This document is published in:

*IEEE Communications Magazin*e, 2013, 54(4), 66-73.
DOI: http://dx.doi.org/10.1109/MCOM.2013.6495763

SAVI: The IETF Standard in Address Validation

Marcelo Bagnulo and Alberto García-Martínez, Universidad Carlos III de Madrid, Spain

ABSTRACT

In this article we describe Source Address Validation Implementation (SAVI), a security architecture being standardized by the IETF to prevent source address spoofing within a link. SAVI devices, usually layer 2 switches, create bindings between the IP address of a node and a property of the host's network attachment, such as the port through which the packet is received. Bindings are created by monitoring the packet exchange associated with IP address configuration mechanisms such as DHCP, SLAAC, or SEND. SAVI devices filter out packets whose source IP address does not match with an existing binding.

INTRODUCTION

The connectionless paradigm of the network layer of the Internet has naturally made possible *source address spoofing*, that is, the use of a source address by a node which is not allowed to use that address. Attackers can use spoofed source addresses to prevent tracing and to defeat source-based filtering when performing flood-based denial-of-service or poisoning attacks, or when propagating worms or malware [1].

The concern about the risks induced by source address spoofing has resulted in the recommendation of the deployment of *ingress filtering* [2]. This technique consists of the filtering of any packet with a source address that does not belong to the set of prefixes assigned to the part of the topology from which the packet comes. Ingress filtering is usually performed close to the site at which packets are originated, in either the egress router of the site or the ingress router of the direct provider. This strategy of deploying filters close to the site implies not only more effective protection, but also that it is easier to determine the prefixes corresponding to the site, by either explicit configuration or inference from the routing tables. The effectiveness of this technique largely depends on its widespread deployment, since any host connected to a site where ingress filtering is not performed could generate packets with any forged address.

However, even if ingress filtering were universally deployed, we would still face residual vulnerabilities. In particular, because ingress filtering operates at the prefix level, an attacker can still spoof any address from the prefix assigned to that part of the topology. This enables a number of serious attack vectors, allowing, for example, worms/malware to spoof a source address in order to hide the identity of the infected system. For a thorough description of this and other attacks enabled by source address spoofing that are not protected by ingress filtering, the reader is referred to [1].

In this article, we present SAVI, a mechanism that complements ingress filtering and provides increased protection. SAVI is in the final stages of standardization in the Internet Engineering Task Force (IETF).

In a nutshell, SAVI protects each individual address from spoofing by placing the source address filters closer to the nodes, preferably in the layer 2 switches that connect the nodes of a link.[1] In this deployment scenario, a switch configures a SAVI binding between an IP address and a node-specific layer 2 *binding anchor*. The physical port of the switch to which the node is attached is the canonical example of a binding anchor. The bindings are automatically created by inspecting the protocol exchange used to configure the IP addresses of the nodes. This allows the switch to filter out packets that do not correspond to an existing SAVI binding.

The contributions of this article are the following. We provide an integrated perspective of the SAVI solution, whose components are described in multiple IETF documents, we provide insights into the rationale for key design decisions, and we provide the background needed to understand the different requirements that lead to the final design; we also compare the solutions and discuss their applicability.

The rest of the article is organized as follows. First, we present the design background for SAVI. We then discuss the SAVI architecture. Next, we describe the solutions for the different address configuration mechanisms: Dynamic Host Configuration Protocol (DHCP), nodes locally configuring IPv6 addresses, and SEND. We compare the three SAVI solutions. Finally, we draw some conclusions.

DESIGN BACKGROUND

As the filtering process is moved closer to the end nodes, the difficulty in determining which source addresses should or should not be filtered

[1] *The term link refers to a topological area bounded by routers that decrement the IPv4 time to live (TTL) or IPv6 hop limit when forwarding the packet, as defined in RFC 4903.*

0163-6804/13/$25.00 © 2013 IEEE

increases. When ingress filtering is performed for a site, the set of source addresses that can be used legitimately results from the address assignment process at the prefix level, and the configuration of the filter is straightforward and fairly static. However, when filtering is to be performed for each individual IP address, management costs rise. Static configuration of the bindings is cumbersome and prevents *layer 2 host mobility* (i.e., prevents hosts from continuing communication after changing the point to which they attach to the layer 2 infrastructure). An alternative would be to define new protocols to allow nodes to prove address ownership themselves, but this would imply changing the end nodes and would certainly limit the deployment of such a solution. Finally, the traffic being exchanged by the nodes could be monitored and used to infer address ownership. For example, the underlying layer 2 infrastructure could inspect the DHCP messages to learn which addresses are assigned to which nodes, and prevent nodes using other binding anchors (for example, connected to other ports) from using the addresses illegitimately. Note that the extent to which address ownership can be successfully inferred largely depends on the method used for configuring addresses in nodes. In particular, if there is no central authority to assign addresses, as occurs in IPv6 auto-configured addresses, a specific mechanism should be devised to allow the filtering device to determine which of the nodes trying to use the same address should be granted communication.

