Troubleshoot SISF on Catalyst 9000 Series Switches

Contents

Introduction

Prerequisites

- Requirements
- Components Used
- Related Products

Background Information

- Overview
  - SISF Programmatic and Client Features
  - IPv4 Features that consume SISF information
  - IPv6 Features that consume SISF information
  - Device Tracking
  - SISF on a Port-Channel
  - Probe and Database Tuning
  - IP Device Tracking
  - Theft Detection
  - IP Security features
  - SISF Caveats

Troubleshoot

- Topology
- Configuration
- Verification
- Common Scenarios
  - Duplicate IPv4 Address Error on Host Device
  - Duplicate IPv6 Address Error
  - Increased Memory and CPU Utilization
  - Device Tracking Reachable Time Too Short
  - Switches Onboarded to Meraki Tool (CPU Increase & Port Flushes)
  - IP Addresses With Same MAC Not in SISF Table

Related Information

Introduction

This document describes the Switch Integrated Security Features (SISF) used in Catalyst 9000 Family Switches. It also explains how SISF can be used and how interacts with other features.

Prerequisites

Requirements

There are no specific requirements for this document.

Components Used

The information in this document is based on Cisco Catalyst 9300-48P that run Cisco IOS® XE 17.3.x

The information in this document was created from the devices in a specific lab environment. All of the devices used in this document started with a cleared (default) configuration. If your network is live, ensure that you understand the potential impact of any command.

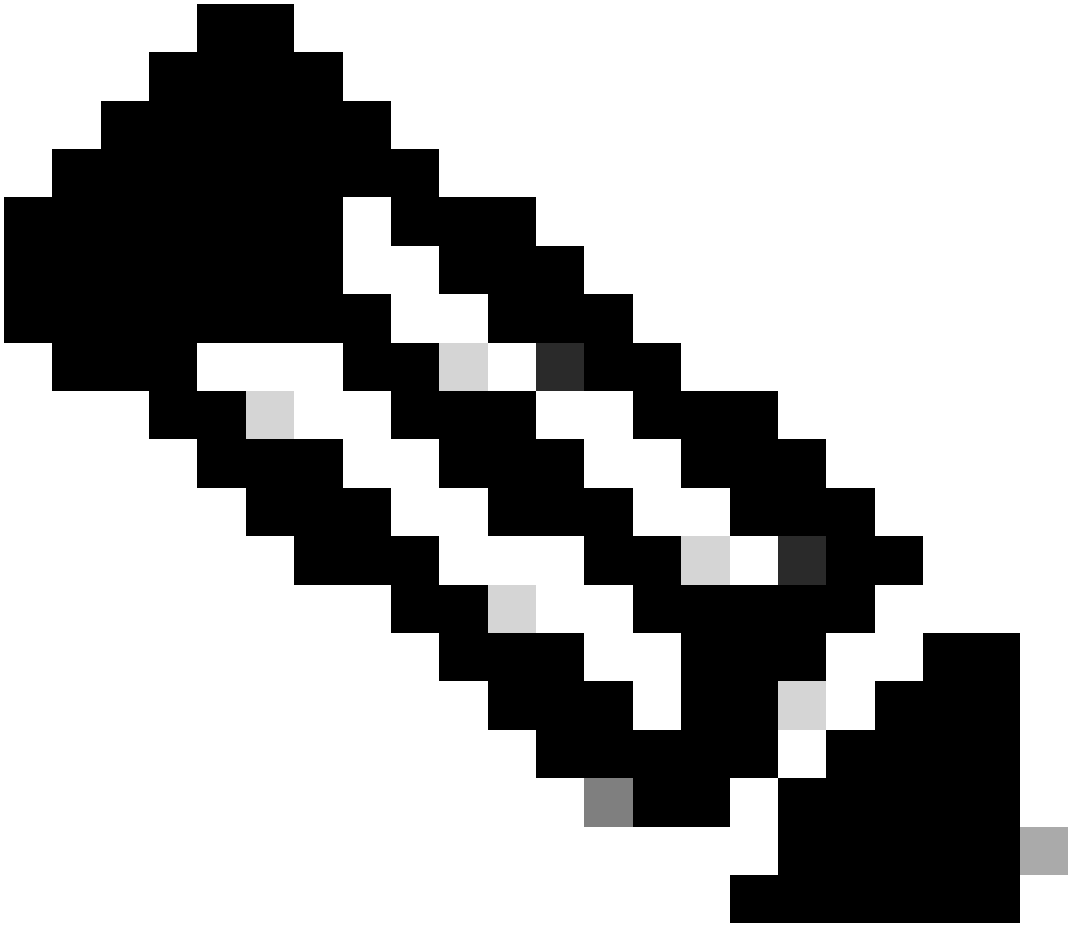
Note: Consult the appropriate configuration guide for the commands that are used in order to enable these features on other Cisco platforms.

Related Products

This document can also be used with these hardware and software versions:

- Catalyst 9200
- Catalyst 9300
- Catalyst 9400
- Catalyst 9500
- Catalyst 9600

With 17.3.4 and later Cisco IOS XE software versions



Note: This document is also applicable to most Cisco IOS XE versions that use SISF versus Device Tracking.

Background Information

Overview

SISF provides a host binding table, and there are feature clients who use the information from it. The entries are populated into the table by gleaning packets like DHCP, ARP, ND, RA which track the host activity and help to dynamically populate the table. If there are silent hosts into the L2 domain, static entries can be used to add entries into the SISF table.

