SAVI
Internet-Draft
Intended status: Standards Track
Expires: January 7, 2013

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

July 6, 2012

SAVI Solution for DHCP
draft-ietf-savi-dhcp-13

Abstract

This document specifies the procedure for creating bindings between a DHCPv4/DHCPv6 assigned source IP address and a binding anchor on SAVI (Source Address Validation Improvements) device. The bindings can be used to filter packets generated on the local link with forged source IP address in DHCP scenario. This mechanism is proposed as a complement of ingress filtering to provide finer granularity source IP address validation.

Status of this Memo

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

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

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

This Internet-Draft will expire on January 7, 2013.

Copyright Notice

Copyright (c) 2012 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

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

Table of Contents

1. Introduction 5
2. Requirements Language 5
3. Terminology 5
4. SAVI-DHCP Scenario 6
5. Binding Anchor Attributes 7
5.1. No Attribute 7
5.2. SAVI-Validation Attribute 8
5.3. SAVI-DHCP-Trust Attribute 8
5.4. SAVI-SAVI Attribute 8
5.5. SAVI-BindRecovery Attribute 9
5.6. Table of Mutual Exclusions 9
6. Data Structures 10
6.1. Binding State Table (BST) 10
6.2. Mapping Table from Link Layer Address to Binding Anchor . 11
7. Procedure of Regular Binding Set Up 11
7.1. Rationale 11
7.2. Binding States Description 12
7.3. Events 12
7.3.1. Timer Expiration Event 12
7.3.2. Control Message Arriving Events 12
7.4. State Machine of DHCP Packet Snooping 13
7.4.1. From NO_BIND to Other States 13
7.4.1.1. Trigger Event 13
7.4.1.2. Following Actions 13
7.4.2. From INIT_BIND to Other States 15
7.4.2.1. Trigger Event 15
7.4.2.2. Following Actions 15
7.4.3. From BOUND to Other States 16
7.4.3.1. Trigger Event 16
7.4.3.2. Following Actions 17
7.5. Table of State Machine 17
8. Supplemental Binding Process 18
8.1. Rationale 19
8.2. Additional Binding States Description 19
8.3. Events 19
8.4. State Machine of Binding Recovery Process 19
8.4.1. From NO_BIND to Other States 19
8.4.1.1. Trigger Event 19
8.4.1.2. Following Action 20
8.4.2. From DETECTION to Other States 20
8.4.2.1. Trigger Event 20
8.4.2.2. Following Action 21
8.4.3. From RECOVERY to Other States 21
8.4.3.1. Trigger Event 21
8.4.3.2. Following Action 21
8.4.4. After BOUND 22

8.4.4.1. Trigger Event 22
8.4.4.2. Following Action 22
9. Filtering Specification 22
9.1. Data Packet Filtering 22
9.2. Control Packet Filtering 23
10. State Restoration 23
10.1. Binding Anchor Attribute Restoration 23
10.2. Binding State Restoration 24
11. Constants 24
12. MLD Considerations 24
13. Security Considerations 24
13.1. Security Problem about Binding Triggered by EVE_DHCP_REPLY_NULL 24
13.2. Binding Number Limitation 25
13.3. Risk from Link Layer Routing Dynamic 26
13.4. Duplicate Bindings of Same Address 26
13.5. Security Problems about Binding Recovery Process 26
13.6. Compatibility with DNA (Detecting Network Attachment) . . 27
13.7. Bogus DHCP Server Threat 28
13.8. Handle Binding Anchor Off-link Event 28
13.9. Residual Threats 28
14. IANA Considerations 29
15. Acknowledgment 29
16. References 30
16.1. Informative References 30
16.2. Normative References 30
Appendix A. change log 31
Authors' Addresses 32

1. Introduction

This document describes the procedure for creating a binding between an address allocated to a network attachment point by DHCP and a suitable binding anchor on a SAVI device. Binding anchor is defined to be a link layer property of network attachment in [savi-framework]. A list of proper binding anchors can be found in the Section 3.2 of [savi-framework]. The bindings can be used to filter or identify packets with forged source IP address. Section 9 suggests usage of these bindings to filter spoofing traffic in common practice.

The mechanism specified in this document is designed to provide a finer granularity source IP address validation, as a supplement to [BCP38]. This mechanism mainly performs DHCP snooping to set up bindings between IP addresses assigned by DHCP and corresponding binding anchors. The binding process is inspired by the work of [BA2007]. This mechanism covers DHCPv6 and binding recovery procedure, which are not discussed in [BA2007].

This solution is primarily designed for a pure DHCP scenario in which only DHCP addresses are legitimate global addresses. And it is designed for stateful DHCP scenario [rfc2131], [rfc3315]. In stateless DHCP scenarios [rfc3736], a node must have obtained its IPv6 addresses through some other mechanisms and so the address of the client SHOULD be bound based on other SAVI solutions.

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 RFC 2119 [rfc2119].