The SAVI solution relies on traffic monitoring to prove address ownership. Three solutions for source address validation (being referred to as *SAVI solutions*) have been produced, each one tailored to a specific address configuration mechanism. The first solution, DHCP SAVI [3], inspects DHCP and DHCPv6 messages to deduce which addresses are assigned to which nodes. The second solution, FCFS (First-Come, First-Served) SAVI [4], is aimed at nodes locally configuring IPv6 addresses, using for example the Stateless Address Autoconfiguration mechanism [5]. Finally, SEND SAVI [6] benefits from the ability of nodes already implementing SEND (Secure Neighbor Discovery protocol, [7]) to prove address ownership by means of cryptographic addresses and router certificates.

SAVI Architecture

SAVI prevents IP source address spoofing by filtering out packets for which a *SAVI binding* does not exist. A SAVI binding is an association between an IP address and a *binding anchor*, a property of the host's network attachment, such as the physical port of the SAVI device to which the host connects. The binding anchor must be verifiable and hard to spoof. The current SAVI specifications consider the host's attachment port as the binding anchor. In this case, packets with a given IP source address are forwarded only when arriving from a particular physical port. However, SAVI specifications are flexible enough to support other binding anchors, such as IEEE 802.1X security associations [8].

We can identify two main architectural components of SAVI, the *binding creation* mechanism and the *filtering* mechanism. SAVI bindings are created dynamically as a result of the traffic inspection process, according to the address configuration mechanism used in the link. Each SAVI solution defines its own rules for the creation and refreshing of bindings. The filtering mechanism inspects data packets and verifies their source address and anchors against the existing list of bindings. The address configuration messages used to create the bindings are processed according to special filtering rules, as we describe in detail for each particular SAVI solution. Note that the filtering and binding creation processes are essentially orthogonal. In this section we describe the filtering mechanisms that are common for the different means to create bindings. Binding creation is described in the following sections.

Ideally, SAVI functionality should be deployed as close to the end hosts as possible (e.g., in the link layer switches). This is so because while SAVI filtering can be performed in any packet forwarding device in the same link as the hosts, the protection is more effective if SAVI filtering is performed close to the end hosts so that all the traffic is validated. In addition, when the port is used as the binding anchor, optimal protection is achieved by having only one node connecting to one port of the device performing SAVI filtering (hereafter *SAVI device*). However, in real-life deployments it may be infeasible to enable SAVI functionality in every switch of the network. In this case SAVI can be deployed in a switch higher in the hierarchy, still providing some protection.

In a link we usually find hosts and routers. Both hosts and routers generate packets with their own addresses as the source address. Routers, in addition, forward packets coming from other links. SAVI devices validate source addresses local to the link and also prevent hosts from generating packets with off-link addresses. In order to achieve that, SAVI defines two types of ports, *Validating ports* and *Trusted ports*. Validating ports are ports where filtering is performed by verifying the source addresses of packets coming through these ports against an existing binding. Trusted ports are ports in which validation is not performed. Hosts should be connected to Validating ports, and routers should be connected to Trusted ports, allowing the packets with off-link source addresses to be forwarded by routers and not by hosts. Failing to do the former would result in the SAVI device discarding off-link traffic, and failing to do the latter would imply that malicious hosts could generate packets with spoofed source addresses.

It is common that current links connect many switches in order to accommodate a large number of hosts. The deployment of SAVI in scenarios composed of many switches presents scalability challenges in terms of both memory and processing. Large memory requirements for SAVI devices result from the need to store the binding information. In addition, processing requirements spring from the need to create and maintain the binding information for the large number of connected nodes, and validate the source address of the many packets exchanged

A SAVI binding is an association between an IP address and a binding anchor, a property of the host's network attachment, such as the physical port of the SAVI device to which the host connects. The binding anchor must be verifiable and hard to spoof.

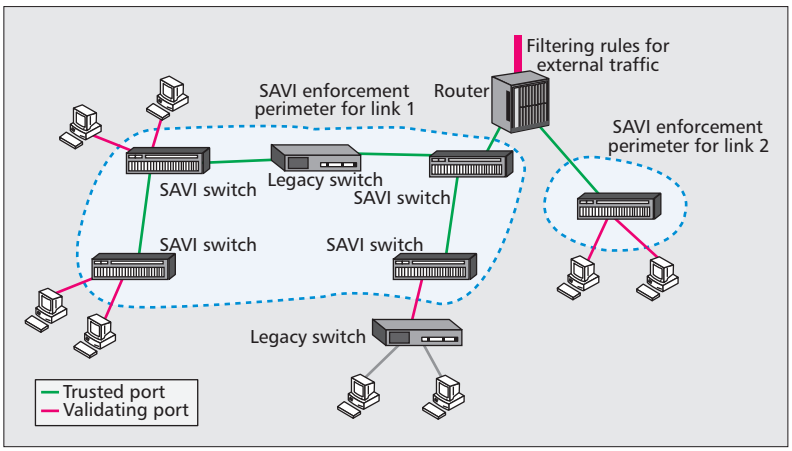

Figure 1. *Example of the deployment of a protection perimeter for SAVI.*

among them. In order to reduce the number of packets each switch must validate and the state required for this operation, the SAVI architecture supports the configuration of a *protection perimeter* [9]. The protection perimeter divides the link into trusted and non-trusted zones. The ports of the SAVI devices that are connected to a trusted zone are configured as *Trusted* ports, because they connect to trustworthy devices, such as other SAVI switches or routers. The ports connecting to a non-trusted zone are configured as *Validating* ports. In particular, ports connecting to hosts are configured as Validating ports. The performance gain of deploying a protection perimeter results from checking the validity of each packet only when it ingresses into the protection perimeter. An example of the configuration of the SAVI protection perimeter is depicted in Fig. 1.

