a SAVI device is an access device that hosts are attached to directly. In such a case, the hosts are direct attachments (i.e., they attach directly) to the SAVI device.

Indirect attachment: A SAVI device MAY be an aggregation device that other access devices are attached to and that hosts in turn attach to. In such a case, the hosts are indirect attachments (i.e., they attach indirectly) to the SAVI device.

Unprotected link: Unprotected links are links that connect to hosts or networks of hosts that receive their DHCP traffic by another path and are therefore outside the SAVI perimeter.

Unprotected device: An unprotected device is a device associated with an unprotected link. One example might be the gateway router of a network.

Protected link: If DHCP messages for a given attached device always use a given link, the link is considered to be "protected" by the SAVI device and is therefore within the SAVI perimeter.

Protected device: A protected device is a device associated with a protected link. One example might be a desktop switch in the network, or a host.

Cut vertex: A cut vertex is any vertex whose removal increases the number of connected components in a (network) graph. This is a concept in graph theory. This term is used in Section 6.1 to accurately specify the required deployment location of SAVI devices when they only perform the DHCP Snooping Process.

Identity Association (IA): "A collection of addresses assigned to a client" [RFC3315].

Detection message: A Neighbor Solicitation or ARP message intended by the Data Snooping Process to detect a duplicate address.

DHCP_DEFAULT_LEASE: Default lifetime for a DHCPv6 address when the binding is triggered by a DHCPv6 Confirm message but a DHCPv6 Leasequery exchange [RFC5007] cannot be performed by the SAVI device to fetch the lease.

## 4. Deployment Scenario and Configuration

### 4.1. Elements and Scenario

The essential elements in a SAVI-DHCP deployment scenario include at least one DHCP server (which may or may not be assigned an address using DHCP and therefore may or may not be protected), zero or more protected DHCP clients, and one or more SAVI devices. It may also include DHCP relays, when the DHCP server is not co-located with a set of clients, and zero or more protected non-SAVI devices. Outside the perimeter, via unprotected links, there may be many unprotected devices.

```
                                 +-------------+
                                 | Unprotected |
                                 |   Device    |
                                 +------+------+
                                        |
                   +--------+     +-----+------+    +----------+
                   |DHCP    +-----+  Non-SAVI  +----+Bogus DHCP|
                   |Server A|     |  Device 1  |    |Server    |
                   +--------+     +-----+------+    +----------+
                                        |trusted, unprotected link
       . . . . . . . . . . . . . . . . .|. . . . . . . . . . . . . .
      .                                 |                           .
      .             Protection      +---+------+ trusted link       .
      .             Perimeter       | SAVI     +--------------+     .
      .                             | Device C |              |     .
      .                             +---+------+              |     .
      .                                 |                     |     .
      .  untrusted, +----------+    +---+------+       +------+---+ .
      .  protected  | SAVI     |    | Non-SAVI |       | SAVI     | .
      .  link+------+ Device A +----+ Device 3 +-------+ Device B | .
      .      |      +----+--+--+    +----------+       +-+---+----+ .
      .      |           |  +----------+    . . . .  .   |   |      .
      .      |       . . . . . .       |   .          .  |   |      .
      .      |      .    |      .      |   .    +--------+   |      .
      . +----+-----+. +--+---+  . +----+-+ . +--+---+ .  +---+----+ .
      . | Non-SAVI |. |Client|  . |DHCP  | . |Client| .  |DHCP    | .
      . | Device 2 |. |A     |  . |Relay | . |B     | .  |Server B| .
      . +----------+. +------+  . +------+ . +------+ .  +--------+ .
       . . . . . . .             . . . . .             . . . . . . .
```

Figure 1: SAVI-DHCP Scenario

Figure 1 shows a deployment scenario that contains these elements. Note that a physical device can instantiate multiple elements, e.g., a switch can be both a SAVI device and a DHCP relay, or in a cloud-computing environment, a physical host may contain a virtual switch plus some number of virtual hosts. In such cases, the links are logical links rather than physical links.

Networks are not usually isolated. As a result, traffic from other networks, including transit traffic as specified in [RFC6620] (e.g., traffic from another SAVI switch or a router) may enter a SAVI-DHCP network through the unprotected links. Since SAVI solutions are limited to validating traffic generated from a local link, SAVI-DHCP does not set up bindings for addresses assigned in other networks and cannot validate them. Traffic from unprotected links should be checked by an unprotected device or mechanisms described in

[RFC2827]. The generation and deployment of such a mechanism is beyond the scope of this document.

Traffic from protected links is, however, locally generated and should have its source addresses validated by SAVI-DHCP if possible. In the event that there is an intervening protected non-SAVI device between the host and the SAVI device, however, use of the physical attachment point alone as a binding anchor is insufficiently secure, as several devices on a port or other point of attachment can spoof each other. Hence, additional information such as a MAC address SHOULD be used to disambiguate them.

### 4.2. SAVI Binding Type Attributes

As illustrated in Figure 1, a system attached to a SAVI device can be a DHCP client, a DHCP relay/server, a SAVI device, or a non-SAVI device. Different actions are performed on traffic originated from different elements. To distinguish among their requirements, several properties are associated with their point of attachment on the SAVI device.

When a binding association is uninstantiated, e.g., when no host is attached to the SAVI device using a given port or other binding anchor, the binding port attributes take default values unless overridden by configuration. By default, a SAVI switch does not filter DHCP messages, nor does it attempt to validate source addresses, which is to say that the binding attributes are ignored until SAVI-DHCP is itself enabled. This is because a SAVI switch that depends on DHCP cannot tell, a priori, which ports have valid DHCP servers attached, or which have routers or other equipment that would validly appear to use an arbitrary set of source addresses. When SAVI has been enabled, the attributes take effect.

#### 4.2.1. Trust Attribute

The "Trust Attribute" is a Boolean value. If TRUE, it indicates that the packets from the corresponding attached device need not have their source addresses validated. Examples of a trusted attachment would be a port to another SAVI device, or to an IP router, as shown in Figure 1. In both cases, traffic using many source IP addresses will be seen. By default, the Trust attribute is FALSE, indicating that any device found on that port will seek an address using DHCP and be limited to using such addresses.

SAVI devices will not set up bindings for points of attachment with the Trust attribute set TRUE; no packets, including DHCP messages, from devices with this attribute on their attachments will be validated. However, DHCP Server-to-Client messages will be snooped on attachment points with the Trust attribute set TRUE in the same way as if they had the DHCP-Trust attribute set (see Section 4.2.2).

### 4.2.2. DHCP-Trust Attribute

The "DHCP-Trust Attribute" is similarly a Boolean attribute. It indicates whether the attached device is permitted to initiate DHCP Server-to-Client messages. In Figure 1, the points of attachment of the DHCP server and the DHCP relay would have this attribute set TRUE, and attachment points that have Trust set TRUE are implicitly treated as if DHCP-Trust is TRUE.

If the DHCP-Trust attribute is TRUE, SAVI devices will forward DHCP Server-to-Client messages from the points of attachment with this attribute. If the DHCP Server-to-Client messages can trigger the state transitions, the binding setup processes specified in Sections 6 and 7 will handle them. By default, the DHCP-Trust attribute is FALSE, indicating that the attached system is not a DHCP server.

A DHCPv6 implementor can refer to [DHCPv6-SHIELD] for more details.

### 4.2.3. DHCP-Snooping Attribute

The "DHCP-Snooping Attribute" is similarly a Boolean attribute. It indicates whether bindings will be set up based on DHCP snooping.

If this attribute is TRUE, DHCP Client-to-Server messages to points of attachment with this attribute will trigger creation of bindings based on the DHCP Snooping Process described in Section 6. If it is FALSE, either the Trust attribute must be TRUE (so that bindings become irrelevant) or another SAVI mechanism such as FCFS SAVI must be used on the point of attachment.

The DHCP-Snooping attribute is configured on the DHCP client’s point of attachment. This attribute can be also used on the attachments to protected non-SAVI devices that are used by DHCP clients. In Figure 1, the attachment from Client A to SAVI Device A, the attachment from Client B to SAVI Device B, and the attachment from Non-SAVI Device 2 to SAVI Device A can be configured with this attribute.

### 4.2.4. Data-Snooping Attribute

The "Data-Snooping Attribute" is a Boolean attribute. It indicates whether data packets from the corresponding point of attachment may trigger the binding setup procedure.

Data packets from points of attachment with this attribute may trigger the setup of bindings. SAVI devices will set up bindings on points of attachment with this attribute based on the data-triggered process described in Section 7.

If the DHCP-Snooping attribute is configured on a point of attachment, the bindings on this attachment are set up based on DHCP message snooping. However, in some scenarios, a DHCP client may use a DHCP address without the DHCP address assignment procedure being performed on its current attachment. For such attached devices, the Data Snooping Process, which is described in Section 7, is necessary. This attribute is configured on such attachments. The usage of this attribute is further discussed in Section 7.

Since some networks require DHCP deployment and others avoid it, there is no obvious universal default value for the Data-Snooping attribute. Hence, the Data-Snooping attribute should default to FALSE, and a mechanism should be implemented to conveniently set it to TRUE on all points of attachment for which the Trust attribute is FALSE.

#### 4.2.5. Validating Attribute

The "Validating Attribute" is a Boolean attribute. It indicates whether packets from the corresponding attachment will have their IP source addresses validated based on binding entries on the attachment.

If it is TRUE, packets coming from attachments with this attribute will be validated based on binding entries on the attachment as specified in Section 8. If it is FALSE, they will not. Since the binding table is used in common with other SAVI algorithms, it merely signifies whether the check will be done, not whether it will be done for SAVI-DHCP originated bindings.

This attribute is by default the inverse of the Trust attribute; source addresses on untrusted links are validated by default. It MAY be set FALSE by the administration.

The expected use case is when SAVI is used to monitor but not block forged transmissions. The network manager, in that case, may set the DHCP-Snooping and/or Data-Snooping attribute TRUE but the Validating attribute FALSE.