3. Terminology

SAVI-DHCP: The name of this SAVI instance for address assigned through DHCP

SAVI device: A network device which enables this SAVI instance

Non-SAVI device: A network device without this SAVI instance

DHCP server/relay type message: Messages that can only originate from DHCP server or relay. The list of such messages contains DHCPv4 ACK, DHCPv4 NAK, DHCPv4 OFFER, DHCPv6 Reply, DHCPv6 Advertise, DHCPv6 Reconfiguration, DHCPv6 Relay-forward and DHCPv6 Relay-reply

Lease time: Lease time in IPv4 [rfc2131] and valid lifetime in IPv6 [rfc3315]

4. SAVI-DHCP Scenario

Figure 1 shows the main elements in a network where this mechanism is deployed. One or more DHCP servers mediate the allocation and distribution of IP addresses to hosts requesting them using the DHCP protocol. DHCP relay may be used to relay message between client and server. Multiple SAVI devices and non-SAVI devices can co-exist on link. A SAVI device can be connected to DHCP client, DHCP relay (even DHCP server), SAVI device and non-SAVI device. The attribute of a binding anchor is determined by the role to which it is connected, as specified in Section 5.

Protection perimeter (refer to Section 4 in [savi-framework]) is formed by SAVI Device A and SAVI Device B for scalability. There is no need to setup a binding for client A on SAVI device B, or to setup a binding for client B on SAVI device A.

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

Note that in an IPv6 environment, every interface has a link-local address, which is not assigned by DHCP. This solution will not validate the link-local address. It is RECOMMENDED to enable a SAVI solution for link-local addresses, e.g. [savi-fcfs].

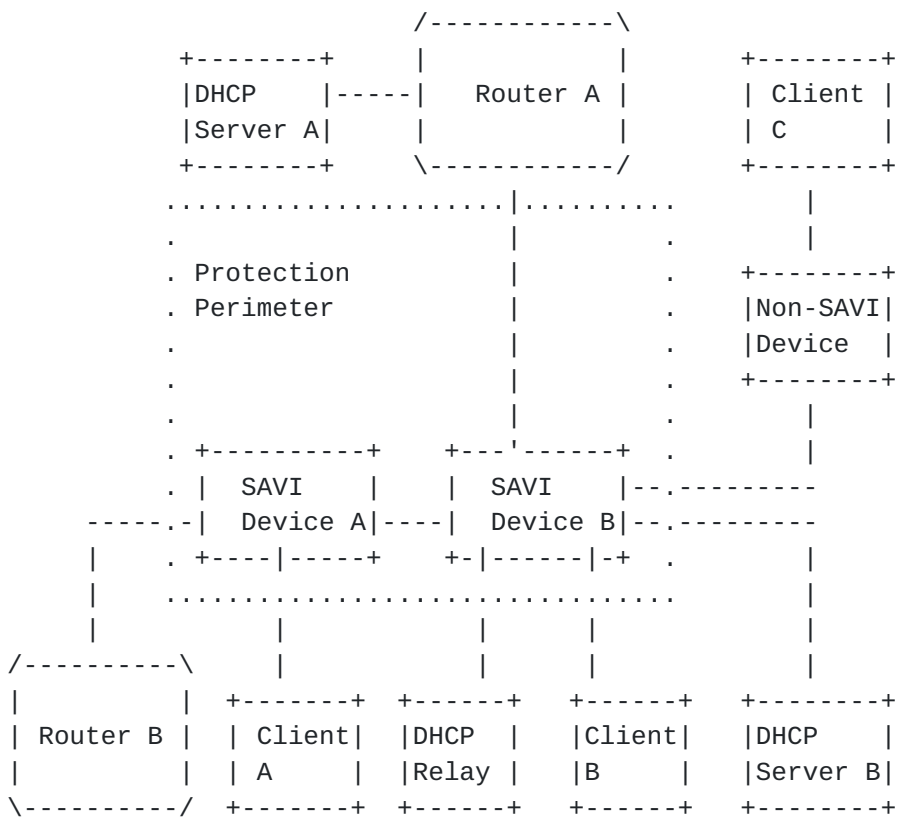


Figure 1: SAVI-DHCP Scenario

5. Binding Anchor Attributes

This section specifies the binding anchor attributes used by this mechanism. Attributes are used to distinguish different types of binding anchors. The procedure performed on each binding anchor is determined by the attributes set on the binding anchor.

By default, a binding anchor has no attribute. A binding anchor MAY be configured to have one or more compatible attributes. The attribute of each binding anchor should be manually configured before enabling this SAVI-DHCP function.

5.1. No Attribute

Before configuration, by default, a binding anchor has no attribute. Generally, each binding anchor is configured to have one or more attributes after configuration.

Considering the trivial of configuration, it is expected SAVI-DHCP

function will not cause a break of the network even though an attribute is not configured. Thus, data packet filtering will not be performed on binding anchor with no attribute. Note that if a binding anchor has no attribute, it means SAVI-DHCP-Trust is not configured on the binding anchor; thus, server type DHCP messages from the binding anchor with no attribute will be discarded.

However, a binding anchor MAY have no attribute after configuration. For example, in Figure 1, on the binding anchor of SAVI Device A connected to Router B, it is unreasonable to set up bindings for the host behind Router B, or filter out traffic from Router B, or allow forged DHCP message from Router B. Thus, no attribute should be configured on this binding anchor.

5.2. SAVI-Validation Attribute

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

5.3. SAVI-DHCP-Trust Attribute

SAVI-DHCP-Trust Attribute is used on binding anchor on the path to a trustable DHCP server/relay. DHCP server/relay type message coming from binding anchor with this attribute will be forwarded. In Figure 1, the binding anchor between SAVI Device B and DHCP Relay, the binding anchor between SAVI Device B and DHCP Server B, and the binding anchor between SAVI Device B and Router A should be configured to have this attribute.