SISF uses a policy model to configure device roles and additional settings on the switch. A single policy can be applied on interface or VLAN level. If a policy is applied on VLAN and a different policy is applied on interface, the interface policy takes precedence.

SISF can also be used to limit the number of hosts into the table but there are differences between IPv4 and IPv6 behavior. If SISF limit is set and it is reached:

- IPv4 hosts continue to operate but no more entries over the limit are to be added to the SISF table
- IPv6 hosts that do not make it into the SISF table are not allowed to enter into the network and no new entries are to be added to the SISF table.

From 16.9.x and newer release is introduced a SISF client feature priority. It adds options to control the updates into SISF and if two or more clients are using the binding table, updates from higher priority feature are applied. The exceptions here are the "limit address-count for IPv4//IPv6 per mac" settings, the settings of the policy with the lowest priority are effective.

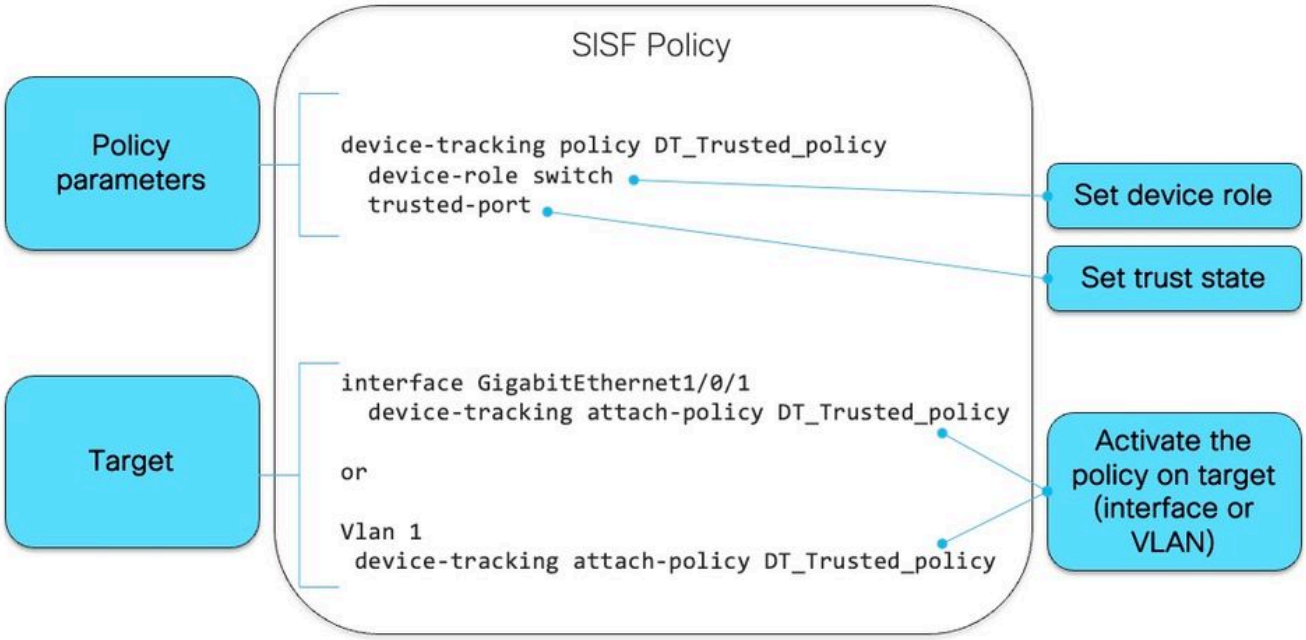
Some example features that requires device-tracking to be enabled are:

- LISP/EVPN
- Dot1x
- Web Auth
- CTS
- DHCP Snooping

Note: Priority is used to select policy settings.

Policy created from CLI has the highest priority (128), therefore allowing users to apply a policy setting different from the one in the programmatic policies. All the configurable settings under the customized policy can be manually changed.

Next image is example of a SISF policy and how to read it:



Inside of the policy, under protocol keyword, you have the option to see what type of packets are used to populate the SISF database:

<#root>

```
switch(config-device-tracking)#
```

?

```
device-tracking policy configuration mode:
  data-glean            binding recovery by data traffic source address
                        gleaning
  default               Set a command to its defaults
  destination-glean     binding recovery by data traffic destination address
                        gleaning
  device-role           Sets the role of the device attached to the port
  distribution-switch   Distribution switch to sync with
  exit                  Exit from device-tracking policy configuration mode
  limit                 Specifies a limit
  medium-type-wireless  Force medium type to wireless
  no                    Negate a command or set its defaults
  prefix-glean          Glean prefixes in RA and DHCP-PD traffic


protocol              Sets the protocol to glean (default all) <--

  security-level        setup security level
  tracking              Override default tracking behavior
  trusted-port          setup trusted port
  vpc                   setup vpc port

switch(config-device-tracking)#

protocol ?

  arp    Glean addresses in ARP packets
  dhcp4  Glean addresses in DHCPv4 packets
  dhcp6  Glean addresses in DHCPv6 packets
  ndp    Glean addresses in NDP packets
  udp    Gleaning from UDP packets
```

SISF Programmatic and Client Features

The features in the next table either enable SISF programatically when they are enabled or act as clients to SISF:

SISF Programmatic Feature	SISF Client Features
LISP on VLAN	Dot1x
EVPN on VLAN	Web Auth
DHCP Snooping	CTS

If a SISF client feature is enabled on a device configured without a feature that enables SISF, a custom policy must be configured on interfaces connecting to hosts.