### 4.2.6. Table of Mutual Exclusions

Different types of attributes may indicate mutually exclusive actions on a packet. Mutually exclusive attributes MUST NOT be set TRUE on the same attachment. The compatibility of different attributes is listed in Figure 2. Note that although Trust and DHCP-Trust are compatible, there is no need to configure DHCP-Trust to TRUE on an attachment with Trust attribute TRUE.

| | Trust | DHCP-Trust | DHCP-Snooping | Data-Snooping | Validating |
|---|---|---|---|---|---|
| Trust | - | compatible | mutually exclusive | mutually exclusive | mutually exclusive |
| DHCP-Trust | compatible | - | compatible | compatible | compatible |
| DHCP-Snooping | mutually exclusive | compatible | - | compatible | compatible |
| Data-Snooping | mutually exclusive | compatible | compatible | - | compatible |
| Validating | mutually exclusive | compatible | compatible | compatible | - |

Figure 2: Table of Mutual Exclusions

## 4.3. Perimeter

### 4.3.1. SAVI-DHCP Perimeter Overview

SAVI devices form a perimeter separating trusted and untrusted regions of a network, as FCFS SAVI does (Section 2.5 of [RFC6620]). The perimeter is primarily designed for scalability. It has two implications.

- SAVI devices only need to establish bindings for directly attached clients, or clients indirectly attached through a non-SAVI protected device, rather than all of the clients in the network.

- Each SAVI device only needs to validate the source addresses in traffic from clients attached to it, without checking all the traffic passing by.

Consider the example in Figure 1. The protection perimeter is formed by SAVI Devices A, B, and C. In this case, SAVI Device B does not create a binding for Client A. However, because SAVI Device A filters spoofed traffic from Client A, SAVI Device B can avoid receiving spoofed traffic from Client A.

The perimeter in SAVI-DHCP is not only a perimeter for data packets but also a perimeter for DHCP messages. DHCP server response messages incoming across the perimeter will be dropped (Section 8). The placement of the DHCP relay and DHCP server, which are not involved in [RFC6620], is related to the construction of the perimeter. The requirement on the placement and configuration of the DHCP relay and DHCP server is discussed in Section 4.3.3.

#### 4.3.2. SAVI-DHCP Perimeter Configuration Guideline

A perimeter separating trusted and untrusted regions of the network is formed as follows:

(1) Configure the Validating and DHCP-Snooping attributes TRUE on the direct attachments of all DHCP clients.

(2) Configure the Validating and DHCP-Snooping attributes TRUE on the indirect attachments of all DHCP clients (i.e., DHCP clients on protected links).

(3) Configure the Trust attribute TRUE on the attachments to other SAVI devices.

(4) If a non-SAVI device, or a number of connected non-SAVI devices, are attached only to SAVI devices, set the Trust attribute TRUE on their attachments.

(5) Configure the DHCP-Trust attribute TRUE on the direct attachments to trusted DHCP relays and servers.

In this way, the points of attachments with the Validating attribute TRUE (and generally together with attachments of unprotected devices) on SAVI devices can form a perimeter separating DHCP clients and trusted devices. Data packet checks are only performed on the perimeter. The perimeter is also a perimeter for DHCP messages. The DHCP-Trust attribute is only TRUE on links inside the perimeter. Only DHCP Server-to-Client messages originated within the perimeter are trusted.

### 4.3.3. On the Placement of the DHCP Server and Relay

As a result of the configuration guidelines, SAVI devices only trust DHCP Server-to-Client messages originated inside the perimeter. Thus, the trusted DHCP relays and DHCP servers must be placed within the perimeter. DHCP Server-to-Client messages will be filtered on the perimeter. Server-to-Relay messages will not be filtered, as they are within the perimeter. In this way, DHCP Server-to-Client messages from bogus DHCP servers are filtered on the perimeter, having entered through untrusted points of attachment. The SAVI devices are protected from forged DHCP messages.

DHCP Server-to-Client messages arriving at the perimeter from outside the perimeter are not trusted. There is no distinction between a DHCP server owned and operated by the correct administration but outside the SAVI perimeter and a bogus DHCP server. For example, in Figure 1, DHCP Server A is valid, but it is attached to Non-SAVI Device 1. A bogus DHCP server is also attached to Non-SAVI Device 1. While one could imagine a scenario in which the valid one had a statistically configured port number and MAC address, and therefore a binding, by default SAVI-DHCP cannot distinguish whether a message received from the port of Non-SAVI Device 1 is from DHCP Server A or the bogus DHCP server. If DHCP Server A is contained in the perimeter, Non-SAVI Device 1 will also be contained in the perimeter. Thus, DHCP Server A cannot be contained within the perimeter apart from manual configuration of the binding anchor.

Another consideration on the placement is that if the DHCP server/relay is not inside the perimeter, the SAVI devices may not be able to set up bindings correctly because the SAVI devices may not be on the path between the clients and the server/relay, or the DHCP messages are encapsulated (e.g., Relay-reply and Relay-forward).

### 4.3.4. An Alternative Deployment

In common deployment practice, the traffic from the unprotected network is treated as trustworthy, which is to say that it is not filtered. In such a case, the Trust attribute can be set TRUE on the unprotected link. If non-SAVI devices, or a number of connected non-SAVI devices, are only attached to SAVI devices and unprotected devices, their attachment to SAVI devices can have the Trust attribute set TRUE. Then an unclosed perimeter will be formed, as illustrated in Figure 3.

```
           |             .             .           Protection |
           |             |             |           Perimeter  |
           |             |             |                      |
           | Unprotected |             | Unprotected          |
           | Link        |             | Link                 |
           |             |             |                      |
           |             |             |                      |
           |        +----+---+    +----+---+    +--------+    |
           |        |SAVI    +----+Non-SAVI+----+SAVI    |    |
           |        |Device  |    |Device  |    |Device  |    |
           |        +----+---+    +--------+    +----+---+    |
           |             |                           |        |
           \_____________+___________________________+________/
                         |                           |
                         |                           |
                    +--------+                  +--------+
                    |DHCP    |                  |DHCP    |
                    |Client  |                  |Client  |
                    +--------+                  +--------+
```

Figure 3: Alternative Perimeter Configuration

### 4.3.5. Considerations regarding Binding Anchors

The strength of this binding-based mechanism depends on the strength of the binding anchor. The sample binding anchors in [RFC7039] have the property in which they associate an IP address with a direct physical or secure virtual interface such as a switch port, a subscriber association, or a security association. In addition, especially in the case where a protected non-SAVI device such as a desktop switch or a hub is between the client and SAVI devices, they MAY be extended to also include a MAC address or other link-layer attribute. In short, a binding anchor is intended to associate an IP address with something unspoofable that identifies a single-client system or one of its interfaces; this may be a physical or virtual interface or that plus disambiguating link-layer information.

If the binding anchor is spoofable, such as a plain MAC address, or non-exclusive, such as a switch port extended using a non-SAVI device, an attacker can use a forged binding anchor to evade validation. Indeed, using a binding anchor that can be easily spoofed can lead to worse outcomes than allowing spoofed IP traffic. Thus, a SAVI device MUST use a non-spoofable and exclusive binding anchor.

### 4.4. Other Device Configuration

In addition to a possible binding anchor configuration specified in Section 4.2, an implementation has the following configuration requirements:

(1) Address configuration. For DHCPv4: the SAVI device MUST have an IPv4 address. For DHCPv6: the client of a SAVI device MUST have a link-local address; when the DHCPv6 server is not on the same link as the SAVI device, the SAVI device MUST also have an IPv6 address of at least the same scope as the DHCPv6 Server.

(2) DHCP server address configuration: a SAVI device MUST store the list of the DHCP server addresses that it could contact during a leasequery process.

(3) A SAVI device may also require security parameters, such as preconfigured keys to establish a secure connection for the leasequery process [RFC4388] [RFC5007] connection.

## 5. Binding State Table (BST)

The Binding State Table, which may be implemented centrally in the switch or distributed among its ports, is used to contain the bindings between the IP addresses assigned to the attachments and the corresponding binding anchors of the attachments. Note that in this description, there is a binding entry for each IPv4 or IPv6 address associated with each binding anchor, and there may be several of each such address, especially if the port is extended using a protected non-SAVI device. Each binding entry has six fields:

- Binding Anchor (listed as "Anchor" in subsequent figures): the binding anchor, i.e., one or more physical and/or link-layer properties of the attachment.

- IP Address (listed as "Address" in subsequent figures): the IPv4 or IPv6 address assigned to the attachment by DHCP.

- State: the state of the binding. Possible values of this field are listed in Sections 6.2 and 7.3.

- Lifetime: the remaining seconds of the binding. Internally, this MAY be stored as the timestamp value at which the lifetime expires.

- Transaction ID (TID): the Transaction ID [RFC2131] [RFC3315] of the corresponding DHCP transaction. The TID field is used to associate DHCP Server-to-Client messages with corresponding binding entries.

- Timeouts: the number of timeouts that expired in the current state (only used in the Data Snooping Process; see Section 7).

The IA is not present in the BST for three reasons:

- The lease of each address in one IA is assigned separately.

- When the binding is set up based on data snooping, the IA cannot be recovered from the leasequery protocol.

- DHCPv4 does not define an IA.

An example of such a table is shown in Figure 4.

| Anchor | Address | State | Lifetime | TID | Timeouts |
|---|---|---|---|---|---|
| Port_1 | IP_1 | BOUND | 65535 | TID_1 | 0 |
| Port_1 | IP_2 | BOUND | 10000 | TID_2 | 0 |
| Port_2 | IP_3 | INIT_BIND | 1 | TID_3 | 0 |

Figure 4: Example Binding State Table

## 6. DHCP Snooping Process

This section specifies the process of setting up bindings based on DHCP snooping. This process is illustrated using a state machine.

### 6.1. Rationale

The rationale of the DHCP Snooping Process is that if a DHCP client is legitimately using a DHCP-assigned address, the DHCP address assignment procedure that assigns the IP address to the client must have been performed via the client’s point of attachment. This assumption works when the SAVI device is always on the path(s) from the DHCP client to the DHCP server(s)/relay(s). Without considering the movement of DHCP clients, the SAVI device should be the cut vertex whose removal will separate the DHCP client and the remaining network containing the DHCP server(s)/relay(s). For most of the networks whose topologies are simple, it is possible to deploy this SAVI function at proper devices to meet this requirement.