Note that if the binding anchor is not exclusive for the valid DHCP server, messages from a valid DHCP server and a bogus DHCP server may arrive with the same binding anchor with this attribute. Thus, only relying on configuring this attribute may not protect the network from bogus DHCP servers. In deployment, it is RECOMMENDED the binding anchor for any DHCP server should be exclusive to the DHCP server on the protection perimeter. The security problem related with the bogus DHCP server and deployment options are discussed in section 14.7.

5.4. SAVI-SAVI Attribute

This attribute is used on the binding anchor from which the data traffic is not to be checked. Binding will not be set up on binding anchor with this attribute. All data packets will be allowed

directly. In Figure 1, the binding anchor between SAVI Device A and SAVI Device B should be configured with this attribute. DHCP server/ relay type messages from a binding anchor with this attribute but without SAVI-DHCP-Trust attribute will be filtered out.

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

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

5.5. SAVI-BindRecovery Attribute

If SAVI-Validation attribute is configured on a binding anchor, the binding on this attribute is set up primarily based on DHCP message snooping described in Section 7. However, in some scenarios, a DHCP address may be used without previous DHCP exchange procedure performed on the binding anchor, as discussed in Section 8. In such scenarios, data-triggered binding procedure, which is described in Section 8, is required to be performed.

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

5.6. Table of Mutual Exclusions

Mutually exclusive attributes MUST NOT be set on the same binding anchor. The compatibility of different attributes is listed in Figure 2.

	SAVI-Validation	SAVI-DHCP-Trust	SAVI-SAVI	SAVI-BindRecovery
SAVI-Validation	-	compatible	mutually exclusive	compatible
SAVI-DHCP-Trust	compatible	-	compatible	compatible
SAVI-SAVI	mutually exclusive	compatible	-	mutually exclusive
SAVI-Bind Recovery	compatible	compatible	mutually exclusive	-

Figure 2: Table of Mutual Exclusions

6. Data Structures

This section describes the data structures used in this mechanism. The main data structure, named Binding State Table, is used to record bindings and their states. A mapping table from the link layer address to the binding anchor may be required, as described in Section 13.7

6.1. Binding State Table (BST)

This table contains the state of a binding between a source address and a binding anchor. Entries are keyed on the binding anchor and source IP address. Each entry has a lifetime field recording the remaining lifetime of the entry, a state field recording the state of the binding and a TID field recording Transaction ID (TID) [rfc2131] [rfc3315] of DHCP message. The lifetime field is used to help remove expired bindings. The state field changes over time to identify the current state of the binding. States of bindings are specified in Section 7.2 and Section 8.2. The TID field is used to keep the TID in DHCP request. The TID field can be cleared after the state is changed to BOUND. An instance of this table is shown in Figure 3.

Anchor	Address	State	Lifetime	TID
A	IP_1	BOUND	65535	TID_1
A	IP_2	BOUND	10000	TID_2
B	IP_3	INIT_BIND	1	TID_3

Figure 3: Instance of BST

6.2. Mapping Table from Link Layer Address to Binding Anchor

As described in Section 7.4.2.2, whenever binding anchors must be recovered from DCHP Reply, such a mapping table is required. This table maps link layer address to binding anchor, so that the SAVI device can determine on which binding anchor to set up a binding only based on a DHCP Reply message.

Such a table can already exist on SAVI devices. For example, if the binding anchor is a switch port, the mapping table from MAC address to switch port is required for switching frames. We don!_t require SAVI devices to set up a different mapping table from the existing ones. Instead, the SAVI device MUST only use the existing one.

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

7. Procedure of Regular Binding Set Up

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

7.1. Rationale

The rationale of this mechanism is that if a node attaching to a binding anchor is legitimate to use a DHCP address, the DHCP procedure which assigns the address to the node must have been performed on the same binding anchor. This basis stands when the link layer routing is stable. However, layer-2 mobility and unstable link layer routing may result in that data packet is received from a different binding anchor. Infrequent link layer path change can be

handled (but not perfectly) by the mechanism described in Section 8. Section 13.3 discusses the situation that the link layer routing is naturally unstable. A solution for this issue is outside the scope of this document.

7.2. Binding States Description

This section describes the binding states of this mechanism.

NO_BIND: The state before a binding has been set up.

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

BOUND: The address is authorized to the client.

7.3. Events

7.3.1. Timer Expiration Event

EVE_ENTRY_EXPIRE: The lifetime of an entry expires.

7.3.2. Control Message Arriving Events

Only if a control message can pass the check in Section 9.2, the corresponding event is a valid event. For any message that may trigger a new binding, the binding entry limit (cf. Section 13.2) on the corresponding binding anchor MUST NOT have not been reached.

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

EVE_DHCP_CONFIRM: A DHCPv6 Confirm message is received from a binding anchor with SAVI-Validation attribute.

EVE_DHCP_REBIND: A DHCPv6 Rebind message is received from a binding anchor with SAVI-Validation attribute.

EVE_DHCP_OPTION_RC: A DHCPv6 Solicitation message with Rapid Commit option is received from a binding anchor with SAVI-Validation attribute.