DHCP SAVI

The Dynamic Host Configuration Protocol (for either IPv4 [10] or IPv6 [11]) defines a mechanism for configuring IP addresses to hosts. Typical operation for DHCP for IPv4 occurs as follows: a node broadcasts a DHCPDISCOVER message (with the unspecified IP source address) to request addresses from DHCP servers. One or more servers respond with a DHCPOFFER message, which includes possible address(es) to configure. The node selects one of these address offers and broadcasts a DHCPREQUEST message, which is acknowledged by the corresponding server with a DHCPACK message.

Address assignment for DHCPv6 proceeds with slight variations: after auto-configuring a Link-Local address, the node issues a Router Solicitation message to discover routers in the link and receive other configuration data. If the link is intended to use DHCPv6 for address configuration, the router responds with a Router Advertisement message with the M (managed address configuration) flag set [12]. With the Link-Local address as source address and the *All_DHCP_Relay_Agents_and_Servers* multicast address as destination, the node issues a DHCPv6 SOLICIT message to discover the available servers. This message is answered with an ADVERTISE message in which the address(es) to configure are included. The node chooses one of the responding servers, and issues a REQUEST message including the address and other parameters selected for configuration. The selection is confirmed by the server by means of a REPLY message. A Rapid Commit option can be included in a DHCPv6 SOLICIT message in order to request a REPLY message directly from the server, thus reducing the exchange from four to two messages.

DHCP SAVI [3] relies on the ability of properly placed SAVI devices to monitor the DHCP message exchanges in order to infer address ownership. DHCP SAVI devices create a binding between the IP address granted by the DHCP server and the binding anchor used by the node to make the request. In particular, DHCP SAVI devices snoop IPv4's DHCPREQUEST, IPv6's REQUEST, or IPv6's SOLICIT with Rapid Commit option messages to identify the binding anchor of the sending node and obtain the IP address associated with the node from the inspection of the DHCPACK (IPv4) or REPLY (IPv6) messages. The lease time included in the server response is used to configure the Expiration timer associated with the SAVI state for the IP address. DHCP messages are also monitored to refresh or remove existing bindings.

Figure 2 illustrates the creation of a binding for DHCPv4 SAVI. Node N broadcasts a DHCPDISCOVER message (Fig. 2a). DHCP servers receive the message and respond to it. In Fig. 2b, node N selects server S1, and issues a DHCPREQUEST message. Upon reception of this message, SAVI switch B1 creates a temporary entry in the binding database associated with a *Maximum DHCP Response* timer. This entry is moved to *forwarding state* when switch B1 snoops the DHCPACK message sent by server S1 (Fig. 2c).

In Fig. 3 we illustrate the creation of a binding for DHCPv6 SAVI. After configuring a Link-Local address, node N generates a Router Solicitation message, which is responded to by a router with the indication to perform DHCPv6 address configuration. Node N initiates a two-message Rapid Commit exchange (Fig. 3b), triggering the creation of a temporary state in the binding database of B1. The message is responded to by the DHCPv6 Relay, after communicating with server S, with a REPLY message that makes B1 change the entry state to forwarding.

FCFS SAVI for IPv6 SLAAC Nodes

Stateless Address Autoconfiguration (SLAAC for short) [5] defines a mechanism by which a node can locally generate IPv6 addresses and check the uniqueness of these addresses. To generate the IPv6 addresses, *interface identifiers* are derived from permanent configuration of the link layer interface, such as the MAC address. In the case of global addresses, the prefix is obtained from Router Advertisement messages generated by local routers.

When configuring an IPv6 address, a node N executes the Duplicate Address Detection

(DAD) procedure [5] to verify that the address is not in use by any other node in the link. To do so, it issues a Neighbor Solicitation message (hereafter DAD_NS) to the Solicited Node multicast address corresponding to the address being configured. If another node L is already using the address, L must have joined the Solicited Node multicast group, so it receives the DAD_NS message. Upon reception of this message, L responds with a Neighbor Advertisement (DAD_NA) message, and node N receiving the message does not configure the address. If no other node is using the address, no response is obtained, and after some period of time the address is configured in node N. Until the process completes, the address is said to be in tentative state.

SLAAC is used by nodes for which changing its address each time they attach to the network is not an issue (e.g., a host running client applications). For these nodes, it is not so relevant to preserve their addresses over a long period of time, but to avoid the temporal coincidence of two nodes using the same address. Then the SAVI solution for SLAAC should ensure that as long as node N is using an address, no other node can use it. If node N releases the address and stops using it for some time, it is acceptable to allow another node, M, to use it. If node N tries to configure the address again, the DAD procedure will indicate that node M is using it, and N should configure a different address to regain connectivity. In summary, SLAAC address ownership is based on the "first come, first served" (FCFS) paradigm, where the first node that claims an address is the node that configures it.