However, if there are multiple paths from a DHCP client to the DHCP server and the SAVI device is only on one of them, there is an obvious failure case: the SAVI device may not be able to snoop the DHCP procedure. Host movement may also make this requirement difficult to meet. For example, when a DHCP client moves from one attachment to another attachment in the same network, it may fail to reinitialize its interface or send a Confirm message because of incomplete protocol implementation. Thus, there can be scenarios in which only performing this DHCP Snooping Process is insufficient to set up bindings for all the valid DHCP addresses. These exceptions and the solutions are discussed in Section 7.

## 6.2. Binding States Description

The following binding states are present in this process and the corresponding state machine:

NO_BIND: No binding has been set up.

INIT_BIND: A potential binding has been set up.

BOUND: The binding has been set up.

## 6.3. Events

This section describes events in this process and the corresponding state machine transitions. The DHCP message categories (e.g., DHCPv4 Discover) defined in Section 3 are used extensively in the definitions of events and elsewhere in the state machine definition. If an event will trigger the creation of a new binding entry, the binding entry limit on the binding anchor MUST NOT be exceeded.

### 6.3.1. Timer Expiration Event

EVE_ENTRY_EXPIRE: The lifetime of a binding entry expires.

### 6.3.2. Control Message Arriving Events

EVE_DHCP_REQUEST: A DHCPv4 Request or a DHCPv6 Request message is received.

EVE_DHCP_CONFIRM: A DHCPv6 Confirm message is received.

EVE_DHCP_REBOOT: A DHCPv4 Reboot message is received.

EVE_DHCP_REBIND: A DHCPv4 Rebind or a DHCPv6 Rebind message is received.

EVE_DHCP_RENEW: A DHCPv4 Renew or a DHCPv6 Renew message is received.

EVE_DHCP_SOLICIT_RC: A DHCPv6 Solicitation message with the Rapid Commit option is received.

EVE_DHCP_REPLY: A DHCPv4 ACK or a DHCPv6 Reply message is received.

EVE_DHCP_DECLINE: A DHCPv4 Decline or a DHCPv6 Decline message is received.

EVE_DHCP_RELEASE: A DHCPv4 Release or a DHCPv6 Release message is received.

EVE_DHCP_LEASEQUERY: A successful DHCPv6 LEASEQUERY-REPLY (refer to Section 4.3.3 of [RFC5007]) is received.

Note: the events listed here do not cover all the DHCP messages in Section 3. The messages that do not really determine address usage (DHCPv4 Discover, DHCPv4 Inform, DHCPv6 Solicit without Rapid Commit, DHCPv6 Information-Request, DHCPv4 Offer, DHCPv6 Advertise, and DHCPv6 Reconfigure) and that are not necessary to snoop (DHCPv4 Negative Acknowledgment (NAK); refer to Section 6.4.2.3) are not included. Note also that DHCPv4 DHCPLEASEQUERY is not used in the DHCP Snooping Process to avoid confusion with Section 7. Also, since the LEASEQUERY should have been originated by the SAVI device itself, the destination check should verify that the message is directed to this SAVI device, and it should not be forwarded once it has been processed here.

Moreover, only if a DHCP message can pass the following checks, the corresponding event is regarded as a valid event:

o Attribute check: the DHCP Server-to-Client messages and LEASEQUERY-REPLY should be from attachments with the DHCP-Trust attribute; the DHCP Client-to-Server messages should be from attachments with the DHCP-Snooping attribute.

o Destination check: the DHCP Server-to-Client messages should be destined to attachments with the DHCP-Snooping attribute. This check is performed to ensure the binding is set up on the SAVI device that is nearest to the destination client.

o Binding anchor check: the DHCP Client-to-Server messages that may trigger modification or removal of an existing binding entry must have a matching binding anchor with the corresponding entry.

- TID check: the DHCP Server-to-Client/Client-to-Server messages that may cause modification of existing binding entries must have a matched TID with the corresponding entry. Note that this check is not performed on LEASEQUERY and LEASEQUERY-REPLY messages as they are exchanged between the SAVI devices and the DHCP servers. Besides, this check is not performed on DHCP Renew/Rebind messages.

- Binding limitation check: the DHCP messages must not cause new binding setup on an attachment whose binding entry limitation has been reached (refer to Section 11.5).

- Address check: the source address of the DHCP messages should pass the check specified in Section 8.2.

On receiving a DHCP message without triggering a valid event, the state will not change, and the actions will not be performed. Note that if a message does not trigger a valid event but it can pass the checks in Section 8.2, it MUST be forwarded.

## 6.4. The State Machine of DHCP Snooping Process

This section specifies state transitions and their corresponding actions.

### 6.4.1. Initial State: NO_BIND

#### 6.4.1.1. Event: EVE_DHCP_REQUEST - A DHCPv4 Request or a DHCPv6 Request message is received

The SAVI device MUST forward the message.

The SAVI device will generate an entry in the BST. The Binding Anchor field is set to the binding anchor of the attachment from which the message is received. The State field is set to INIT_BIND. The Lifetime field is set to be MAX_DHCP_RESPONSE_TIME. The TID field is set to the TID of the message. If the message is DHCPv4 Request, the Address field can be set to the address to request, i.e., the ’requested IP address’. An example of the entry is illustrated in Figure 5.

| Anchor | Address | State | Lifetime | TID | Timeouts |
|---|---|---|---|---|---|
| Port_1 | | INIT_BIND | MAX_DHCP_RESPONSE_TIME | TID | 0 |

Figure 5: Binding Entry in BST on Initialization Triggered by Request/Rapid Commit/Reboot Messages

Resulting state: INIT_BIND - A potential binding has been set up.

#### 6.4.1.2. Event: EVE_DHCP_REBOOT - A DHCPv4 Reboot message is received

The SAVI device MUST forward the message.

The SAVI device will generate an entry in the BST. The Binding Anchor field is set to the binding anchor of the attachment from which the message is received. The State field is set to INIT_BIND. The Lifetime field is set to be MAX_DHCP_RESPONSE_TIME. The TID field is set to the TID of the message. If the message is DHCPv4 Reboot, the Address field can be set to the address to request, i.e., the 'requested IP address'. An example of the entry is illustrated in Figure 5.

Resulting state: INIT_BIND - A potential binding has been set up.

#### 6.4.1.3. Event: EVE_DHCP_SOLICIT_RC - A DHCPv6 Solicitation message with the Rapid Commit option is received

The SAVI device MUST forward the message.

The SAVI device will generate an entry in the BST. The Binding Anchor field is set to the binding anchor of the attachment from which the message is received. The State field is set to INIT_BIND. The Lifetime field is set to be MAX_DHCP_RESPONSE_TIME. The TID field is set to the TID of the message. An example of the entry is illustrated in Figure 5.

Resulting state: INIT_BIND - A potential binding has been set up.

#### 6.4.1.4. Event: EVE_DHCP_CONFIRM - A DHCPv6 Confirm message is received

The SAVI device MUST forward the message.

The SAVI device will generate corresponding entries in the BST for each address in each Identity Association (IA) option of the Confirm message. The Binding Anchor field is set to the binding anchor of the attachment from which the message is received. The State field is set to INIT_BIND. The Lifetime field is set to be MAX_DHCP_RESPONSE_TIME. The TID field is set to the TID of the message. The Address field is set to the address(es) to confirm. An example of the entries is illustrated in Figure 6.

| Anchor | Address | State | Lifetime | TID | Timeouts |
|---|---|---|---|---|---|
| Port_1 | Addr1 | INIT_BIND | MAX_DHCP_RESPONSE_TIME | TID | 0 |
| Port_1 | Addr2 | INIT_BIND | MAX_DHCP_RESPONSE_TIME | TID | 0 |

Figure 6: Binding Entry in BST on Confirm-Triggered Initialization

Resulting state: INIT_BIND - A potential binding has been set up.

##### 6.4.1.5. Events That Cannot Happen in the NO_BIND State

- EVE_ENTRY_EXPIRE: The lifetime of a binding entry expires
- EVE_DHCP_REBIND: A DHCPv4 Rebind or a DHCPv6 Rebind message is received
- EVE_DHCP_RENEW: A DHCPv4 Renew or a DHCPv6 Renew message is received
- EVE_DHCP_REPLY: A DHCPv4 ACK or a DHCPv6 Reply message is received
- EVE_DHCP_DECLINE: A DHCPv4 Decline or a DHCPv6 Decline message is received
- EVE_DHCP_RELEASE: A DHCPv4 Release or a DHCPv6 Release message is received
- EVE_DHCP_LEASEQUERY: A successful DHCPv6 LEASEQUERY-REPLY is received

These cannot happen because they are each something that happens AFTER a binding has been created.

### 6.4.2. Initial State: INIT_BIND

#### 6.4.2.1. Event: EVE_DHCP_REPLY - A DHCPv4 ACK or a DHCPv6 Reply message is received

The message MUST be forwarded to the corresponding client.

If the message is DHCPv4 ACK, the Address field of the corresponding entry (i.e., the binding entry whose TID is the same as the message) is set to the address in the message (i.e., 'yiaddr' in DHCPv4 ACK). The Lifetime field is set to the sum of the lease time in the ACK message and MAX_DHCP_RESPONSE_TIME. The State field is changed to BOUND.

If the message is DHCPv6 Reply, note the following cases:

1. If the status code is not "Success", no modification of corresponding entries will be made. Corresponding entries will expire automatically if no "Success" Reply is received during the lifetime. The entries are not removed immediately because the client may be able to use the addresses whenever a "Success" Reply is received ("If the client receives any Reply messages that do not indicate a NotOnLink status, the client can use the addresses in the IA and ignore any messages that indicate a NotOnLink status" [RFC3315]).

2. If the status code is "Success", the SAVI device checks the IA options in the Reply message.

    A. If there are IA options in the Reply message, the SAVI device checks each IA option. When the first assigned address is found, the Address field of the binding entry with a matched TID is set to the address. The Lifetime field is set to the sum of the lease time in the Reply message and MAX_DHCP_RESPONSE_TIME. The State field is changed to BOUND. If there is more than one address assigned in the message, new binding entries are set up for the remaining address assigned in the IA options. An example of the entries is illustrated in Figure 8. SAVI devices do not specially process IA options with a NoAddrsAvail status because there should be no address contained in such IA options.