EVE_DHCP_REPLY_LEASE: A DHCPv4 Acknowledgement or DHCPv6 Reply message is received, and there is an entry with matched TID in the BST, and lease time is contained in the message.

EVE_DHCP_REPLY_NOLEASE: A DHCPv4 Acknowledgement or DHCPv6 Reply

message is received, and there is an entry with matched TID in the BST, and no lease time is contained in the message.

EVE_DHCP_REPLY_NULL: A DHCPv4 Acknowledgement or DHCPv6 Reply message is received, and there is no entry in the BST contains the same TID as the message.

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

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

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

7.4. State Machine of DHCP Packet Snooping

7.4.1. From NO_BIND to Other States

7.4.1.1. Trigger Event

EVE_DHCP_REQUEST, EVE_DHCP_OPTION_RC, EVE_DHCP_CONFIRM, EVE_DHCP_REBIND, EVE_DHCP_REPLY_NULL.

7.4.1.2. Following Actions

If the triggering event is EVE_DHCP_REQUEST/EVE_DHCP_OPTION_RC:

The SAVI device MUST forward the message.

As illustrated in Figure 4, the SAVI device MUST generate an entry for the binding anchor in the Binding State Table (BST) and set the state field to INIT_BIND. The lifetime of this entry is set to be MAX_DHCP_RESPONSE_TIME. The TID field of the triggering message MUST be recorded in the entry.

The TID is stored because it will be used to correctly associate message from the DHCP server with target binding anchor.

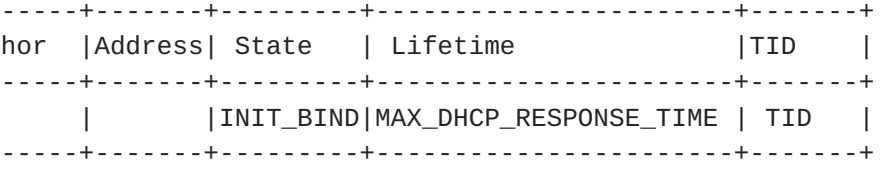


Figure 4: Binding entry in BST on Request/Rapid Commit triggered initialization

If the triggering event is EVE_DHCP_CONFIRM/EVE_DHCP_REBIND:

Besides forwarding the message and generating corresponding entry, the address to confirm/rebind MUST be recorded in the entry, as illustrated in Figure 5. The Lifetime field is set to MAX_DHCP_RESPONSE_TIME.

Anchor	Address	State	Lifetime	TID
A	Addr	INIT_BIND	MAX_DHCP_RESPONSE_TIME	TID

Figure 5: Binding entry in BST on Confirm/Rebind triggered initialization

If the triggering event is EVE_DHCP_REPLY_NULL:

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

Else:

The SAVI device MUST generate as many new entries in BST as the number of IADDR found in the message. If the binding anchor type inside the BST is not a link layer address, the binding anchor for an entry is recovered from the mapping table from the link layer address to the binding anchor (cf. Section 6.2) based on the destination link layer address inside the DHCP message.

As illustrated in Figure 6, the states of the corresponding entries are set to be BOUND. The lifetimes of the entries are set to be the lease time in the message.

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

If the binding anchor is a switch port, there can be a vulnerability

in this process which is discussed in Section 13.1. Similar problems can happen with other binding anchors.

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

Figure 6: Binding entry in BST on Reply triggered initialization

7.4.2. From INIT_BIND to Other States

7.4.2.1. Trigger Event

EVE_DHCP_REPLY_LEASE, EVE_LEASEQUERY_REPLY, EVE_ENTRY_EXPIRE.

7.4.2.2. Following Actions

If the trigger event is EVE_DHCP_REPLY_LEASE:

As illustrated in Figure 7, If the Address field is null, the lifetime field of the entry with matched TID is set to the sum of the lease time in Reply message and MAX_DHCP_RESPONSE_TIME. The state of the entry is changed to be BOUND. If more than one IADDR is found in the message and there is enough binding entry resources, corresponding new entries MUST be generated even when the binding number limit is exceeded. If there is not enough resources left, the message MUST be discarded.

Anchor	Address	State	Lifetime	TID
A	Addr	BOUND	Lease time+ MAX_DHCP_RESPONSE_TIME	TID

Figure 7: From INIT_BIND to BOUND with Lease Time

If the trigger event is EVE_DHCP_REPLY_NOLEASE:

The triggering message is in response to a Confirm message. If the message doesn!_t contain Status Code Success, discard the message. If the message is of Status Code Success, the state of the entry is changed to be BOUND. Because no lease time will be contained in the REPLY from DHCP server, the SAVI device MUST lookup in BST to determine whether there is an entry with the same address and TID. If there is such an entry and the corresponding binding anchor is off-link, it can be a local movement and the lifetime can be recovered from the entry. In this case, set the Lifetime to be the remaining value of Lifetime field of the existing entry, and remove the existing entry. If there is no such an entry, the SAVI device MUST send a LEASEQUERY [RFC5007] message querying by IP address to All_DHCP_Relay_Agents_and_Servers multicast address [RFC3315] or a configured server address. In this case, set the Lifetime of corresponding entry to MAX_LEASEQUERY_DELAY, as illustrated in Figure 8.