IPv4 Features that consume SISF information

- CTS

- IEEE 802.1x
- LISP
- EVPN
- DHCP snooping (only activates SISF but does not use it)
- IP Source Guard

IPv6 Features that consume SISF information

- IPv6 Router Advertisement (RA) Guard
- IPv6 DHCP Guard, Layer 2 DHCP Relay
- IPv6 Duplicate Address Detection (DAD) Proxy
- Flooding Suppression
- IPv6 Source Guard
- IPv6 Destination Guard
- RA Throttler
- IPv6 Prefix Guard

Device Tracking

The main role of device tracking is to track the presence, location, and movement of end-nodes in the network. SISF snoops traffic received by the switch, extracts device identity (MAC and IP address), and stores them in a binding table. Many features, such as, IEEE 802.1X, web authentication, Cisco TrustSec and LISP and so on, depend on the accuracy of this information to operate properly. SISF-based device tracking supports both IPv4 and IPv6. There are five supportedmethods by which client can learn IP:

- DHCPv4
- DHCPv6
- ARP
- NDP
- Data gleaning

SISF on a Port-Channel

Device tracking on port-channel (or ether-channel) is supported. But the configuration must be applied on the channel group, not the individual port channel members. The only interface that shows up (and is known) from the binding standpoint is the port-channel.

Probe and Database Tuning

Probe:

- In IPDT there was a command to help with duplicate address issues by delaying the initial probe for 10 seconds: "ip device tracking probe delay" upon link up.
- In SISF there is already a wait timer built in that waits before sending first probe. It is not configurable, and solves the same problem. Since this is in the SISF code, there is no need for this command anymore

Database:

In SISF you can configure a few options to control how long an entry is kept in the database:

```
tracking enable reachable-lifetime <second|infinite>
<-- how long an entry is kept reachable (or keep permanently reachable)

tracking disable stale-lifetime <seconds|infinite>
<-- how long and entry is kept inactive before deletion (or keep permanently inactive)
```

IP Device Tracking

Lifecycle of an entry where the host is polled:

- SISF maintains IPv4/IPv6 binding per mac, once IP learn is successful, binding transitions to REACHABLE state
- SISF keep track liveness client by monitoring control packet
- If there is no control packet from the client for 5 mins, Binding transitions to VERIFY state and sends probe to client
- If clients does not respond to probe, Binding transitions to STALE state else REACHABLE state
- Default timeout for STALE entry is 24 hours and configurable
- STALE entries are deleted after 24 hours (or configured timeout value)

Theft Detection

Types of node thefts:

- IP Theft (same ip, different mac, different/same port)
- MAC THEFT (same MAC, different IP, different port)
- MAC IP THEFT (same mac, same ip, different port)

IP Security features

These are some of the SISF dependent features:

- **NDP inspection:** Inspect IPv6 NDP messages
- **NDP address-gleaning:**Populate the binding table with information glean by snooping NDP traffic
- **Device tracking:** Monitor end-device activity, including thru some liveness mechanism
- **Snooping:** Glean addresses in NDP ,ARP and DHCP messages. Block unauthorized messages
- **DHCPv4 Relay:** Relay DHCP broadcasted packet to configured helper address.
- **NDP & ARP multicast suppress:** Suppress multicast NDP messages by either converting to unicast, to responding on behalf of targets.
- **DAD proxy:**Duplicate address detection and sending NA behalf of target client
- **DHCPv4 Require:** It enforces the client to get IP through only by DHCP

SISF Caveats

Some of the most frequent behaviors observed related to SISF are:

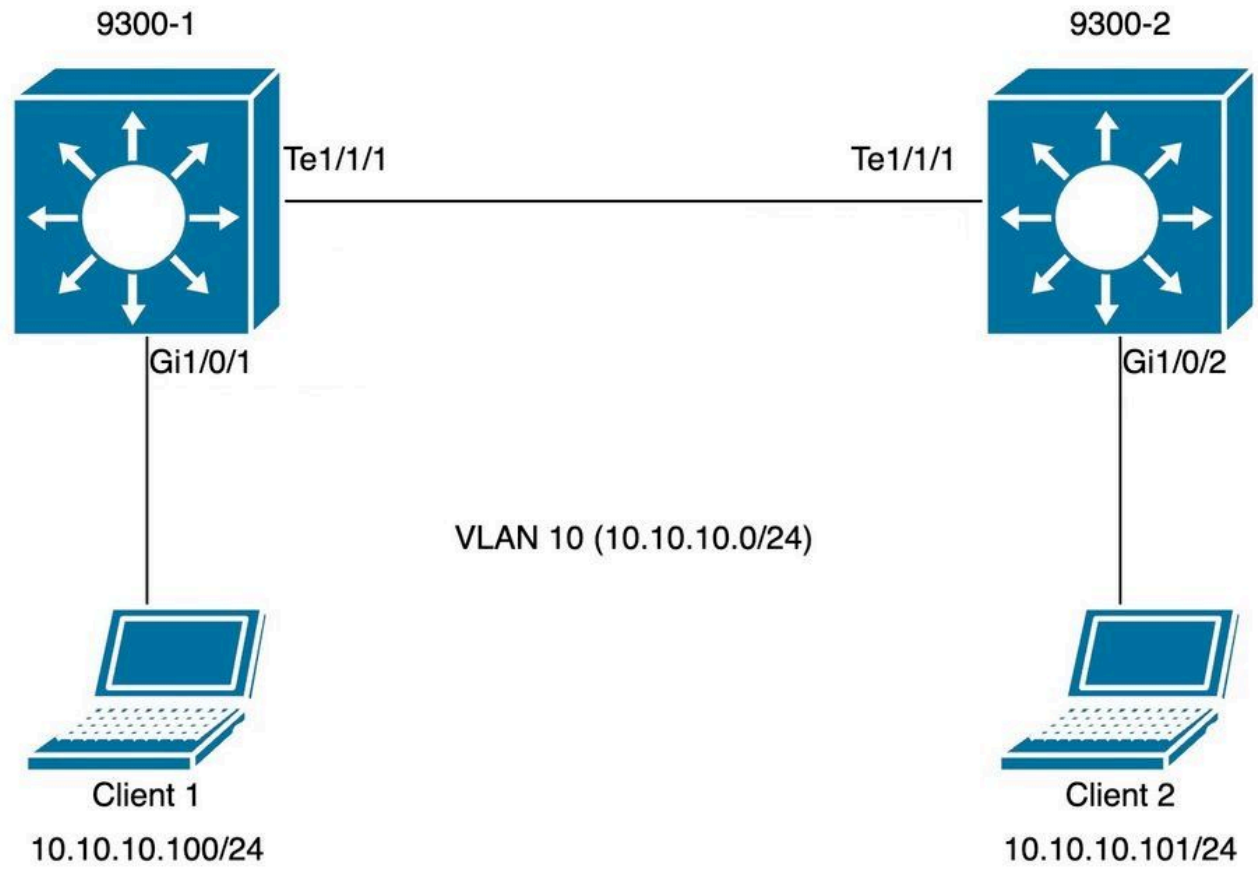
- SISF can be enabled by enabling other features such as dhcp snooping
- SISF’s default probe behavior can impact client IP address assignment.
- When SISF is enabled it is also enabled on uplink ports which can cause network impact.

Troubleshoot

Topology

The topology diagram is used on the next SISF scenario. 9300 switches are layer 2 only and do NOT have SVI configured in client Vlan 10.

Note: SISF is enabled in this lab manually.



Configuration

Default SISF configuration was set up on both 9300 switches facing access ports whereas custom policy was applied on trunk ports to illustrate the expected SISF outputs.

Switch 9300-1:

```
<#root>

9300-1#

show running-config interface GigabitEthernet 1/0/1

Building configuration...

Current configuration : 111 bytes
!
interface GigabitEthernet1/0/1
```

```
 switchport access vlan 10
 switchport mode access
 device-tracking <-- enable default SISF policy
end
9300-1#

9300-1#
show running-config | section trunk-policy


device-tracking policy trunk-policy <-- custom policy


trusted-port                        <-- custom policy parameters


device-role switch


<-- custom policy parameters


no protocol udp
9300-1#

9300-1#
show running-config interface tenGigabitEthernet 1/1/1

Building configuration...

Current configuration : 109 bytes
!
interface TenGigabitEthernet1/1/1
 switchport mode trunk
 device-tracking attach-policy trunk-policy <-- enable custom SISF policy


end
```

Switch 9300-2:

```
<#root>

9300-2#
show running-config interface GigabitEthernet 1/0/2

Building configuration...

Current configuration : 105 bytes
!
interface GigabitEthernet1/0/2
 switchport access vlan 10
 switchport mode access
 device-tracking
```

```
<-- enable default SISF policy

end

9300-2#

show running-config | section trunk-policy


device-tracking policy trunk-policy <-- custom policy


trusted-port                        <-- custom policy parameters


device-role switch


<-- custom policy parameters


no protocol udp

9300-2#

show running-config interface tenGigabitEthernet 1/1/1

Building configuration...

Current configuration : 109 bytes
!
interface TenGigabitEthernet1/1/1
 switchport mode trunk

 device-tracking attach-policy trunk-policy <-- custom policy applied to interface


end
```

Verification

You can use these commands to validate the policies applied:

```
show device-tracking policy <policy name>
show device-tracking policies
show device-tracking database
```