    B. Otherwise, the DHCP Reply message is in response to a Confirm message. The state of the binding entries with a matched TID is changed to BOUND. Because [RFC3315] does not require the lease time of addresses to be contained in the Reply message, the SAVI device SHOULD send a LEASEQUERY [RFC5007] message querying by IP address to the All_DHCP_Servers multicast address [RFC3315] or a list of configured DHCP server addresses. The LEASEQUERY message is generated for each IP address if multiple addresses are confirmed. The lifetime of corresponding entries is set to 2*MAX_LEASEQUERY_DELAY. If there is no response message after MAX_LEASEQUERY_DELAY, send the LEASEQUERY message again. An example of the entries is illustrated in Figure 7. If the SAVI device does not send the LEASEQUERY message, a preconfigured lifetime DHCP_DEFAULT_LEASE MUST be set on the corresponding entry. (Note: it is RECOMMENDED to use T1 configured on DHCP servers as the DHCP_DEFAULT_LEASE.)

Note: the SAVI devices do not check if the assigned addresses are duplicated because in SAVI-DHCP scenarios, the DHCP servers are the only source of valid addresses. However, the DHCP servers should be configured to make sure no duplicated addresses are assigned.

| Anchor | Address | State | Lifetime | TID | Timeouts |
|---|---|---|---|---|---|
| Port_1 | Addr1 | BOUND | 2*MAX_LEASEQUERY_DELAY | TID | 0 |
| Port_1 | Addr2 | BOUND | 2*MAX_LEASEQUERY_DELAY | TID | 0 |

Figure 7: From INIT_BIND to BOUND on DHCP Reply in Response to Confirm

Transition

| Anchor | Address | State | Lifetime | TID | Timeouts |
|---|---|---|---|---|---|
| Port_1 | Addr1 | BOUND | Lease time+ MAX_DHCP_RESPONSE_TIME | TID | 0 |
| Port_1 | Addr2 | BOUND | Lease time+ MAX_DHCP_RESPONSE_TIME | TID | 0 |

Figure 8: From INIT_BIND to BOUND on DHCP Reply in Response to Request

Resulting state: BOUND - The binding has been set up.

#### 6.4.2.2. Event: EVE_ENTRY_EXPIRE - The lifetime of a binding entry expires

The entry MUST be deleted from the BST.

Resulting state: An entry that has been deleted from the BST may be considered to be in the "NO_BIND" state - No binding has been set up.

#### 6.4.2.3. Events That Are Ignored in INIT_BIND

If no DHCP Server-to-Client messages that assign addresses or confirm addresses are received, corresponding entries will expire automatically. Thus, other DHCP Server-to-Client messages (e.g., DHCPv4 NAK) are not specially processed.

As a result, the following events, should they occur, are ignored until either a DHCPv4 ACK or a DHCPv6 Reply message is received or the lifetime of the binding entry expires.

- EVE_DHCP_REQUEST: A DHCPv4 Request or a DHCPv6 Request message is received
- EVE_DHCP_CONFIRM: A DHCPv6 Confirm message is received
- EVE_DHCP_REBOOT: A DHCPv4 Reboot message is received
- EVE_DHCP_REBIND: A DHCPv4 Rebind or a DHCPv6 Rebind message is received
- EVE_DHCP_RENEW: A DHCPv4 Renew or a DHCPv6 Renew message is received
- EVE_DHCP_SOLICIT_RC: A DHCPv6 Solicitation message with the Rapid Commit option is received
- EVE_DHCP_DECLINE: A DHCPv4 Decline or a DHCPv6 Decline message is received
- EVE_DHCP_RELEASE: A DHCPv4 Release or a DHCPv6 Release message is received
- EVE_DHCP_LEASEQUERY: A successful DHCPv6 LEASEQUERY-REPLY is received

In each case, the message MUST be forwarded.

Resulting state: INIT_BIND - A potential binding has been set up.

### 6.4.3. Initial State: BOUND

#### 6.4.3.1. Event: EVE_ENTRY_EXPIRE - The lifetime of a binding entry expires

The entry MUST be deleted from the BST.

Resulting state: An entry that has been deleted from the BST may be considered to be in the "NO_BIND" state - No binding has been set up.

#### 6.4.3.2. Event: EVE_DHCP_DECLINE - A DHCPv4 Decline or a DHCPv6 Decline message is received

The message MUST be forwarded.

First, the SAVI device gets all the addresses ("Requested IP address" in DHCPv4 Decline, "ciaddr" in DHCPv4 Release, and addresses in all the IA options of DHCPv6 Decline/Release) to decline/release in the message. Then, the corresponding entries MUST be removed.

Resulting state in each relevant BST entry: An entry that has been deleted from the BST may be considered to be in the "NO_BIND" state - No binding has been set up.

#### 6.4.3.3. Event: EVE_DHCP_RELEASE - A DHCPv4 Release or a DHCPv6 Release message is received

The message MUST be forwarded.

First, the SAVI device gets all the addresses ("Requested IP address" in DHCPv4 Decline, "ciaddr" in DHCPv4 Release, and addresses in all the IA options of DHCPv6 Decline/Release) to decline/release in the message. Then, the corresponding entries MUST be removed.

Resulting state in each relevant BST entry: An entry that has been deleted from the BST may be considered to be in the "NO_BIND" state - No binding has been set up.

#### 6.4.3.4. Event: EVE_DHCP_REBIND - A DHCPv4 Rebind or a DHCPv6 Rebind message is received

The message MUST be forwarded.

In such a case, a new TID will be used by the client. The TID field of the corresponding entries MUST be set to the new TID. Note that the TID check will not be performed on such messages.

Resulting state: BOUND: The binding has been set up.

#### 6.4.3.5. Event: EVE_DHCP_RENEW - A DHCPv4 Renew or a DHCPv6 Renew message is received

The message MUST be forwarded.

In such a case, a new TID will be used by the client. The TID field of the corresponding entries MUST be set to the new TID. Note that the TID check will not be performed on such messages.

Resulting state: BOUND: The binding has been set up.

#### 6.4.3.6. Event: EVE_DHCP_REPLY - A DHCPv4 ACK or a DHCPv6 Reply message is received

The message MUST be forwarded.

The DHCP Reply messages received in current states should be in response to DHCP Renew/Rebind.

If the message is DHCPv4 ACK, the SAVI device updates the binding entry with a matched TID, with the Lifetime field set to be the sum of the new lease time and MAX_DHCP_RESPONSE_TIME, leaving the entry in the BOUND state.

If the message is DHCPv6 Reply, the SAVI device checks each IA Address option in each IA option. For each:

1. If the IA entry in the REPLY message has the status "NoBinding", there is no address in the option, and no operation on an address is performed.

2. If the valid lifetime of an IA Address option is 0, the binding entry with a matched TID and address is removed, leaving it effectively in the NO_BIND state.

3. Otherwise, set the Lifetime field of the binding entry with the matched TID and address to be the sum of the new valid lifetime and MAX_DHCP_RESPONSE_TIME, leaving the entry in the BOUND state.

Resulting state: NO_BIND or BOUND, as specified.

#### 6.4.3.7. Event: EVE_DHCP_LEASEQUERY - A successful DHCPv6 LEASEQUERY_REPLY is received

The message MUST be forwarded.

The message should be in response to the LEASEQUERY message sent in Section 6.4.2. The related binding entry can be determined based on the address in the IA Address option in the LEASEQUERY-REPLY message. The Lifetime field of the corresponding binding entry is set to the sum of the lease time in the LEASEQUERY-REPLY message and MAX_DHCP_RESPONSE_TIME.

Resulting state: BOUND: The binding has been set up.

#### 6.4.3.8. Events Not Processed in the State BOUND

The following events are ignored if received while the indicated entry is in the BOUND state. Any required action will be the result of the next message in the client/server exchange.

- EVE_DHCP_REQUEST: A DHCPv4 Request or a DHCPv6 Request message is received

- EVE_DHCP_CONFIRM: A DHCPv6 Confirm message is received

- EVE_DHCP_REBOOT: A DHCPv4 Reboot message is received

- EVE_DHCP_SOLICIT_RC: A DHCPv6 Solicitation message with the Rapid Commit option is received

### 6.4.4. Table of State Machine

The main state transits are listed as follows. Note that not all the details are specified in the table and the diagram.

| State | Event | Action | Next State |
|---|---|---|---|
| NO_BIND | RQ/RC/CF/RE | Generate entry | INIT_BIND |
| INIT_BIND | RPL | Record lease time<br>(send leasequery if no lease) | BOUND |
| INIT_BIND | EVE_ENTRY_EXPIRE | Remove entry | NO_BIND |
| BOUND | RLS/DCL | Remove entry | NO_BIND |
| BOUND | EVE_ENTRY_EXPIRE | Remove entry | NO_BIND |
| BOUND | RPL | Set new lifetime | BOUND |
| BOUND | LQR | Record lease time | BOUND |

Figure 9: State Transition Table

RQ: EVE_DHCP_REQUEST
RC: EVE_DHCP_SOLICIT_RC
CF: EVE_DHCP_CONFIRM
RE: EVE_DHCP_REBOOT
RPL: EVE_DHCP_REPLY
RLS: EVE_DHCP_RELEASE
DCL: EVE_DHCP_DECLINE
LQR: EVE_DHCP_LEASEQUERY

```
                               +-------------+
                               |             |
                      /--------+   NO_BIND   |<--------\
                      |  ----->|             |         |
                      |  |     +-------------+         |EVE_DHCP_RELEASE
   EVE_DHCP_REQUEST   |  |                             |EVE_DHCP_DECLINE
   EVE_DHCP_CONFIRM   |  |EVE_ENTRY_EXPIRE             |EVE_ENTRY_EXPIRE
   EVE_DHCP_SOLICIT_RC|  |                             |
   EVE_DHCP_REBOOT    |  |                             |
                      |  |                             |
                      |  |                             |
                      v  |                             |
              +-------------+                       +------------+
              |             |    EVE_DHCP_REPLY   |            |
              |  INIT_BIND  --------------------->|    BOUND   |<-\
              |             |                     |            |  |
              +-------------+                     +------------+  |
                                                         |        |
                                                         \--------/
                                               EVE_DHCP_REPLY
                                               EVE_DHCP_LEASEQUERY
```

Figure 10: Diagram of Transit

## 7. Data Snooping Process

### 7.1. Scenario

The rationale of the DHCP Snooping Process specified in Section 6 is that if a DHCP client’s use of a DHCP address is legitimate, the corresponding DHCP address assignment procedure must have been finished during the attachment of the DHCP client. This is the case when the SAVI device is continuously on the path(s) from the DHCP client to the DHCP server(s)/relay(s). However, there are two cases in which this does not work:

- Multiple paths: there is more than one feasible link-layer path from the client to the DHCP server/relay, and the SAVI device is not on every one of them. The client may get its address through one of the paths that does not pass through the SAVI device, but packets from the client can travel on paths that pass through the SAVI device, such as when the path through the link-layer network changes. Because the SAVI device could not snoop the DHCP packet exchange procedure, the DHCP Snooping Process cannot set up the corresponding binding.