Anchor	Address	State	Lifetime	TID
A	Addr	BOUND	MAX_LEASEQUERY_DELAY	TID

Figure 8: From INIT_BIND to BOUND without Lease Time

If the trigger event is EVE_ENTRY_EXPIRE:

The entry MUST be deleted from BST.

7.4.3. From BOUND to Other States

7.4.3.1. Trigger Event

EVE_ENTRY_EXPIRE, EVE_DHCP_RELEASE, EVE_DHCP_DECLINE, EVE_DHCP_REPLY_LEASE, EVE_LEASEQUERY_REPLY.

7.4.3.2. Following Actions

If the trigger event is EVE_ENTRY_EXPIRE:

Remove the corresponding entry in BST.

If the trigger event is EVE_DHCP_RELEASE/EVE_DHCP_DECLINE:

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

If the trigger event is EVE_DHCP_REPLY_LEASE:

Set the lifetime of the entry with the corresponding address and TID to be the sum of the new lease time and MAX_DHCP_RESPONSE_TIME.

If the trigger event is EVE_LEASEQUERY_REPLY:

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

7.5. Table of State Machine

The main state transits are listed as follows.

State	Event	Action	Next State
NO_BIND	REQ/RC/RB/CFM	Generate entry	INIT_BIND
*NO_BIND	RPL_NULL	Generate entry with lease	BOUND
INIT_BIND	RPLL	Record lease time	BOUND
INIT_BIND	RPLN	Send Leasequery/set LQ_DLY	BOUND
INIT_BIND	Timeout	Remove entry	NO_BIND
BOUND	RLS/DCL	Remove entry	NO_BIND
BOUND	Timeout	Remove entry	NO_BIND
BOUND	RPLL	Set new lifetime	BOUND
BOUND	LQR	Record lease time	BOUND

Figure 9: Table of Transit

*: optional but NOT RECOMMENDED.

REQ: EVE_DHCP_REQUEST

CFM: EVE_DHCP_CONFIRM

RC: EVE_DHCP_OPTION_RC

RB: EVE_DHCP_REBIND

RPLL: EVE_DHCP REPLY_LEASE

RPLN: EVE_DHCP REPLY_NOLEASE

RPL_NULL: EVE_DHCP REPLY_NULL

DCL: DHCP DECLINE

RLS: DHCP RELEASE

LQR: EVE_LEASEQUERY_REPLY

Timeout: EVE_ENTRY_EXPIRE

LQ_DLY: MAX_LEASEQUERY_DELAY

8. Supplemental Binding Process

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

The binding recovery process is designed to avoid permanently blocking legitimate traffic. This process is performed on binding anchor with both SAVI-Validation and SAVI-BindRecovery attributes. It is not supposed to set up a binding whenever a data packet with an unbound source address is received. Generally, longer time and more packets are needed to trigger supplemental binding processes.

Considering the overhead of this procedure, the implementation of binding recovery process is a conditional SHOULD. This function SHOULD be implemented unless the implementation is known to be directly attached to the host. If an implementation is directly attached to host, change in link topology will not affect the bindings, and host will always start the re-configuration process after the interface is re-connected. Thus, there is no need to use additional processes to recovery bindings. If the mechanism is not implemented and managed hosts are not directly attached, legitimate traffic will be blocked until the node is reconfigured.

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

8.1. Rationale

If a DHCP address is allocated on the link, and the address is not used by another node in the network, the address can be bound with the binding anchor on which a message is received.

8.2. Additional Binding States Description

In addition to Section 7.2, new states used in this process are required here:

DETECTION The address is under local detection.

RECOVERY The address is in binding recovery process.

8.3. Events

Additional events in this process are described here. Also, if an event will trigger to set up a new binding entry, the binding entry limit on the binding anchor MUST NOT have not been reached.

EVE_BR_UNMATCH A data packet without matched binding of BOUND state in BST is received on a binding anchor with both SAVI-Validation and SAVI-BindRecovery attributes.

EVE_BR_CONFLICT Response against an address in DETECTION state is received.

EVE_BR_LEASEQUERY IPv4: a DHCPLEASEACTIVE message with IP Address Lease Time option is received from a binding anchor with SAVI-DHCP-Trust attribute; IPv6: a successful LEASEQUERY-REPLY is received.

8.4. State Machine of Binding Recovery Process

Through using additional states, the state machine of this process doesn!_t conflict the regular process described in Section 7. Thus, it can be implemented separately without changing the state machine in Section 7.

8.4.1. From NO_BIND to Other States

8.4.1.1. Trigger Event

EVE_BR_UNMATCH.

8.4.1.2. Following Action

Determine whether to process this event with a probability. The probability can be configured or calculated based on the state of the SAVI device. This probability should be low enough to mitigate the damage from DoS attack against this process. How to generate this probability is out of the scope of this document.

If there is a corresponding binding in BST on another binding anchor with the same source address, and the binding anchor is not off-link, the packet SHOULD be dropped, and no further actions will not be taken.

Check if the time since the last EVE_BR_UNMATCH on the same binding anchor is larger than BIND_RECOVERY_INTERVAL. No further actions will be taken if the last EVE_BR_UNMATCH on the same binding anchor in the last BIND_RECOVERY_INTERVAL.