FCFS SAVI enforces FCFS address ownership by inspecting the traffic generated by the nodes, as illustrated in Fig. 4. When node N is configuring address A, it verifies if another node has the same address already configured by generating a DAD_NS message [5]. A SAVI switch B1 that receives the DAD_NS message for address A through a Validating port and does not have a previous binding for this address checks if a binding exists for the requested address in another switch. To do so, B1 creates a tentative entry for A in its binding database associated with a *Tentative State* timer,[2] and forwards the DAD_NS message to the neighboring switches. The switches receiving this message, such as B2 and B3, proceed in the same way: forward the message to other neighboring switches and to any Validating port that is included as a binding anchor for address A. If there are no hosts with address A configured, no response arrives before the Tentative State timer at B1 expires (Fig. 4b). In this case a VALID binding to address A is created at B1 for the port through which N attaches.

However, if another host in the link has address A configured, and its closer SAVI switch has a binding for A, this host answers to the received DAD_NS message with a DAD_NA message. This case is depicted in Fig. 5. Note that only nodes for which a binding existed previously for A receive the DAD_NS message, preventing rogue nodes from hindering address configuration in legitimate nodes (i.e., host M is

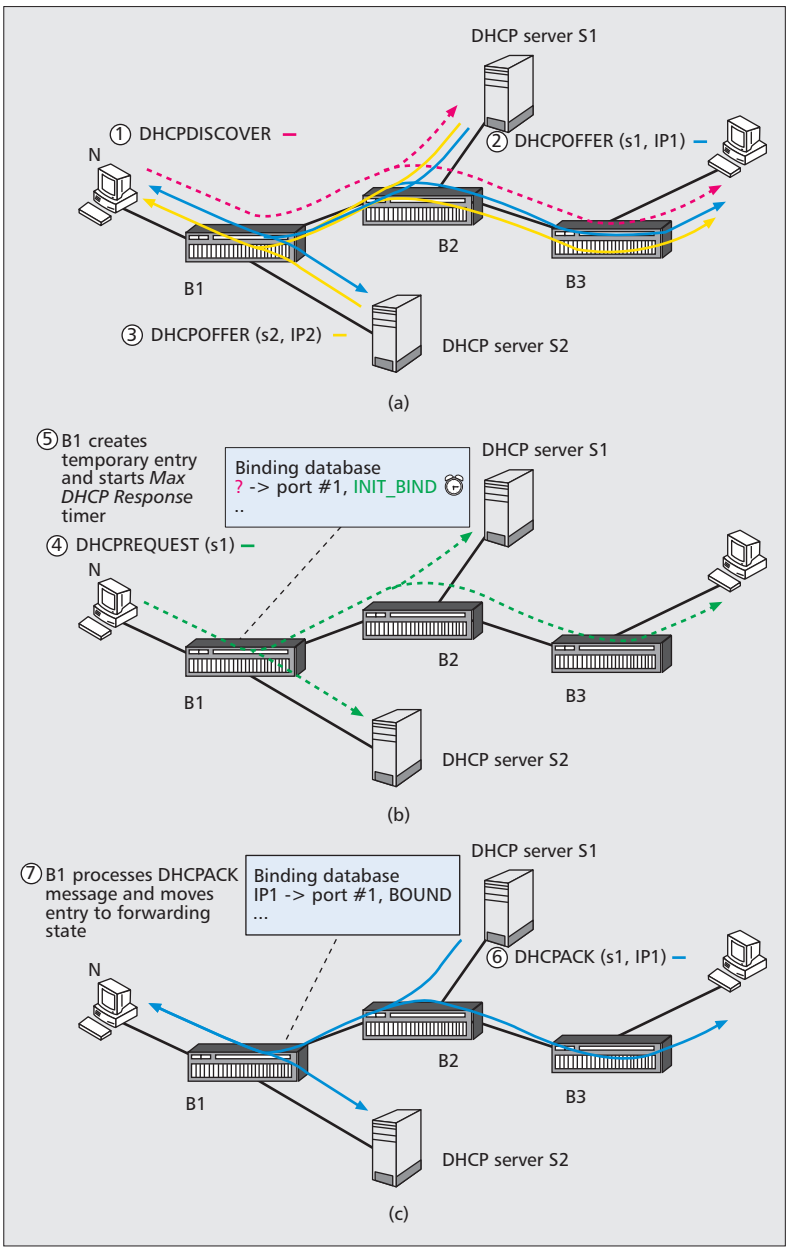

Figure 2. *Example of DHCP SAVI operation.*

the only host of the network receiving the DAD_NS message). When B1 receives the DAD_NA, it realizes that a binding already exists for A, and the local binding is not created. B1 forwards the DAD_NA message to the Validating port through which the DAD_NS was received so that host N is signaled about the address collision and can configure a different address.