Switch 9300-1:

```
<#root>

9300-1#

show device-tracking policy default
```

```
Device-tracking policy default configuration:
  security-level guard

device-role node  <--

  gleaning from Neighbor Discovery
  gleaning from DHCP
  gleaning from ARP
  gleaning from DHCP4
  NOT gleaning from protocol unkn
Policy default is applied on the following targets:

Target

               Type

Policy



Feature

       Target range

Gi1/0/1

              PORT

default



Device-tracking

 vlan all

9300-1#

show device-tracking policy trunk-policy

Device-tracking policy trunk-policy configuration:

  trusted-port       <--

  security-level guard

  device-role switch <--

  gleaning from Neighbor Discovery
  gleaning from DHCP
  gleaning from ARP
  gleaning from DHCP4
  NOT gleaning from protocol unkn
Policy trunk-policy is applied on the following targets:

Target

               Type

Policy



Feature

      Target range

Te1/1/1
```

```
              PORT
trunk-policy


Device-tracking

 vlan all
9300-1#

9300-1#
show device-tracking policies

Target               Type  Policy               Feature        Target range
Te1/1/1              PORT  trunk-policy         Device-tracking vlan all
Gi1/0/1              PORT  default              Device-tracking vlan all

9300-1#
show device-tracking database


Binding Table has 1 entries, 1 dynamic (limit 200000)
Codes: L - Local, S - Static, ND - Neighbor Discovery, ARP - Address Resolution Protocol, DH4 - IPv4 DH
Preflevel flags (prlvl):
0001:MAC and LLA match     0002:Orig trunk            0004:Orig access
0008:Orig trusted trunk    0010:Orig trusted access   0020:DHCP assigned
0040:Cga authenticated     0080:Cert authenticated    0100:Statically assigned


    Network Layer Address                  Link Layer Address Interface  vlan  prlvl age    state     T
ARP 10.10.10.100                           98a2.c07e.7902     Gi1/0/1    10    0005  8s     REACHABLE 3


9300-1#
```

Switch 9300-2:

<#root>

```
9300-2#
show device-tracking policy default

Device-tracking policy default configuration:
  security-level guard
  device-role node   <--
  gleaning from Neighbor Discovery
  gleaning from DHCP
  gleaning from ARP
  gleaning from DHCP4
  NOT gleaning from protocol unkn
Policy default is applied on the following targets:
Target
               Type
```

Policy

Feature

Target range

Gi1/0/2

PORT

default

Device-tracking

vlan all

9300-2#

show device-tracking policy trunk-policy

Device-tracking policy trunk-policy configuration:

trusted-port <--

security-level guard

device-role switch <--

gleaning from Neighbor Discovery
gleaning from DHCP
gleaning from ARP
gleaning from DHCP4
NOT gleaning from protocol unkn

Policy trunk-policy is applied on the following targets:

Target

Type

Policy

Feature

Target range

Te1/1/1

PORT

trunk-policy

Device-tracking

vlan all

9300-2#

9300-2#

show device-tracking policies

Target	Type	Policy	Feature	Target range
Te1/1/1	PORT	trunk-policy	Device-tracking	vlan all

```
Gi1/0/2             PORT  default             Device-tracking vlan all

9300-2#
```

show device-tracking database

```
Binding Table has 1 entries, 1 dynamic (limit 200000)
Codes: L - Local, S - Static, ND - Neighbor Discovery, ARP - Address Resolution Protocol, DH4 - IPv4 DH
Preflevel flags (prlvl):
0001:MAC and LLA match     0002:Orig trunk            0004:Orig access
0008:Orig trusted trunk    0010:Orig trusted access   0020:DHCP assigned
0040:Cga authenticated     0080:Cert authenticated    0100:Statically assigned

    Network Layer Address                  Link Layer Address Interface  vlan  prlvl age    state     T
```

ARP 10.10.10.101   98a2.c07e.9902   Gi1/0/2   10   0005 41s   REACHABLE 2

```
9300-2#
```

Common Scenarios

Duplicate IPv4 Address Error on Host Device

Problem

The “keepalive” probe sent by the switch is a L2 check. As such from the switch’s point of view, the IP addresses used as source in the ARPs are not important: this feature can be used on devices with no IP address configured at all, so the IP source of 0.0.0.0 is not relevant. When the host receives this messages, it replies back and populates the destination IP field with the only IP address available in the received packet, which is its own IP address. This can cause false duplicate IP address alerts, because the host that replies sees its own IP address as both the source and the destination of the packet.

It is recommended to configure the SISF policy to use an auto-source for its keepalive probes.

Note: See this article on duplicate address issues for further information

---

Default probe

This is the probe packet when there is no local SVI present and default probes settings:

<#root>

```
Ethernet II,
```

Src: c0:64:e4:cc:66:02 (c0:64:e4:cc:66:02)

```
, Dst: Cisco_76:63:c6 (00:41:d2:76:63:c6)
```

<-- Probe source MAC is the BIA of physical interface connected to client

```
    Destination: Cisco_76:63:c6 (00:41:d2:76:63:c6)
        Address: Cisco_76:63:c6 (00:41:d2:76:63:c6)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
```

```
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Source: c0:64:e4:cc:66:02 (c0:64:e4:cc:66:02)
        Address: c0:64:e4:cc:66:02 (c0:64:e4:cc:66:02)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)
    Type: ARP (0x0806)
    Padding: 000000000000000000000000000000000000
Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: c0:64:e4:cc:66:02 (c0:64:e4:cc:66:02)
    Sender IP address: 0.0.0.0                                  <-- Sender IP is 0.0.0.0 (default)
    Target MAC address: Cisco_76:63:c6 (00:41:d2:76:63:c6)
    Target IP address: 10.10.10.101                             <-- Target IP is client IP
```

Solution

Configure the probe to use an address other than the host PC for the probe. This can be accomplished by these methods

Auto-Source for "Keep-Alive" Probe

Configure an auto-source for the "keep-alive" probes to reduce usage of 0.0.0.0 as the source IP:

```
device-tracking tracking auto-source fallback <IP> <MASK> [override]
```

The logic if applying the auto-source command works as follows:

<#root>

```
device-tracking tracking auto-source fallback 0.0.0.253 255.255.255.0 [override]
```

<-- Optional parameter

1. Set the source to VLAN SVI if present.
2. Search for a source/MAC pair in the IP host table for the same subnet. The probe as sourced from the switch physical interface MAC + the IP of some other host in the subnet already in the database.
3. Compute the source IP from the destination IP with the host bit and mask provided. Probe is generated from hearing client IP and creating a probe in the subnet with the last bits configured.