Create a new entry in the BST. Set the Binding Anchor field to the corresponding binding anchor. Set the Address field to be source address of the packet. Set the state field to DETECTION. Set the lifetime of the created entry to 2*DAD_TIMEOUT.

Check if the address has a local conflict (it violates an address being used by another node) through:

(1) IPv4 address: sending two Address Resolution Protocol (ARP) Requests [rfc826]or two ARP probes [rfc5227] on the address;

(2) IPv6 address: performing Duplicate Address Detection (DAD) [rfc4862] twice on the address.

The delay between the two messages is DAD_TIMEOUT. The detection is performed twice to reduce the probability that the detection doesn!_t reach targeting node in case of packet loss.

The messages MUST NOT be sent to the link with the binding anchor of the triggering packet.

The packet which triggers this event SHOULD be discarded.

8.4.2. From DETECTION to Other States

8.4.2.1. Trigger Event

EVE_ENTRY_EXPIRE, EVE_BR_CONFLICT.

8.4.2.2. Following Action

If the trigger event is EVE_ENTRY_EXPIRE:

(1) IPv4 address: Send a DHCPLEASEQUERY [rfc4388] message querying by IP address to all DHCPv4 servers with IP Address Lease Time option (option 51). The server addresses can be found through DHCPv4 Discovery or from configuration. Change the state of the corresponding entry to RECOVERY. Change the lifetime of the entry to be MAX_LEASEQUERY_DELAY.

(2) IPv6 address: Send a LEASEQUERY [rfc5007] message querying by IP address to All_DHCP_Relay_Agents_and_Servers multicast address or a configured server address.

Change the state of the corresponding entry to RECOVERY. Change the lifetime of the entry to be MAX_LEASEQUERY_DELAY.

If the trigger event is EVE_BR_CONFLICT:

Remove the entry.

8.4.3. From RECOVERY to Other States

8.4.3.1. Trigger Event

EVE_ENTRY_EXPIRE, EVE_BR_LEASEQUERY.

8.4.3.2. Following Action

If the trigger event is EVE_BR_LEASEQUERY:

(1) IPv4 address: Change the state of the corresponding binding to BOUND. Set life time to the sum of the value encoded in IP Address Lease Time option of the DHCPLEASEACTIVE message and MAX_DHCP_RESPONSE_TIME.

(2) IPv6 address: Change the state of the corresponding binding to BOUND. Set the lifetime to the sum of the valid lifetime extracted from OPTION_CLIENT_DATA option in the LEASEQUERY-REPLY message and MAX_DHCP_RESPONSE_TIME.

If multiple addresses are specified in the LEASEQUERY-REPLY message, new entries MUST also be created correspondingly on the same binding anchor.

If the trigger event is EVE_ENTRY_EXPIRE:

Remove the entry.

8.4.4. After BOUND

Note that the TID field contains no value after the binding state changes to BOUND. Because TID is used to associate entry with message from server, attaching node will be prevented from renewing the bound address. Thus, the TID field is recovered from snooping DHCP Renew message.

8.4.4.1. Trigger Event

EVE_DHCP_RENEW.

8.4.4.2. Following Action

Set the TID field of the corresponding entry to the TID in the trigger message.

9. Filtering Specification

This section specifies how to use bindings to filter packets.

Filtering policies are different for data packet and control packet. DHCP and NDP (Neighbor Discovery Protocol) [rfc4861] messages that may cause state transit are classified into control packet. Neighbor Advertisement (NA) and ARP Response are also included in control packet, because the Target Address of NA and ARP Response should be checked to prevent spoofing. All other packets are considered to be data packets.

9.1. Data Packet Filtering

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

Packet whose source IP address is a link-local address SHOULD be forwarded.

If the source IP address of a packet is not a link-local address, but the address is not bound with the corresponding binding anchor, this packet MUST be discarded.

The SAVI device MAY record any violation.

9.2. Control Packet Filtering

For binding anchors with SAVI-Validation attribute:

Discard DHCPv4 REQUEST message whose source IP address is neither all zeros nor a bound address in BST.

Discard DHCPv6 Request message whose source is neither a link-local address nor bound with the corresponding binding anchor in BST.

Discard NDP messages whose source address is neither a link-local address nor bound with the corresponding binding anchor. In addition, discard NA message whose target address is neither a link-local address nor bound with the corresponding binding anchor.

Discard ARP messages whose protocol is IP and sender protocol address is neither all zeros address nor bound with the corresponding binding anchor. In addition, discard ARP Reply messages whose target address is not bound with the corresponding binding anchor.

For other binding anchors:

Discard DHCP server/relay type message not from binding anchor with the SAVI-DHCP-Trust attribute or SAVI-SAVI attribute.

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

10. State Restoration

If a SAVI device reboots accidentally or designedly, the information kept in volatile memory will be lost. This section specifies the restoration of binding anchor attribute and binding state.

10.1. Binding Anchor Attribute Restoration

The configuration of binding anchor attribute is critical to this mechanism. If this configuration is lost, DHCPv4 Acknowledgement/ DHCPv6 Reply will be filtered. Though legitimate packet will not be discarded, host will be prevented from getting new DHCP address or renewing existing address.

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 SAVI device, if the configuration of binding anchor attribute can be found in non-volatile storage, the configuration MUST be used.