- Dynamic path: there is only one feasible link-layer path from the client to the DHCP server/relay, but the path is dynamic due to topology change (for example, some link becomes broken due to failure or some planned change) or link-layer path change. This situation also covers the local-link movement of clients without the address confirm/reconfiguration process. For example, a host changes its attached switch port in a very short time. In such cases, the DHCP Snooping Process will not set up the corresponding binding.

The Data Snooping Process can avoid the permanent blocking of legitimate traffic in case one of these two exceptions occurs. This process is performed on attachments with the Data-Snooping attribute. Data packets without a matching binding entry may trigger this process to set up bindings.

Snooping data traffic introduces a considerable burden on the processor and ASIC-to-Processor bandwidth of SAVI devices. Because of the overhead of this process, the implementation of this process is OPTIONAL. This function SHOULD be enabled unless the implementation is known to be used in the scenarios without the above exceptions. For example, if the implementation is to be used in networks with tree topology and without host local-link movement, there is no need to implement this process in such scenarios.

This process is not intended to set up a binding whenever a data packet without a matched binding entry is received. Instead, unmatched data packets trigger this process probabilistically, and generally a number of unmatched packets will be discarded before the binding is set up. The parameter(s) of this probabilistic process SHOULD be configurable, defaulting to a situation where data snooping is disabled.

## 7.2. Rationale

This process makes use of NS/ARP and DHCP Leasequery to set up bindings. If an address is not used by another client in the network, and the address has been assigned in the network, the address can be bound with the binding anchor of the attachment from which the unmatched packet is received.

The Data Snooping Process provides an alternative path for binding entries to reach the BOUND state in the exceptional cases explained in Section 7.1 when there are no DHCP messages that can be snooped by the SAVI device.

In some of the exceptional cases (especially the dynamic topology case), by the time the binding has reached the BOUND state, the DHCP messages may be passing through the SAVI device. In this case, the events driven by DHCP messages that are expected in the BOUND state in the DHCP Snooping Process may occur, and the binding can be handled by the DHCP Snooping Process state machine.

In any event, the lease expiry timeout event will occur even if no others do. This will cause the binding to be deleted and the state to logically return to NO_BIND state. Either the DHCP or the Data Snooping Process will be reinvoked if the lease is still in place. If DHCP messages are still not passing through the SAVI device, there will be a brief disconnection during which data packets passing through the SAVI device will be dropped. The probabilistic initiation of the Data Snooping Process can then take over again and return the binding state to BOUND in due course.

The security issues concerning this process are discussed in Section 11.1.

### 7.3. Additional Binding States Description

In addition to NO_BIND and BOUND from Section 6.2, three new states used in this process are listed here. The INIT_BIND state is not used, as it is entered by observing a DHCP message.

DETECTION: The address in the entry is undergoing local duplication detection.

RECOVERY: The SAVI device is querying the assignment and lease time of the address in the entry through DHCP Leasequery.

VERIFY: The SAVI device is verifying that the device connected to the attachment point has a hardware address that matches the one returned in the DHCP Leasequery.

Because the mechanisms used for the operations carried out while the binding is in these three states operate over unreliable protocols, each operation is carried out twice with a timeout that is triggered if no response is received.

### 7.4. Events

To handle the Data Snooping Process, six extra events, described here, are needed in addition to those used by the DHCP Snooping Process (see Section 6.3). If an event will trigger the creation of a new binding entry, the binding entry limit on the binding anchor MUST NOT be exceeded.

EVE_DATA_UNMATCH: A data packet without a matched binding is received.

EVE_DATA_CONFLICT: An ARP Reply / Neighbor Advertisement (NA) message against an address in the DETECTION state is received from a host other than the one for which the entry was added (i.e., a host attached at a point other than the one on which the triggering data packet was received).

EVE_DATA_LEASEQUERY:

- IPv4: A DHCPLEASEACTIVE message with the IP Address Lease Time option is received. Note that the DHCPLEASEUNKNOWN and DHCPLEASEUNASSIGNED replies are ignored.
- IPv6: A successful LEASEQUERY-REPLY is received.

EVE_DATA_VERIFY: An ARP Reply / NA message has been received in the VERIFY state from the device connected to the attachment point on which the data packet was received.

The triggering packet should pass the following checks to trigger a valid event:

- Attribute check: the data packet should be from attachments with the Data-Snooping attribute; the DHCPLEASEACTIVE/LEASEQUERY-REPLY messages should be from attachments with the DHCP-Snooping attribute.
- Binding limitation check: the data messages must not cause new binding setup on an attachment whose binding entry limitation has been reached (refer to Section 11.5).
- Address check: For EVE_DATA_LEASEQUERY, the source address of the DHCPLEASEQUERY messages must pass the check specified in Section 8.2. For EVE_DATA_CONFLICT and EVE_DATA_VERIFY, the source address and target address of the ARP or NA messages must pass the check specified in Section 8.2.
- Interval check: the interval between two successive EVE_DATA_UNMATCH events triggered by an attachment MUST be no smaller than DATA_SNOOPING_INTERVAL.
- TID check: the DHCPLEASEACTIVE/LEASEQUERY-REPLY messages must have a matched TID with the corresponding entry.
- Prefix check: the source address of the data packet should be of a valid local prefix, as specified in Section 7 of [RFC7039].

EVE_DATA_EXPIRE: A timer expires indicating that a response to a hardware address verification message sent in the VERIFY state has not been received within the specified DETECTION_TIMEOUT period.

EVE_ENTRY_EXPIRE: A timer expires after the Lifetime indicated in the relevant BST entry has elapsed. This is identical to the usage in the DHCP Snooping Process.

### 7.5. Message Sender Functions

The Data Snooping Process involves sending three different messages to other network devices. Each message may be sent up to two times since they are sent over unreliable transports and are sent in different states. The functions defined in this section specify the messages to be sent in the three cases. In each case, the message to be sent depends on whether the triggering data packet is an IPv4 or an IPv6 packet.

#### 7.5.1. Duplicate Detection Message Sender

Send a message to check if the source address in the data packet that triggered the Data Snooping Process has a local conflict (that is, it uses an address that is being used by another node):

IPv4 address: Broadcast an Address Resolution Protocol (ARP) Request [RFC826] or an ARP Probe [RFC5227] for the address to the local network. An ARP Response will be expected from the device on the attachment point on which the triggering data packet was received. An ARP Reply received on any other port indicates a duplicate address.

IPv6 address: Send a Duplicate Address Detection (DAD) message (Neighbor Solicitation message) to the solicited-node multicast address [RFC4861] targeting the address. Ideally, only the host on that point of attachment responds with a Neighbor Advertisement. A Neighbor Advertisement received on any other port indicates a duplicate address.

As both the ARP and DAD processes are unreliable (the packet either to or from the other system may be lost in transit; see [RFC6620]), if there is no response after the DETECTION_TIMEOUT, an EVE_ENTRY_EXPIRE is generated.

### 7.5.2. Leasequery Message Sender

Send a DHCPLEASEQUERY message to the DHCP server(s) to determine if it has given out a lease for the source address in the triggering data packet. A list of authorized DHCP servers is kept by the SAVI device. The list should be either preconfigured with the IPv4 and/or IPv6 addresses or dynamically discovered: For networks using IPv4, this can be done by sending DHCPv4 Discover messages and parsing the returned DHCPv4 Offer messages; for networks using IPv6, discovery can be done by sending DHCPv6 SOLICIT messages and parsing the returned ADVERTISE messages. The same TID should be used for all LEASEQUERY messages sent in response to a triggering data message on an attachment point. The TID is generated if the TID field in the BST entry is empty and recorded in the TID field of the BST entry when the first message is sent. Subsequent messages use the TID from the BST entry.

(1) IPv4 address: Send a DHCPLEASEQUERY [RFC4388] message querying by IP address to each DHCPv4 server in the list of authorized servers with an IP Address Lease Time option (option 51). If the server has a valid lease for the address, the requested information will be returned in a DHCPLEASEACTIVE message.

(2) IPv6 address: Send a LEASEQUERY [RFC5007] message querying by IP address to each DHCPv6 server in the list of authorized servers using the server address as the link-address in the LEASEQUERY message. If the server has a valid lease for the address, the requested information will be returned in a LEASEQUERY-REPLY message marked as successful (i.e., without an OPTION_STATUS_CODE in the reply). The IA Address option(s) returned contains any IPv6 addresses bound to the same link together with the lease validity time.

As DHCP Leasequeries are an unreliable process (the packet either to or from the server may be lost in transit), if there is no response after the MAX_LEASEQUERY_DELAY, an EVE_DATA_EXPIRE is generated. Note that multiple response messages may be received if the list of authorized servers contains more than one address of the appropriate type and, in the case of DHCPv6, the responses may contain additional addresses for which leases have been allocated.