Since the DAD procedure is inherently unreliable, FCFS SAVI design must be robust to the loss of the DAD messages used to create the bindings and to synchronize the SAVI devices within a realm. Thus, if DAD_NS messages are lost before reaching the first SAVI device in the path, or a node starts sending data packets without previously performing the DAD procedure, the SAVI device receiving the data packets synthesizes a DAD_NS message on behalf of the node to request information about the existence

[2] *The value of the Tentative State timer is related to the default value of the time specified for the IPv6 hosts to complete the DAD operation.*

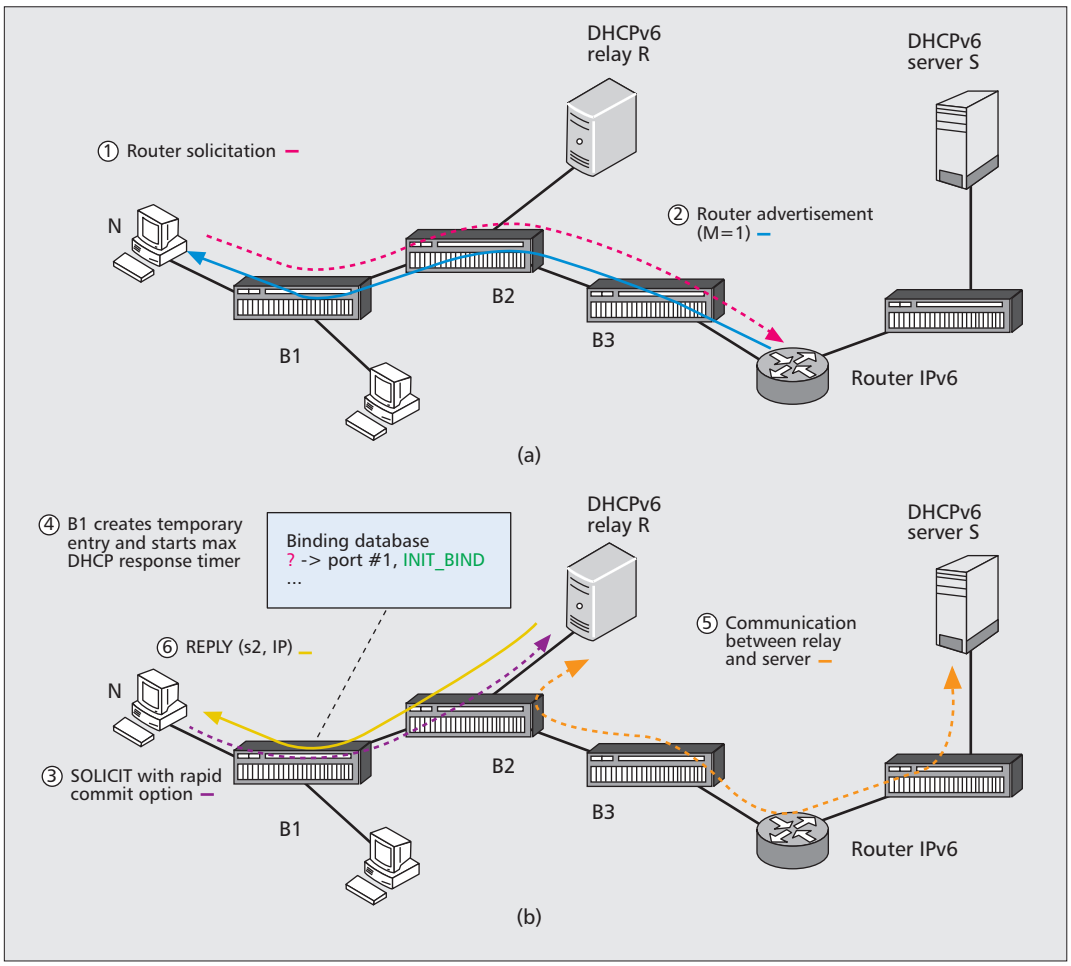

Figure 3. *Example of DHCPv6 SAVI operation.*

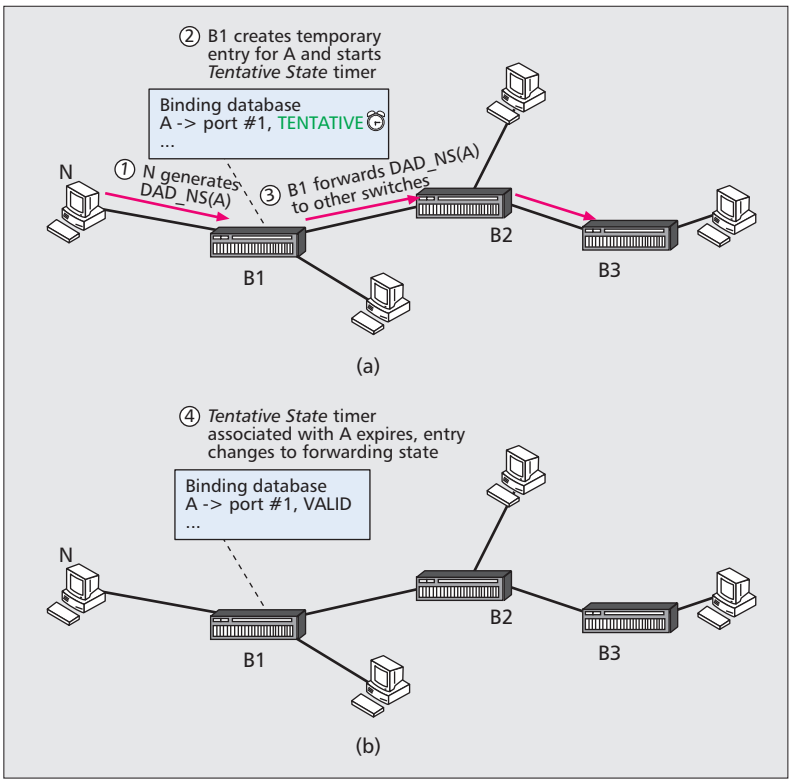

Figure 4. *Example of the successful creation of a SAVI binding in FCFS SAVI.*

of bindings in other ports or devices. If no response is received, a binding is created for the node.

We summarize the decision process of a SAVI switch upon reception of a packet received through a Validating port in the flowchart depicted in Fig. 6.

Valid bindings have an associated *Binding Lifetime* timer with a value similar to the expiration time of the *filtering database* entries of the switches where layer 2 forwarding information is stored. The SAVI device resets the Binding Lifetime timer when data packets are received from the rightful owner, and explicitly issues a DAD_NS to the currently registered owner of the address when the Binding Lifetime timer expires, to confirm the binding validity.

Layer 2 host mobility is supported by FCFS SAVI without any additional modification. If a node changes the attachment point from port P to port Q, it issues a DAD_NS message from port Q to start the DAD process again, as required by [5]. The message is forwarded to port P, but no host answers the request, so the node configures the address, the binding at Q is configured, and the binding at P is removed.