10.2. Binding State Restoration

The loss of binding state will cause legitimate traffic from host indirectly attached to the SAVI device to be blocked until binding recovery. Purely using the Binding Recovery Process to recover a large number of bindings is of heavy overhead and results considerable delay. Thus, recovery from non-volatile storage, as specified below, is RECOMMENDED.

If this function is supported by hardware, 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.

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

11. Constants

MAX_DHCP_RESPONSE_TIME 120s

BIND_RECOVERY_INTERVAL 60s and configurable

MAX_LEASEQUERY_DELAY 10s

OFFLINK_DELAY 30s

DAD_TIMEOUT 0.5s

12. MLD Considerations

To perform the binding recovery procedure in Section 8, the SAVI device MUST join the tentative address multicast group whenever performing duplicate detection.

13. Security Considerations

13.1. Security Problem about Binding Triggered by EVE_DHCP_REPLY_NULL

When the binding anchor is a switch port, binding based on EVE_DHCP_REPLY_NULL can result in security threats. The assigned

address could be bound to a wrong switch port if an attacker can maliciously pollute the table mapping a link layer address to switch port (cf. Section 6.2).

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

Protection from this attack can be ensured by making sure that one of the following conditions is satisfied:

(1) DHCP Option 82 is used to keep binding anchor in DHCP Request and Reply. DHCP Option 82 can be used to keep the circuit information of the client and returned by the DHCP server. Thus the binding anchor can be determined from the circuit information in the Option. It can be used whenever an implementation doesn!_t want to create an entry on the DHCP Request message.

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

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

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

It is NOT RECOMMENDED to initialize a binding based on DHCP Reply, unless a mechanism protecting the mapping table from corruption is also implemented. Similar problem may happen with binding anchors not based on link layer addresses.

13.2. Binding Number Limitation

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

(1) Set the upper bound of binding number for each binding anchor with SAVI-Validation.

(2) Reserve a number of binding entries for each binding anchor with SAVI-Validation attribute and all binding anchors share a pool of the other binding entries.

(3) Limit DHCP Request rate per binding anchor.

13.3. Risk from Link Layer Routing Dynamic

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

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

If the assumption doesn!_t stand, a better solution is requiring inter-operation between SAVI protocol and the link layer routing protocol to make SAVI protocol sensitive to the link layer routing change. This solution is above the scope of this document.

13.4. Duplicate Bindings of Same Address

The same address may be bound with multiple binding anchors, only if the binding processes are finished on each binding anchor successfully. This mechanism is designed in consideration that a node may move on the local link, and a node may have multiple binding anchors. However, the traceability of address is reduced.

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

13.5. Security Problems about Binding Recovery Process

The Binding Recovery Process (cf. Section 8) MUST be rate limited to avoid Denial of Services attack against the SAVI device itself. A constant BIND_RECOVERY_INTERVAL is used to control the frequency. Two data-triggered recovery processes on one binding anchor MUST have a minimum interval time BIND_RECOVERY_INTERVAL. This constant SHOULD

be configured prudently to avoid Denial of Service attacks.

This process is not strictly secure. The node attached with a SAVI-BindRecovery binding anchor has the ability to use the address of an inactive node, which doesn!_t reply to the detection probes.

13.6. Compatibility with DNA (Detecting Network Attachment)

DNA [rfc4436] [rfc6059] is designed to decrease the handover latency after re-attachment to the same network. DNA mainly relies on performing reachability test through sending unicast Neighbor Solicitation/Router Solicitation/ARP Request message to determine whether a previously configured address is still valid. Though DNA provides optimization for host, it doesn!_t provide sufficient information for this mechanism to migrate or establish a binding. If a binding is set up only through snooping the reachability test message, the binding can be invalid. For example, an attacker can perform reachability test with address bound to another host. If binding is migrated to the attacker, the attacker can successful 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 re-configuration of the interface is slowed but not prevented. Details are discussed as follows.

In Simple DNAv6 [rfc6059], the probe is sent with source address set to link-local address, and such messages will not be filtered by the policy specified in section Section 9.2. If an interface is re-attached to a previous network, the detection will be complete and the address will be regarded as valid by host. The candidate address is not contained in the probe. Thus, the binding cannot be recovered through snooping the probe. The binding can only be recovered from the DHCP snooping procedure. The DHCP REQUEST messages wouldn!_t be filtered by this solution as their source address is link-local address. Before the DHCP procedure is completed, packets will be filtered by SAVI device. In another word, in SAVI scenarios, Simple DNAv6 will not help reduce the handover latency. If SAVI-BindRecovery attribute is configured on the new binding anchor, data triggered procedure may reduce the latency.

In DNAv4 [rfc4436], the ARP probe will be filtered because unbound address is used as sender protocol address. As a result, the detection will not complete and false negative is caused. The DHCP REQUEST message sent by the node will not be filtered, because the source IP address field should be all zero as required by [rfc2131]. Thus, if the address is still valid, the binding will be recovered from the DHCP snooping procedure.

13.7. Bogus DHCP Server Threat

SAVI-DHCP-Trust attribute is designed to prevent attacks from bogus DHCP server. However, the security is not strict because messages from valid DHCP server and bogus DHCP server may arrive at the SAVI device with the same binding anchor. As a result, the SAVI device cannot recognize valid messages from bogus messages. Because the bindings are set up primarily based on DHCP message from DHCP server, the results can be quite serious. For example, invalid addresses are assigned to hosts and bindings of the addresses are set up. There can be a lot of other attacks in such a scenario.