Note: If command is applied with <override> we always jump to step 3.

Modified probe

Setting auto-source fallback config to use an IP in the subnet modifies the probe. Since there is no SVI and no other client on the subnet we fall back to the configured IP/Mask in the config.

<#root>

```
switch(config)#device-tracking tracking auto-source fallback 0.0.0.253 255.255.255.0 <-- it uses .253 fo
```

This is the modified probe packet:

<#root>

```
Ethernet II, Src: c0:64:e4:cc:66:02 (c0:64:e4:cc:66:02), Dst: Cisco_76:63:c6 (00:41:d2:76:63:c6)

<-- Probe source MAC is the BIA of physical interface connected to client

    Destination: Cisco_76:63:c6 (00:41:d2:76:63:c6)

        Address: Cisco_76:63:c6 (00:41:d2:76:63:c6)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)

    Source: c0:64:e4:cc:66:02 (c0:64:e4:cc:66:02)

        Address: c0:64:e4:cc:66:02 (c0:64:e4:cc:66:02)
        .... ..0. .... .... .... .... = LG bit: Globally unique address (factory default)
        .... ...0 .... .... .... .... = IG bit: Individual address (unicast)

    Type: ARP (0x0806)

    Padding: 000000000000000000000000000000000000
Address Resolution Protocol (request)
    Hardware type: Ethernet (1)
    Protocol type: IPv4 (0x0800)
    Hardware size: 6
    Protocol size: 4
    Opcode: request (1)
    Sender MAC address: c0:64:e4:cc:66:02 (c0:64:e4:cc:66:02)

    Sender IP address: 10.10.10.253                                 <-- Note the new sender IP is now using t

    Target MAC address: Cisco_76:63:c6 (00:41:d2:76:63:c6)

    Target IP address: 10.10.10.101
```

Further Details on Probe behavior

Command	Action (In order to select source IP and MAC address for device tracking ARP probe)	Notes

device-tracking tracking auto-source	• Set source to VLAN SVI if present. • Look for IP and MAC binding in device-tracking table from same subnet. • Use 0.0.0.0	We recommend that you disable device-tracking on all trunk ports to avoid MAC flapping.
device-tracking tracking auto-source override	• Set source to VLAN SVI if present • Use 0.0.0.0	Not recommended when there is no SVI.
device-tracking tracking auto-source fallback <IP> <MASK>	• Set source to VLAN SVI if present. • Look for IP and MAC binding in device-tracking table from same subnet. • Compute source IP from client IP using host bit and mask provided. Source MAC is taken from the MAC address of the switchport facing the client.	We recommend that you disable device-tracking on all trunk ports to avoid MAC flapping. The computed IPv4 address must not be assigned to any client or network device.
device-tracking tracking auto-source fallback <IP> <MASK> override	• Set source to VLAN SVI if present. • Compute source IP from client IP using host bit and mask provided. Source MAC is taken from the MAC address of the switchport facing the client.	The computed IPv4 address must not be assigned to any client or network device.

Explanation of the **device-tracking tracking auto-source fallback <IP> <MASK> [override]** command:

Depending on the host ip, an IPv4 address needs to be reserved.

```
<reserved IPv4 address> =  (<host-ip> & <MASK> ) | <IP>
```

Note: This is a Boolean formula

Example.

If we use the command:

```
device-tracking tracking auto-source fallback 0.0.0.1 255.255.255.0 override
```

host IP = 10.152.140.25

IP = 0.0.0.1

mask = 24

Lets break the Boolean formula in two parts.

1. 10.152.140.25 AND 255.255.255.0 operation:

```
10.152.140.25 = 00001010.10011000.10001100.00011001
                              AND
255.255.255.0 = 11111111.11111111.11111111.00000000
                            RESULT
10.152.140.0  = 00001010.10011000.10001100.00000000
```

2. 10.152.140.0 OR 0.0.0.1 operation:

```
10.152.140.0  = 00001010.10011000.10001100.00000000
                              OR
0.0.0.1       = 00000000.00000000.00000000.00000001
                            RESULT
10.152.140.1  = 00001010.10011000.10001100.00000001
```

Reserved IP = 10.152.140.1

Reserved IP = (10.152.140.25 & 255.255.255.0) | (0.0.0.1) = 10.152.140.1

Note: Address used as the IP source must be scoped out of the DHCP bindings for the subnet.

Duplicate IPv6 Address Error

Problem

Duplicate IPv6 address error when IPv6 is enabled in the network and a switched virtual interface (SVI) is configured on a VLAN.

In a normal IPv6 DAD packet, the Source Address field in the IPv6 header is set to the unspecified address (0:0:0:0:0:0:0:0). Similar to IPv4 case.