FCFS SAVI devices learn the prefixes associated with local traffic (i.e., the valid prefixes for hosts connecting to Validating ports) either by inspecting Router Advertisement messages

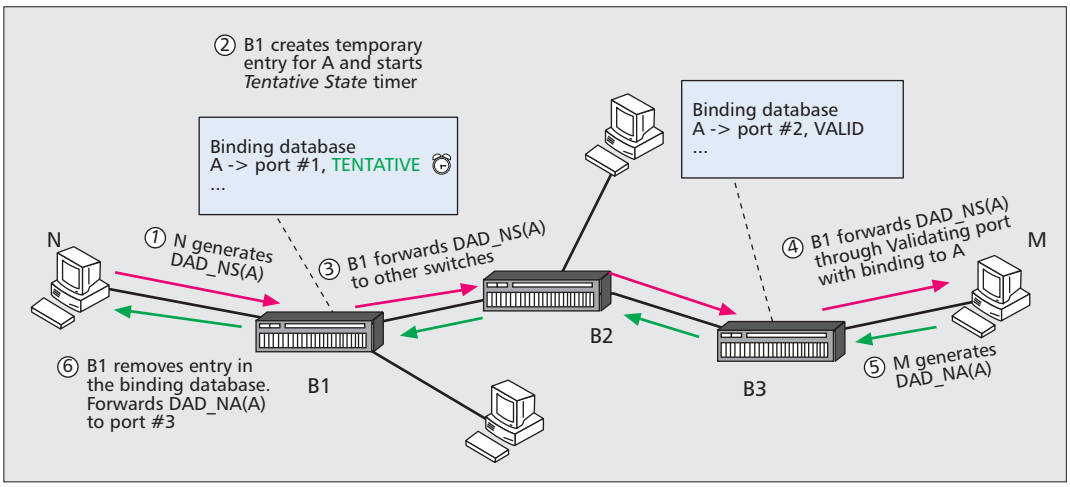

Figure 5. *Example of protection provided against address duplication in FCFS SAVI.*

Different prefixes in the same link may be protected by different mechanisms; for example, Link-Local addresses may be protected by FCFS SAVI, while global addresses may be configured by DHCP, and therefore could be protected by DHCP SAVI.

received through Trusted ports or as a result of manual configuration.

SEND SAVI

Secure neighbor discovery (SEND) [7] defines security extensions to IPv6 neighbor discovery (ND) operation to allow nodes to provide integrity, authentication, and authorization for the ND message exchange. Router authorization relies on certification paths with trust anchors that must be configured in every node of the link. In addition, SEND allows proving address ownership for hosts with locally generated addresses. To do so, hosts are required to create a private/public key pair and to generate a special type of IPv6 address called a cryptographically generated address (CGA) by including the result of a hash of their public key in the lower 64 bits of the address. Then they use the private key associated with the CGA to sign the ND messages. The public key associated with the CGA is also included in the SEND-specific information of the ND message. A node receiving a SEND message first checks that the public key is associated with the CGA, and then checks that the signature was made with the private key associated with the CGA. Therefore, the ability to sign ND messages is bounded to a particular IPv6 address, so the validation of an ND message also results in proving the ownership of the CGA. An attacker willing to impersonate a legitimate node using SEND needs to generate a public/private key pair with a hash of the public key matching the CGA of the target, which is computationally infeasible.