Considering the restraint that no new protocol can be introduced, the countermeasure is quite limited. If placing the DHCP server inside the protection perimeter, or making the path from each valid DHCP servers to the SAVI perimeter exclusive for DHCP servers, or filtering out DHCP server/relay type messages that may get into the path from each DHCP server to the protection perimeter, messages from bogus DHCP server cannot share the same binding anchor with messages from valid DHCP server, and such attack can be prevented. If none of the above deployment requirements can be satisfied, the network administrator should be aware of the above limitation and deploy this mechanism prudently.

13.8. Handle Binding Anchor Off-link Event

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

Whenever a binding anchor with attribute SAVI-Validation turns down, a timer of OFFLINK_DELAY is set. Until the timer becomes zero, the bindings with the binding anchor SHOULD be kept. As an exception to handle node movements, if receiving DAD Neighbor Solicitation/ Gratuitous ARP request targeting at the address during OFFLINK_DELAY, the entry MAY be removed.

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

13.9. Residual Threats

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

(1) The binding anchor is spoofable. If the binding anchor is spoofable, e.g., plain MAC address, an attacker can use forged binding anchor to send packet which will not be regarded as spoofing by SAVI device. Indeed, using binding anchor that can be easily spoofed is dangerous. An attacker can use the binding anchor of another host to perform a lot of DHCP procedures, and the SAVI device will refuse to set up new binding for the host whenever the binding number limitation has been reached. Thus, it is RECOMMENDED to use strong enough binding anchor, e.g., switch port, secure association in 802.11ae/af and 802.11i.

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

14. IANA Considerations

This memo asks the IANA for no new parameters.

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

15. Acknowledgment

Special thanks to Jean-Michel Combes, Christian Vogt, Joel M. Halpern, Eric Levy-Abegnoli, Marcelo Bagnulo Braun, Jari Arkko, Elwyn Davies, Barry Leiba and Alberto Garcia for careful review and valuation comments on the state machine and text.

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

This document was generated using the xml2rfc tool.

16. References

16.1. Informative References

[BA2007] Baker, F., "Cisco IP Version 4 Source Guard", IETF Internet draft (work in progress), November 2007.

[BCP38] Paul, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", RFC 2827, BCP 38, May 2000.

[rfc3736] Droms, R., "Stateless Dynamic Host Configuration Protocol (DHCP) Service for IPv6", RFC 3736, April 2004.

16.2. Normative References

[rfc2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, BCP 14, Match 1997.

[rfc2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, March 1997.

[rfc3315] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.

[rfc4388] Woundy, R. and K. Kinnear, "Dynamic Host Configuration Protocol (DHCP) Leasequery", RFC 4388, February 2006.

[rfc4436] Aboba, B., Carlson, J., and S. Cheshire, "Detecting Network Attachment in IPv4 (DNAv4)", RFC 4436, March 2006.

[rfc4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, September 2007.

[rfc4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, September 2007.

[rfc5007] Brzozowski, J., Kinnear, K., Volz, B., and S. Zeng, "DHCPv6 Leasequery", RFC 5007, September 2007.

[rfc5227] Cheshire, S., "IPv4 Address Conflict Detection", RFC 5227, July 2008.

[rfc6059] Krishnan, S. and G. Daley, "Simple Procedures for Detecting Network Attachment in IPv6", RFC 6059, November 2010.

[rfc826] Plummer, D., "Ethernet Address Resolution Protocol: Or converting network protocol addresses to 48.bit Ethernet address for transmission on Ethernet hardware", RFC 826, November 1982.

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

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

Appendix A. change log

Main changes from 02 to 03:

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

Main changes from 03 to 04:

(1) Section "Prefix configuration" is removed.

(2) Section "Supplemental binding process" is modified in requirement level.

(3) Sub-section 9.1 "Rationale" is added.

(4) Section "Filtering during Detection" is removed.

(5) Section "Handling layer 2 path change" is changed to "Consideration on Link layer routing complexity"

(6) Section "Background and related protocols" is removed.

Main changes from 04 to 05:

(1) Trigger events are listed explicitly in section 8.

(2) Detection and Live states are deleted, together with corresponding sections.

Main change from 05 to 06:

(1) Section 8.1: reference to section 20 is changed to section 15.

Main changes from 06 to 07:

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

Main changes from 07 to 08,09:

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

Main changes from 09 to 11:

(1) DNA issues raised by Jari Arkko

Main changes from 11 to 12:

(1) The modifications are made according to the comments from Eric, http://www.ietf.org/mail-archive/web/savi/current/msg01778.html.

Main changes from 12 to 13:

(1) Main modifications are made based on comments from Elwyn Davies. http://www.ietf.org/mail-archive/web/gen-art/current/msg07297.html.

(2) Other modifications are made based on comments from Barry Leiba.

Authors' Addresses

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

Email: junbi@tsinghua.edu.cn

Jianping Wu
Tsinghua University
Computer Science, Tsinghua University
Beijing 100084
China

Email: jianping@cernet.edu.cn

Guang Yao
Tsinghua University
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