The order for choosing Source Address in SISF probe is:

- Link-local address of SVI, if configured
- Use 0:0:0:0:0:0:0:0

Solution

We recommend that you add the next commands to the SVI configuration. This enables the SVI to acquire a link-local address automatically; this address is used as the source IP address of the SISF probe, thus preventing the duplicate IP address issue.

```
interface vlan <vlan>
 ipv6 enable
```

Increased Memory and CPU Utilization

Problem

The "keepalive" probe sent by the switch is broadcast out of all ports when it is programatically enabled. Attached switches in the same L2 domain send these broadcasts to their hosts resulting in the origin switch adding remote hosts to its device tracking database. The additional host entries increases the memory usage on the device and the process of adding the remote hosts increases the CPU utilization of the device.

It is recommended to scope the programmatic policy by configuring a policy on uplink to attached switches to define the port as trusted and attached to a switch.

Note: Be aware that SISF dependant features such as DHCP snooping enables SISF to work properly, which can trigger this problem.

Solution

Configure a policy on the uplink (trunk) to stop probes and learning of remote hosts that live on other switches (SISF is only needed to maintain a**local** host table)

<#root>

```
device-tracking policy DT_trunk_policy
 trusted-port
 device-role switch

interface <interface>
 device-tracking policy
DT_trunk_policy
```

Device Tracking Reachable Time Too Short

Problem

Due to a migration issue from IPDT to SISF-based device tracking, a non-default reachable time sometimes is introduced when migrating from older release to 16.x and newer releases.

Solution

It is recommended to revert to the default reachable time by configuring:

```
no device-tracking binding reachable-time <seconds>
```

Switches Onboarded to Meraki Tool (CPU Increase & Port Flushes)

Problem

When switches are onboarded to the Meraki Cloud Monitoring tool, such tool pushes custom device-tracking policies.

```
device-tracking policy MERAKI_POLICY
 security-level glean
 no protocol udp
 tracking enable
```

The policy is applied to all interfaces with no distinction, that means, it does not distinguish between edge ports and trunk ports that faces other network devices (for example switches, firewalls routers and so on). Switch can create several SISF entries on trunk ports where **MERAKI_POLICY** is configured, therefore causing flushes on these ports as well as CPU usage increaseds.

<#root>

switch#

show interfaces port-channel 5

Port-channel5 is up, line protocol is up (connected)
<*omitted output*>
  Input queue: 0/2000/0/

112327

 (size/max/drops/

flushes

); Total output drops: 0

<-- we have many flushes

<*omitted output*>

switch#

show process cpu sorted

```
CPU utilization for five seconds: 26%/2%; one minute: 22%; five minutes: 22%
PID Runtime(ms)     Invoked      uSecs   5Sec   1Min   5Min TTY Process

572     1508564      424873       3550 11.35%  8.73%  8.95%   0 SISF Main Thread

105      348502      284345       1225  2.39%  2.03%  2.09%   0 Crimson flush tr
```

Solution

Set up the next policy on all non-edge interfaces:

```
configure terminal
device-tracking policy NOTRACK
 no protocol ndp
 no protocol dhcp6
 no protocol arp
 no protocol dhcp4
 no protocol udp
exit

interface <interface>
device-tracking policy NOTRACK
end
```