SEND SAVI benefits from the ability of SEND nodes to prove address ownership. A SEND SAVI device inspects and validates secure DAD_NS and DAD_NA messages to determine if the nodes involved in the exchange are authorized to use and configure the addresses. To refresh the binding and determine if a node sending data packets owns an address, secure NS messages are generated by the SEND SAVI device and sent through the Validating port to which the node using the address was attached. According to the Neighbor Unreachability [12] specification, hosts must answer with an NA message, which will be secured by SEND.

Secure Router Advertisement messages are used by SEND SAVI devices to determine the on-link prefixes and the routers that are allowed to inject off-link traffic.

Comparison of SAVI Solutions

In this section we compare the three SAVI solutions in terms of functionality and complexity. We summarize the analysis in Table 1.

Each of the SAVI solutions supports a set of address configuration mechanisms. DHCP SAVI supports both IPv4 and IPv6 addresses configured through DHCP and DHCPv6 respectively. FCFS SAVI supports IPv6 addresses, configured through Stateless Address Autoconfiguration, DHCPv6 or manually. This is so because all IPv6 addresses, irrespective of the mechanism through which they are configured, should perform the DAD procedure, enabling FCFS SAVI operation. SEND SAVI supports only IPv6 addresses that have been generated cryptographically, either via SAAC or manually. Different prefixes in the same link may be protected by different mechanisms; for example, Link-Local addresses may be protected by FCFS SAVI, while global addresses may be configured by DHCP, and therefore could be protected by DHCP SAVI. Using different mechanisms for the same prefix over the same link is currently being discussed [13] and is beyond the scope of this article.

The three SAVI solutions differ in the trust model considered. For DHCP SAVI, bindings are only created if the DHCP server explicitly authorizes it through the address assignment message exchange, resulting in a centralized trust model. In the FCFS SAVI case, bindings are created for a node as regular address configuration occurs, unless any other node defends the address of the binding, resulting in a distributed trust model. Finally, SEND SAVI operation relies on the ability of the nodes to generate messages that can be validated by the SAVI device according to the cryptographic key associated with the address.

The different solutions also differ in terms of address persistence protection. DHCP SAVI and SEND SAVI guarantee address persistence,

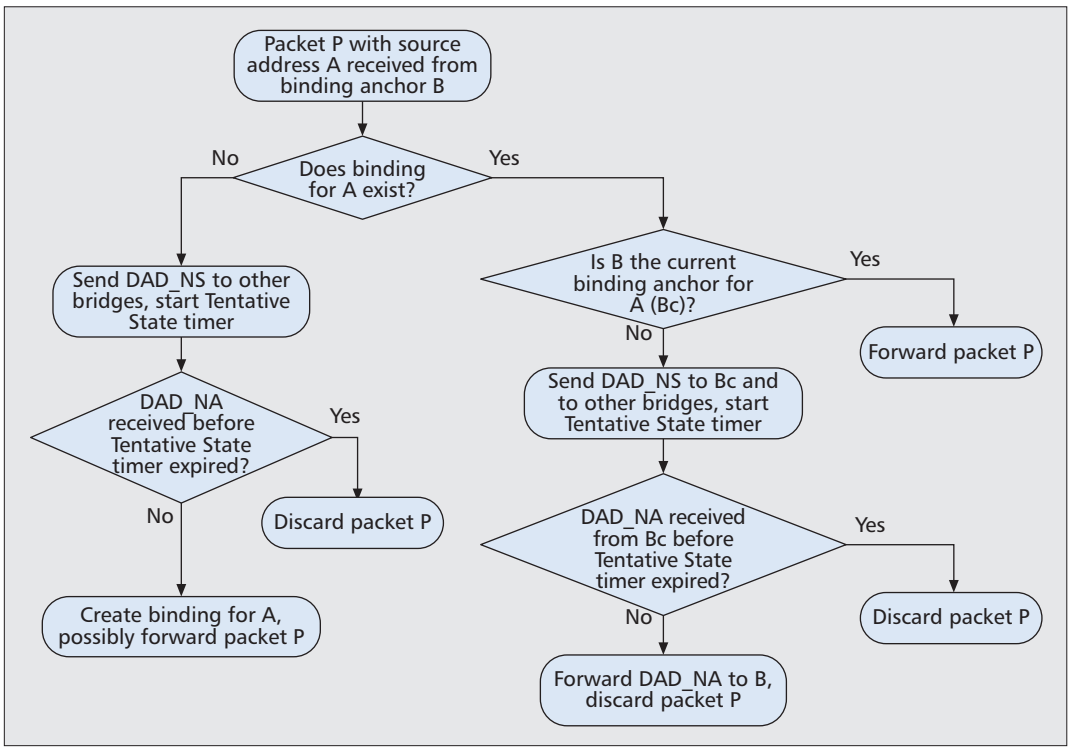

Figure 6. *Flowchart showing the decision process of an FCFS SAVI switch upon reception of a packet through a Validating port.*