### 7.5.3. Address Verification Message Sender

Send a message to verify that the link-layer address in the attached device that sent the triggering data packet matches the link-layer address contained in the leasequery response:

IPv4 address: Send an ARP Request with the Target Protocol Address set to the IP address in the BST entry. The ARP Request is only sent to the attachment that triggered the binding. If the attached device has the IP address bound to the interface attached to the SAVI device, an ARP Reply should be received containing the hardware address of the interface on the attached device that can be compared with the leasequery value.

IPv6 address: Send a Neighbor Solicitation (NS) message with the target address set to the IP address in the BST entry. The NS is only sent to the attachment that triggered the binding. If the attached device has the IP address bound to the interface attached to the SAVI device, an NA should be received indicating that the attached device has the IP address configured on the interface.

As both the ARP and NS/NA processes are unreliable (the packet either to or from the other system may be lost in transit; see [RFC6620]), if there is no response after the DETECTION_TIMEOUT, an EVE_DATA_EXPIRE is generated.

## 7.6. Initial State: NO_BIND

### 7.6.1. Event: EVE_DATA_UNMATCH: A data packet without a matched binding is received

Make a probabilistic determination as to whether to act on this event. The probability may be configured or calculated based on the state of the SAVI device. This probability should be low enough to mitigate the damage from DoS attacks against this process.

Create a new entry in the BST. Set the Binding Anchor field to the corresponding binding anchor of the attachment. Set the Address field to the source address of the packet.

Address conflicts MUST be detected and prevented.

If local address detection is performed:

Set the State field to DETECTION. Set the Lifetime of the created entry to DETECTION_TIMEOUT. Set the Timeouts field to 0. Start the detection of any local address conflicts by sending a Duplicate Address Detection Message (Section 7.5.1). Transition to DETECTION state.

If local address detection is not performed:

Set the State field to RECOVERY. Set the Lifetime of the created entry to LEASEQUERY_DELAY. Set the Timeouts field to 0. Start the recovery of any DHCP lease associated with the source IP address by sending one or more LEASEQUERY messages (Section 7.5.2). Transition to RECOVERY state.

The packet that triggers this event SHOULD be discarded.

An example of the BST entry during duplicate address detection is illustrated in Figure 11.

| Anchor | Address | State | Lifetime | TID | Timeouts |
|---|---|---|---|---|---|
| Port_1 | Addr1 | DETECTION | DETECTION_TIMEOUT | | 0 |

Figure 11: Binding Entry in BST on Data-Triggered Initialization

Resulting state: DETECTION - The address in the entry is undergoing local duplication detection - or RECOVERY - The DHCP lease(s) associated with the address is being queried.

### 7.6.2. Events Not Observed in NO_BIND for Data Snooping

EVE_DATA_CONFLICT: An ARP Reply / NA message is received from an unexpected system.

EVE_DATA_LEASEQUERY: A valid DHCPLEASEACTIVE or LEASEQUERY-REPLY is received.

EVE_DATA_VERIFY: A valid ARP Reply or NA message is received from the attached device.

All EVE_DHCP_* events defined in Section 6.3.2 are treated as described in the DHCP Snooping Process (Section 6.4.1) and may result in that process being triggered.

EVE_ENTRY_EXPIRE: Expiration of the DECTECTION_TIMEOUT

EVE_DATA_EXPIRE: Expiration of the DECTECTION_TIMEOUT

## 7.7. Initial State: DETECTION

### 7.7.1. Event: EVE_ENTRY_EXPIRE

When this event occurs, no address conflict has been detected during the previous DETECTION_TIMEOUT period.

If the Timeouts field in the BST entry is 0:

> Set the Lifetime of the BST entry to DETECTION_TIMEOUT. Set the Timeouts field to 1. Restart the detection of any local address conflicts by sending a second Duplicate Address Detection Message (Section 7.5.1). Remain in DETECTION state.

If the Timeouts field in the BST entry is 1:

> Assume that there is no local address conflict. Set the State field to RECOVERY. Set the Lifetime of the BST entry to LEASEQUERY_DELAY. Set the Timeouts field to 0. Start the recovery of any DHCP lease associated with the source IP address by sending one or more LEASEQUERY messages (Section 7.5.2). Transition to RECOVERY state.

An example of the entry is illustrated in Figure 12.

| Anchor | Address | State | Lifetime | TID | Timeouts |
|---|---|---|---|---|---|
| Port_1 | Addr1 | RECOVERY | MAX_LEASEQUERY_DELAY | TID | 0 |

Figure 12: Binding Entry in BST on Leasequery

Resulting state: DETECTION - If a second local conflict period is required - or RECOVERY - The SAVI device is querying the assignment and lease time of the address in the entry through DHCP Leasequery.

### 7.7.2. Event: EVE_DATA_CONFLICT: ARP Reply / NA Message Received from Unexpected System

Remove the entry.

Resulting state: NO_BIND - No binding has been set up.

### 7.7.3. Events Not Observed in DETECTION

EVE_DATA_UNMATCH: A data packet without a matched binding is received

All EVE_DHCP_* events defined in Section 6.3.2

EVE_DHCP_REBIND: A DHCPv4 Rebind or a DHCPv6 Rebind message is received

## 7.8. Initial State: RECOVERY

### 7.8.1. Event: EVE_DATA_LEASEQUERY: A valid DHCPLEASEACTIVE or successful LEASEQUERY-REPLY is received

Set the State in the BST entry to VERIFY. Depending on the type of triggering source IP address, process the received DHCP Leasequery response:

IPv4 address: Update the Lifetime field in the BST entry to the sum of the value encoded in the IP Address Lease Time option of the DHCPLEASEACTIVE message and MAX_DHCP_RESPONSE_TIME. Record the value of the "chaddr" field (hardware address) in the message for checking against the hardware address received during verification in the next state. Set the Timeouts field to 0. Start the verification process by sending an Address Verification Message (see Section 7.5.3). Transition to VERIFY state. Start an additional verification timer with a duration of DETECTION_TIMEOUT. When this expires, an EVE_DATA_EXPIRE event will be generated.

IPv6 address: Update the Lifetime field in the BST entry to the sum of the valid lifetime extracted from the OPTION_CLIENT_DATA option in the LEASEQUERY-REPLY message and MAX_DHCP_RESPONSE_TIME. Set the Timeouts field to 0. Start the verification process by sending an Address Verification Message (see Section 7.5.3). Transition to VERIFY state. Start an additional verification timer with a duration of DETECTION_TIMEOUT. When this expires, an EVE_DATA_EXPIRE event will be generated.

If multiple addresses are received in the LEASEQUERY-REPLY, new BST entries MUST be created for the additional addresses using the same binding anchor. The entries are created with state set to VERIFY and the other fields set as described in this section for the triggering source IP address. Also, start the verification process and start verification timers for each additional address.

Resulting state: VERIFY - Awaiting verification or otherwise of the association of the IP address with the connected interface.

### 7.8.2. Event: EVE_ENTRY_EXPIRE

Depending on the value of the Timeouts field in the BST entry, either send repeat LEASEQUERY messages or discard the binding:

If the Timeouts field in the BST entry is 0:

> No responses to the LEASEQUERY message(s) sent have been received during the first LEASEQUERY_DELAY period. Set the Lifetime of the BST entry to LEASEQUERY_DELAY. Set the Timeouts field to 1. Restart the recovery of any DHCP lease associated with the source IP address by sending one or more LEASEQUERY messages (Section 7.5.2). Remain in RECOVERY state.

If the Timeouts field in the BST entry is 1:

> No responses to the LEASEQUERY messages sent during two LEASEQUERY_DELAY periods were received. Assume that no leases exist and hence that the source IP address is bogus. Delete the BST entry. Transition to NO_BIND state.

Resulting state: RECOVERY - If repeat leasequeries are sent - or NO_BIND - If no successful responses to LEASEQUERY messages have been received.

### 7.8.3. Events Not Observed in RECOVERY

EVE_DATA_UNMATCH: A data packet without a matched binding is received

EVE_DATA_CONFLICT: An ARP Reply / NA message is received from an unexpected system

EVE_DATA_VERIFY: A valid ARP Reply or NA message is received from the attached device

All EVE_DHCP_* events defined in Section 6.3.2

EVE_DATA_EXPIRE: Expiration of the DECTECTION_TIMEOUT

## 7.9. Initial State: VERIFY

### 7.9.1. Event: EVE_DATA_LEASEQUERY: A valid DHCPLEASEACTIVE or successful LEASEQUERY-REPLY is received

If LEASEQUERY messages were sent to more than one DHCP server during RECOVERY state, additional successful leasequery responses may be received relating to the source IP address. The conflict resolution mechanisms specified in Section 6.8 of [RFC4388] and Section 4.3.4 of [RFC5007] can be used to determine the message from which values are used to update the BST Lifetime entry and the hardware address obtained from DHCP, as described in Section 7.8.1. In the case of DHCPv6 queries, the LEASEQUERY-REPLY may contain additional addresses as described in Section 7.8.1. If so, additional BST entries MUST be created or ones previously created updated as described in that section.

Resulting state: VERIFY (no change).

### 7.9.2. Event: EVE_DATA_VERIFY: A valid ARP Reply or NA is received from the device attached via the binding anchor

Depending on the type of triggering source IP address, this event may indicate that the device attached via the binding anchor in the BST entry is configured by DHCP using the IP address:

IPv4 address: Check that the value of the sender hardware address in the ARP Reply matches the saved "chaddr" field (hardware address) from the previously received DHCPLEASEACTIVE message. If not, ignore this event; a subsequent retry may provide verification. If the hardware addresses match, the binding entry has been verified.

IPv6 address: Simple receipt of a valid NA from the triggering source IP address at the binding anchor port provides verification for the binding entry.

If the binding entry has been verified, set the state in the BST entry to BOUND. Clear the TID field. Cancel the verification timer.

Resulting state: VERIFY (no change) - If the IPv4 DHCPLEASEQUERY "chaddr" address does not match the ARP Reply hardware address. Otherwise, the resulting state is BOUND.

### 7.9.3. Event: EVE_ENTRY_EXPIRE

The DHCP lease lifetime has expired before the entry could be verified. Remove the entry. Transition to NO_BIND state.

Resulting state: NO_BIND - No binding has been set up.

### 7.9.4. Event: EVE_DATA_EXPIRE

Depending on the value of the Timeouts field in the BST entry, either send a repeat validation message or discard the binding:

If the Timeouts field in the BST entry is 0:

> No response to the verification message sent has been received during the first DETECTION_TIMEOUT period. Set the Timeouts field to 1. Restart the verification process by sending an Address Verification Message (see Section 7.5.3). Start a verification timer with a duration of DETECTION_TIMEOUT. When this expires, an EVE_DATA_EXPIRE event will be generated. Remain in VERIFY state.

If the Timeouts field in the BST entry is 1:

> No responses to the verification messages sent during two DETECTION_TIMEOUT periods were received. Assume that the configuration of the triggering source IP address cannot be verified and hence that the source IP address is bogus. Delete the BST entry. Transition to NO_BIND state.

Resulting state: VERIFY - Additional verification message sent - or NO_BIND - No binding has been set up.

### 7.9.5. Events Not Observed in VERIFY

EVE_DATA_UNMATCH: A data packet without a matched binding is received

EVE_DATA_CONFLICT: An ARP Reply / NA message is received from an unexpected system

All EVE_DHCP_* events defined in Section 6.3.2

### 7.10. Initial State: BOUND

Upon entry to the BOUND state, control of the system continues as if a DHCP message assigning the address has been observed, as in Section 6.4.3. The BST entry has been restored.

Note that the TID field contains no value after the binding state changes to BOUND. The TID field is recovered from snooping DHCP Renew/Rebind messages if these are observed as described in the DHCP Snooping Process. Because TID is used to associate binding entries with messages from DHCP servers, it must be recovered or else a number of state transitions of this mechanism will not be executed normally.

## 7.11. Table of State Machine

The main state transitions are listed as follows.

| State | Event | Action | Next State |
|---|---|---|---|
| NO_BIND | EVE_DATA_UNMATCH | Start duplicate detect | DETECTION |
| DETECTION | EVE_ENTRY_EXPIRE 1 | Repeat duplicate detect | DETECTION |
| DETECTION | EVE_ENTRY_EXPIRE 2 | Start leasequery | RECOVERY |
| DETECTION | EVE_DATA_CONFLICT | Remove entry | NO_BIND |
| RECOVERY | EVE_ENTRY_EXPIRE 1 | Repeat leasequery | RECOVERY |
| RECOVERY | EVE_ENTRY_EXPIRE 2 | No lease found; remove entry | NO_BIND |
| RECOVERY | EVE_DATA_LEASEQUERY | Set lease time; start verify | VERIFY |
| VERIFY | EVE_ENTRY_EXPIRE | Lease expiry; remove entry | NO_BIND |
| VERIFY | EVE_DATA_LEASEQUERY | Resolve lease conflict(s) | VERIFY |
| VERIFY | EVE_DATA_VERIFY | Finish validation | BOUND or NO_BIND |
| VERIFY | EVE_DATA_EXPIRE 1 | Repeat verify | VERIFY |
| VERIFY | EVE_DATA_EXPIRE 2 | Verify failed; remove entry | NO_BIND |
| BOUND | EVE_ENTRY_EXPIRE | Lease expiry; remove entry | NO_BIND |
| BOUND | RENEW/REBIND | Record TID | BOUND |

Figure 13: State Transition Table

```
                               +-------------+         EVE_ENTRY_EXPIRE
                     /---------+             |<------------------------\
                     |         |   NO_BIND   |         EVE_DATA_EXPIRE |
    EVE_DATA_UNMATCH |  /----->|             |<----\   (2nd VRF_DELAY) |
                     |  |      +-------------+     |                   |
                     |  |         EVE_ENTRY_EXPIRE |                   |
                     |  |           (2nd LQ_DELAY) |                   |
   EVE_ENTRY_EXPIRE  |  |                          |  EVE_ENTRY_EXPIRE |
   (1st DAD_DELAY)   |  |                          |   (1st LQ_DELAY)  |
         /------\    |  |                          |        /--------\ |
         |      |    |  | EVE_DATA_CONFLICT        \---\    |        | |
         |      v    v  |                              |    v        | |
         |    +-------------+ EVE_ENTRY_EXPIRE       +------------+  | |
         |    |             | (2nd DAD_DELAY)        |            |  | |
         \----+  DETECTION  ------------------------>|  RECOVERY  +--/ |
              |             |                        |            |    |
              +-------------+   (To NO_BIND)         +------------+    |
                                ^                               |      |
                                |           EVE_DATA_LEASEQUERY |      |
                  /----------\  |                               |      |
                  |          |  | EVE_ENTRY_EXPIRE              |      |
    EVE_DHCP_RENEW|          v  |                               v      |
   EVE_DHCP_REBIND|    +-------------+                +-------------+  |
                  |    |             |                |             +--/
                  \----+   BOUND     |<---------------+   VERIFY    |
                       |             | EVE_DATA_VERIFY|             |<-\
                       +-------------+                +-------------+  |
                                                            |          |
                                                            \----------/
                                                     EVE_DATA_LEASEQUERY
                                                         EVE_DATA_EXPIRE
                                                         (1st VRF_DELAY)
```

Figure 14: Diagram of Transit

LQ_DELAY:  MAX_LEASEQUERY_DELAY
VRF_DELAY: DETECTION_TIMEOUT

## 8. Filtering Specification

This section specifies how to use bindings to filter out packets with spoofed source addresses.

Filtering policies are different for data packets and control packets. DHCP, ARP, and Neighbor Discovery Protocol (NDP) [RFC4861] messages are classified as control packets. All other packets are classified as data packets.

### 8.1. Data Packet Filtering

Data packets from attachments with the Validating attribute TRUE MUST have their source addresses validated. There is one exception to this rule.

A packet whose source IP address is a link-local address cannot be checked against DHCP assignments, as it is not assigned using DHCP. Note: as explained in Section 1, a SAVI solution for link-local addresses, e.g., FCFS SAVI [RFC6620], can be enabled to check packets with a link-local source address.

If the source IP address of a packet is not a link-local address, but there is not a matching entry in the BST with BOUND state, this packet MUST be discarded. However, the packet may trigger the Data Snooping Process (Section 7) if the Data-Snooping attribute is set on the attachment.

Data packets from an attachment with the Validating attribute set FALSE will be forwarded without having their source addresses validated.

The SAVI device MAY log packets that fail source address validation.

### 8.2. Control Packet Filtering

For attachments with the Validating attribute:

DHCPv4 Client-to-Server messages in which the source IP address is neither all zeros nor bound with the corresponding binding anchor in the BST MUST be discarded.

DHCPv6 Client-to-Server messages in which the source IP address is neither a link-local address nor bound with the corresponding binding anchor in the BST MUST be discarded.

NDP messages in which the source IP address is neither a link-local address nor bound with the corresponding binding anchor MUST be discarded.

NA messages in which the target address is neither a link-local address nor bound with the corresponding binding anchor MUST be discarded.

ARP messages in which the protocol is IP and the sender protocol address is neither all zeros nor bound with the corresponding binding anchor MUST be discarded.

ARP Reply messages in which the target protocol address is not bound with the corresponding binding anchor MUST be discarded.

For attachments with other attributes:

DHCP Server-to-Client messages not from attachments with the DHCP-Trust attribute or Trust attribute MUST be discarded.

For attachments with no attribute:

DHCP Server-to-Client messages from such attachments MUST be discarded.

The SAVI device MAY record any messages that are discarded.

## 9. State Restoration

If a SAVI device reboots, the information kept in volatile memory will be lost. This section specifies the restoration of attribute configuration and the BST.

### 9.1. Attribute Configuration Restoration

The loss of attribute configuration will not break the network: no action will be performed on traffic from attachments with no attribute. However, the loss of attribute configuration makes this SAVI function unable to work.

To avoid the loss of binding anchor attribute configuration, the configuration MUST be able to be stored in non-volatile storage. After the reboot of the SAVI device, if the configuration of binding anchor attributes is found in non-volatile storage, the configuration MUST be used.

### 9.2. Binding State Restoration

The loss of binding state will cause the SAVI devices to discard legitimate traffic. Simply using the Data Snooping Process to recover a large number of bindings is a heavy overhead and may cause considerable delay. Thus, recovering bindings from non-volatile storage, as specified below, is RECOMMENDED.

Binding entries MAY be saved into non-volatile storage whenever a new binding entry changes to BOUND state. If a binding with BOUND state is removed, the saved entry MUST be removed correspondingly. The time when each binding entry is established is also saved.

If the BST is stored in non-volatile storage, the SAVI device SHOULD restore binding state from the non-volatile storage immediately after reboot. Using the time when each binding entry was saved, the SAVI device should check whether the entry has become obsolete by comparing the saved lifetime and the difference between the current time and time when the binding entry was established. Obsolete entries that would have expired before the reboot MUST be removed.

## 10. Constants

The following constants are recommended for use in this context:

- MAX_DHCP_RESPONSE_TIME (120s): Maximum Solicit timeout value (SOL_MAX_RT from [RFC3315])
- MAX_LEASEQUERY_DELAY (10s): Maximum LEASEQUERY timeout value (LQ_MAX_RT from [RFC5007])
- DETECTION_TIMEOUT (0.5s): Maximum duration of a hardware address verification step in the VERIFY state (TENT_LT from [RFC6620])
- DATA_SNOOPING_INTERVAL: Minimum interval between two successive EVE_DATA_UNMATCH events triggered by an attachment. Recommended interval: 60s and configurable
- OFFLINK_DELAY: Period after a client is last detected before the binding anchor is being removed. Recommended delay: 30s

## 11. Security Considerations

### 11.1. Security Problems with the Data Snooping Process

There are two security problems with the Data Snooping Process (Section 7):

(1) The Data Snooping Process is costly, but an attacker can trigger it simply through sending a number of data packets. To avoid Denial-of-Service attacks against the SAVI device itself, the Data Snooping Process MUST be rate limited. A constant DATA_SNOOPING_INTERVAL is used to control the frequency. Two Data Snooping Processes on one attachment MUST be separated by a minimum interval time of DATA_SNOOPING_INTERVAL. If this value is changed, the value needs to be large enough to minimize DoS attacks.