	DHCP SAVI	FCFS SAVI	SEND SAVI
Address family	IPv4, IPv6	IPv6	IPv6
Address configuration method	DHCP	SLAAC, DHCP, Manual	SLAAC, Manual
Trust model	Centralized	Distributed	Cryptographic
Address persistence	DHCP based	No	Crypto-based
Messages involved	NS, NA, RA, DHCPREQUEST, DHCPACK, DHCPDECLINE, DHCPRELEASE, DHCPLEASEQUERY DHCPv6 REQUEST, DHCPv6 SOLICIT DHCPv6 REBIND, DHCPv6 REPLY, DHCPv6 LEASE	NS, NA, RA	Secure NS, Secure NA, Secure RA

Table 1. *Comparison of SAVI solutions.*

while FCFS SAVI does not. In the case of FCFS SAVI, as soon as the node stops defending an address (i.e., it stops answering the NS messages for a given address), FCFS SAVI will allow any other node to use that address. This is not the case for either DHCP SAVI or SEND SAVI. In the case of DHCP SAVI, the SAVI device honors the address assignments made by the DHCP server, so until the DHCP server does not allocate the address to another node, the address remains blocked. In the case of SEND SAVI, the protection provided by the cryptographic validation mechanism makes it infeasible for a node to use the address of another.

Finally, we discuss the number of messages the SAVI device needs to inspect, which is an indicator of the complexity of the solution. DHCP SAVI needs to inspect the DAD messages, the Router Advertisement, and many DHCP messages, while FCFS SAVI only needs to inspect the DAD messages and Router Advertisement messages. SEND SAVI needs to inspect the Secure Neighbor Discovery messages, but in addition, it needs to perform the crypto verification for them.

Conclusions

We have presented the SAVI framework for preventing source address spoofing within a link. The work to develop this set of specifications started at the IETF in 2008. So far, the document describing FCFS SAVI has been published as a Standards track RFC, the document describing DHCP SAVI is in the last stage of the standardization process (i.e., IESG review), and the SEND SAVI document has completed the Working Group last call.

Currently, there are commercial products that perform packet filtering within the link, with manually configured bindings or bindings derived from DHCP inspection.[3] Due to the modular architecture defined for SAVI, which separates filtering and binding creation, it is possible to leverage on existing implementations to develop SAVI-compliant filtering devices.

The standardization of the SAVI solutions will enable interoperability among the devices of

[3] *For example, this is supported by the IP Source Guard feature in Cisco and Juniper switches.*

different vendors, and the support of the new authorization models defined by FCFS and SEND.

ACKNOWLEDGMENTS

This article has received funding from Comunidad de Madrid, Spain, under the MEDIANET project (S2009/TIC-1468).

REFERENCES

[1] D. McPherson, F. Baker, and J. Halpern, "SAVI Threat Scope," draft-ietf-savi-threat-scope-05, Apr. 2011.
[2] P. Ferguson and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing," IETF RFC 2827, May 2000.
[3] J. Bi *et al.*, "SAVI Solution for DHCP," draft-ietf-savi-dhcp-15, Sept. 2012.
[4] E. Nordmark, M. Bagnulo, and E. Levy-Abegnoli, "FCFS SAVI: First-Come First-Served Source Address Validation Improvement for Locally Assigned IPv6 Addresses," IETF RFC 6620, May 2012.
[5] S. Thomson, T. Narten, and T. Jinmei, "IPv6 Stateless Address Autoconfiguration," IETF RFC 4862, Sept. 2007.
[6] M. Bagnulo and A. Garcia-Martinez, "SEND-Based Source-Address Validation Implementation," draft-ietf-savi-send-08. Sept. 2012.
[7] J. Arkko *et al.*, "Secure Neighbor Discovery (SEND)," IETF RFC 3971, Mar. 2005.
[8] IEEE, "IEEE 802.1X-2010 Port-Based Network Access Control," Feb. 2010.
[9] J. Wu *et al.*, "Source Address Validation Improvement Framework," draft-ietf-savi-framework-06, Dec. 2011.
[10] R. Droms. "Dynamic Host Configuration Protocol," IETF RFC 2131, Mar. 1997.
[11] R. Droms *et al.*, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)," IETF RFC 3315, July 2003.
[12] T. Narten *et al.*, "Neighbor Discovery for IP Version 6 (IPv6)," IETF RFC 4861, Sept. 2007.
[13] J. Bi *et al.*, "SAVI for Mixed Address Assignment Methods Scenario," draft-ietf-savi-mix-03, Nov. 2012.

BIOGRAPHIES

MARCELO BAGNULO (marcelo@it.uc3m.es) received an electrical engineering degree in 1999 from the University of Uruguay and a Ph.D. in telecommunications in 2005 from the University Carlos III of Madrid (UC3M), Spain. In 2000 he joined UC3M, where he has been an associate professor since 2006. He has published several papers in technical journals, magazines, and conferences. His main interest areas are IPv6 and interdomain routing.

ALBERTO GARCÍA-MARTÍNEZ (alberto@it.uc3m.es) received a telecommunications engineering degree in 1995 and a Ph.D. in telecommunications in 1999, both from the Polytechnic University of Madrid, Spain. In 1998 he joined UC3M, where he has been an associate professor since 2001. His main research areas are IPv6 and routing.