(2) The Data Snooping Process may set up incorrect bindings if the clients do not reply to the detection probes (Section 7.6.1). An attack will pass the duplicate detection if the client

assigned the target address does not reply to the detection probes. The DHCP Leasequery procedure performed by the SAVI device just tells whether or not the address is assigned in the network. However, the SAVI device cannot determine whether the address is just assigned to the triggering attachment from the DHCPLEASEQUERY Reply.

### 11.2. Securing Leasequery Operations

In [RFC4388] and [RFC5007], the specific case of DHCP Leasequeries originated by "access concentrators" is addressed extensively. SAVI devices are very similar to access concentrators in that they snoop on DHCP traffic and seek to validate source addresses based on the results. Accordingly, the recommendations for securing leasequery operations for access concentrators in Section 7 of [RFC4388] and Section 5 of [RFC5007] MUST be followed when leasequeries are made from SAVI devices. [RFC5007] RECOMMENDS that communications between the querier and the DHCP server are protected with IPsec. It is pointed out that there are relatively few devices involved in a given administrative domain (SAVI devices, DHCP relays, and DHCP servers) so that manual configuration of keying material would not be overly burdensome.

### 11.3. Client Departure Issues

After a binding is set up, the corresponding client may leave its attachment point. It may depart temporarily due to signal fade or permanently by moving to a new attachment point or leaving the network. In the signal fade case, since the client may return shortly, the binding should be kept momentarily, lest legitimate traffic from the client be blocked. However, if the client leaves permanently, keeping the binding can be a security issue. If the binding anchor is a property of the attachment point rather than the client, e.g., the switch port but not incorporating the MAC address, an attacker using the same binding anchor can send packets using IP addresses assigned to the client. Even if the binding anchor is a property of the client, retaining binding state for a departed client for a long time is a waste of resources.

Whenever a direct client departs from the network, a link-down event associated with the binding anchor will be triggered. SAVI-DHCP monitors such events and performs the following mechanism.

(1) Whenever a client with the Validating attribute leaves, a timer of duration OFFLINK_DELAY is set on the corresponding binding entries.

(2) If a DAD Neighbor Solicitation / Gratuitous ARP request is received that targets the address during OFFLINK_DELAY, the entry MAY be removed.

(3) If the client returns on-link during OFFLINK_DELAY, cancel the timer.

In this way, the bindings of a departing client are kept for OFFLINK_DELAY. In cases of link flapping, the client will not be blocked. If the client leaves permanently, the bindings will be removed after OFFLINK_DELAY.

SAVI-DHCP does not handle the departure of indirect clients because it will not be notified of such events. Switches supporting indirect attachment (e.g., through a separate non-SAVI switch) SHOULD use information specific to the client such as its MAC address as part of the binding anchor.

### 11.4. Compatibility with Detecting Network Attachment (DNA)

DNA [RFC4436] [RFC6059] is designed to decrease the handover latency after reattachment to the same network. DNA mainly relies on performing a reachability test by sending unicast Neighbor Solicitation / Router Solicitation / ARP Request messages to determine whether a previously configured address is still valid.

Although DNA provides optimization for clients, there is insufficient information for this mechanism to migrate the previous binding or establish a new binding. If a binding is set up only by snooping the reachability test message, the binding may be invalid. For example, an attacker can perform the reachability test with an address bound to another client. If a binding is migrated to the attacker, the attacker can successfully obtain the binding from the victim. Because this mechanism wouldn’t set up a binding based on snooping the DNA procedure, it cannot achieve perfect compatibility with DNA. However, it only means the reconfiguration of the interface is slowed but not prevented. Details are discussed as follows.

In Simple DNAv6 [RFC6059], the probe is sent with the source address set to a link-local address, and such messages will not be discarded by the policy specified in Section 8.2. If a client is reattached to a previous network, the detection will be completed, and the address will be regarded as valid by the client. However, the candidate address is not contained in the probe. Thus, the binding cannot be recovered through snooping the probe. As the client will perform DHCP exchange at the same time, the binding will be recovered from the DHCP Snooping Process. The DHCP Request messages will not be filtered out in this case because they have link-local source addresses. Before the DHCP procedure is completed, packets will be filtered out by the SAVI device. In other words, if this SAVI function is enabled, Simple DNAv6 will not help reduce the handover latency. If the Data-Snooping attribute is configured on the new attachment of the client, the data-triggered procedure may reduce latency.

In DNAv4 [RFC4436], the ARP Probe will be discarded because an unbound address is used as the sender protocol address. As a result, the client will regard the address under detection as valid. However, the data traffic will be filtered. The DHCP Request message sent by the client will not be discarded because the source IP address field should be all zeros as required by [RFC2131]. Thus, if the address is still valid, the binding will be recovered from the DHCP Snooping Process.

### 11.5. Binding Number Limitation

A binding entry will consume certain high-speed memory resources. In general, a SAVI device can afford only a quite limited number of binding entries. In order to prevent an attacker from overloading the resources of the SAVI device, a binding entry limit is set on each attachment. The binding entry limit is the maximum number of bindings supported on each attachment with the Validating attribute. No new binding should be set up after the limit has been reached. If a DHCP Reply assigns more addresses than the remaining binding entry quota of each client, the message will be discarded and no binding will be set up.

### 11.6. Privacy Considerations

A SAVI device MUST delete binding anchor information as soon as possible (i.e., as soon as the state for a given address is back to NO_BIND), except where there is an identified reason why that information is likely to be involved in the detection, prevention, or tracing of actual source-address spoofing. Information about hosts that never spoof (probably the majority of hosts) SHOULD NOT be logged.

### 11.7. Fragmented DHCP Messages

This specification does not preclude reassembly of fragmented DHCP messages, but it also does not require it. If DHCP fragmentation proves to be an issue, the issue will need to be specified and addressed. (This topic is beyond the scope of this document.)

# 12. References

## 12.1. Normative References

[RFC826] 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, DOI 10.17487/RFC0826, November 1982, <http://www.rfc-editor.org/info/rfc826>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997, <http://www.rfc-editor.org/info/rfc2131>.

[RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <http://www.rfc-editor.org/info/rfc3315>.

[RFC4388] Woundy, R. and K. Kinnear, "Dynamic Host Configuration Protocol (DHCP) Leasequery", RFC 4388, DOI 10.17487/RFC4388, February 2006, <http://www.rfc-editor.org/info/rfc4388>.

[RFC4436] Aboba, B., Carlson, J., and S. Cheshire, "Detecting Network Attachment in IPv4 (DNAv4)", RFC 4436, DOI 10.17487/RFC4436, March 2006, <http://www.rfc-editor.org/info/rfc4436>.

[RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <http://www.rfc-editor.org/info/rfc4861>.

[RFC5007] Brzozowski, J., Kinnear, K., Volz, B., and S. Zeng, "DHCPv6 Leasequery", RFC 5007, DOI 10.17487/RFC5007, September 2007, <http://www.rfc-editor.org/info/rfc5007>.

[RFC5227] Cheshire, S., "IPv4 Address Conflict Detection", RFC 5227, DOI 10.17487/RFC5227, July 2008, <http://www.rfc-editor.org/info/rfc5227>.

[RFC6059] Krishnan, S. and G. Daley, "Simple Procedures for Detecting Network Attachment in IPv6", RFC 6059, DOI 10.17487/RFC6059, November 2010, <http://www.rfc-editor.org/info/rfc6059>.

[RFC6620] Nordmark, E., Bagnulo, M., and E. Levy-Abegnoli, "FCFS SAVI: First-Come, First-Served Source Address Validation Improvement for Locally Assigned IPv6 Addresses", RFC 6620, DOI 10.17487/RFC6620, May 2012, <http://www.rfc-editor.org/info/rfc6620>.

## 12.2. Informative References

[DHCPv6-SHIELD] Gont, F., Liu, W., and G. Van de Velde, "DHCPv6-Shield: Protecting Against Rogue DHCPv6 Servers", Work in Progress, draft-ietf-opsec-dhcpv6-shield-07, May 2015.

[RFC2827] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, DOI 10.17487/RFC2827, May 2000, <http://www.rfc-editor.org/info/rfc2827>.

[RFC3736] Droms, R., "Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6", RFC 3736, DOI 10.17487/RFC3736, April 2004, <http://www.rfc-editor.org/info/rfc3736>.

[RFC7039] Wu, J., Bi, J., Bagnulo, M., Baker, F., and C. Vogt, Ed., "Source Address Validation Improvement (SAVI) Framework", RFC 7039, DOI 10.17487/RFC7039, October 2013, <http://www.rfc-editor.org/info/rfc7039>.

[RFC7341] Sun, Q., Cui, Y., Siodelski, M., Krishnan, S., and I. Farrer, "DHCPv4-over-DHCPv6 (DHCP 4o6) Transport", RFC 7341, DOI 10.17487/RFC7341, August 2014, <http://www.rfc-editor.org/info/rfc7341>.

## Acknowledgments

Special thanks to Jean-Michel Combes, Christian Vogt, Joel M. Halpern, Eric Levy-Abegnoli, Marcelo Bagnulo Braun, Jari Arkko, Elwyn Davies, Barry Leiba, Ted Lemon, Leaf Yeh, Ralph Droms, and Alberto Garcia for careful review and evaluation comments on the mechanism and text.

Thanks to Mark Williams, Erik Nordmark, Mikael Abrahamsson, David Harrington, Pekka Savola, Xing Li, Lixia Zhang, Bingyang Liu, Duanqi Zhou, Robert Raszuk, Greg Daley, John Kaippallimalil, and Tao Lin for their valuable contributions.

## Authors' Addresses

Jun Bi
Network Research Center, Tsinghua University
Beijing  100084
China

EMail: junbi@tsinghua.edu.cn

Jianping Wu
Dept. of Computer Science, Tsinghua University
Beijing  100084
China

EMail: jianping@cernet.edu.cn

Guang Yao
Network Research Center, Tsinghua University
Beijing  100084
China

EMail: yaoguang@cernet.edu.cn

Fred Baker
Cisco Systems
Santa Barbara, CA  93117
United States

EMail: fred@cisco.com