IP Addresses With Same MAC Not in SISF Table

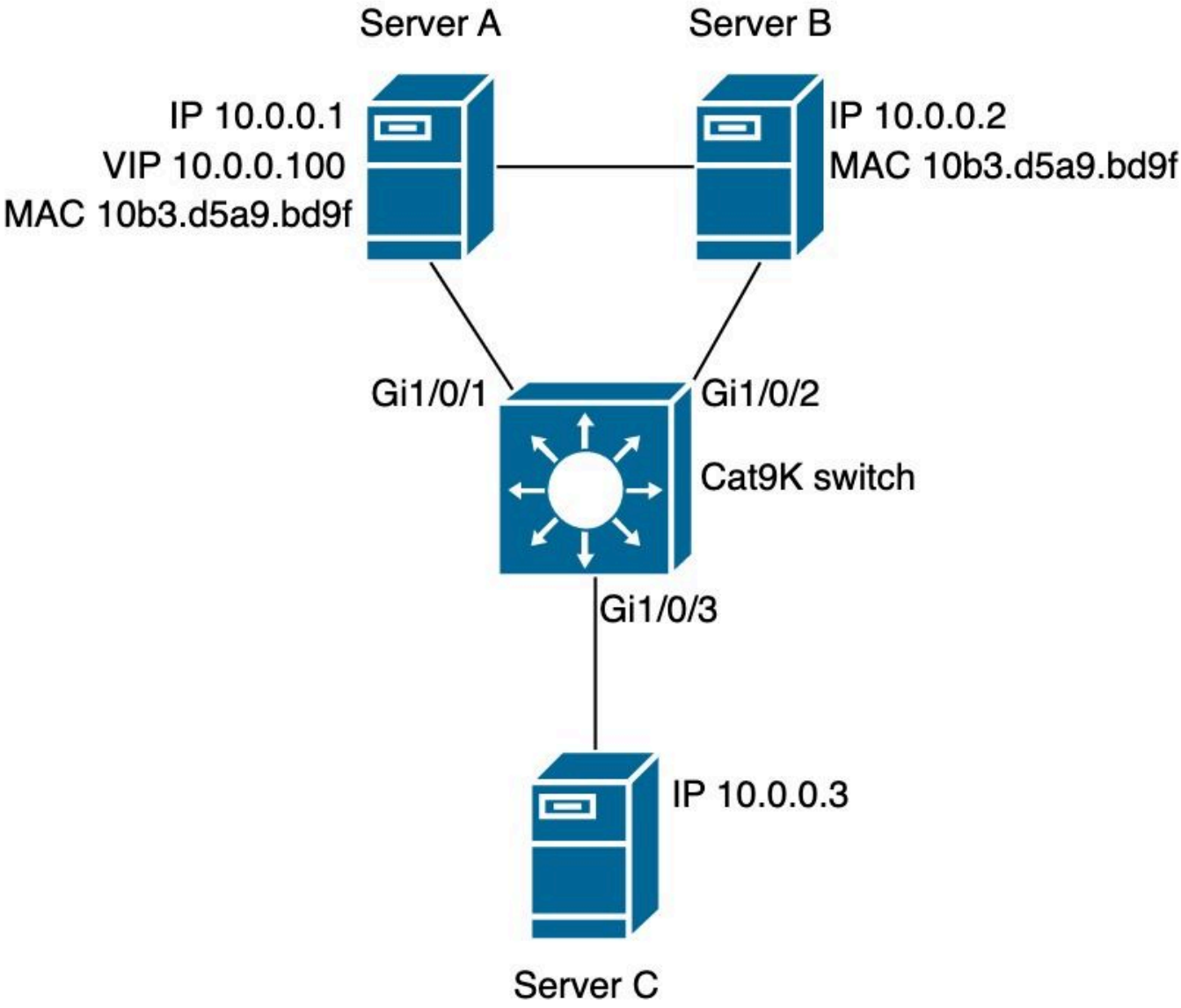
Problem

This scenario is common on appliances in HA (high availability) mode that have different IP addresses, but shares the same MAC address. It is also observed on VM environments that shares the same condition (single MAC address for two or more IP addresses). This condition prevents network connectivity to all those IPs that do not have an entry in the SISF table when custom SISF policy in guard mode is in place.As per SISF feature, only one IP is learnt per MAC address.

Note: This issue is present on 17.7.1 and onward releases

Example:

- IP 10.0.0.1 with MAC address 10b3.d5a9.bd9f is learnt on SISF table and allowed to communicate with the network device 10.0.0.3.
- However second IP 10.0.0.2 and Virtual IP 10.0.0.100 which share MAC address 10b3.d659.7858 is not programmed in SISF table, and communication with the network is not allowed.



SISF policy

<#root>

```
switch#
```

show run | sec IPDT_POLICY

```
device-tracking policy IPDT_POLICY
 no protocol udp
 tracking enable

switch#
```

show device-tracking policy IPDT_POLICY

```
Device-tracking policy IPDT_POLICY configuration:

  security-level guard <-- default mode

  device-role node
  gleaning from Neighbor Discovery
  gleaning from DHCP6
```

```
  gleaning from ARP
  gleaning from DHCP4
  NOT gleaning from protocol unkn
  tracking enable
Policy IPDT_POLICY is applied on the following targets:
Target               Type  Policy               Feature        Target range

Gi1/0/1              PORT  IPDT_POLICY          Device-tracking vlan all
Gi1/0/2              PORT  IPDT_POLICY          Device-tracking vlan all
```

SISF Database

<#root>

switch#

show device-tracking database

```
Binding Table has 2 entries, 2 dynamic (limit 200000)
Codes: L - Local, S - Static, ND - Neighbor Discovery, ARP - Address Resolution Protocol, DH4 - IPv4 DH
Preflevel flags (prlvl):
0001:MAC and LLA match     0002:Orig trunk            0004:Orig access
0008:Orig trusted trunk    0010:Orig trusted access   0020:DHCP assigned
0040:Cga authenticated     0080:Cert authenticated    0100:Statically assigned

    Network Layer Address                  Link Layer Address     Interface  vlan       prlvl      ag

ARP 10.0.0.3                               10b3.d659.7858         Gi1/0/3    10         0005       90s
ARP 10.0.0.1                               10b3.d5a9.bd9f         Gi1/0/1    10         0005       84s
```

Reachability test Server A

<#root>

ServerA#

ping 10.0.0.3 source 10.0.0.1

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.0.0.3, timeout is 2 seconds:
Packet sent with a source address of 10.0.0.1
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/1/1 ms

ServerA#
```

ping 10.0.0.3 source 10.0.0.100 <-- entry for 10.0.0.100 is not on SISF table

```
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.0.0.3, timeout is 2 seconds:
Packet sent with a source address of 10.0.0.100
.....
```

Reachability test Server B.

<#root>

ServerB#

ping 10.0.0.3 <-- entry for 10.0.0.2 is not on SISF table

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.0.0.3, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)

Validating drops on switch.

<#root>

switch(config)#

device-tracking logging

Logs

<#root>

switch#

show logging

<omitted output>
%SISF-4-PAK_DROP: Message dropped

IP=10.0.0.100 VLAN=10 MAC=10b3.d5a9.bd9f

I/F=Gi1/0/1

 P=ARP Reason=Packet accepted but not forwarded
%SISF-4-PAK_DROP: Message dropped

IP=10.0.0.100 VLAN=10 MAC=10b3.d5a9.bd9f

I/F=Gi1/0/1

 P=ARP Reason=Packet accepted but not forwarded
%SISF-4-PAK_DROP: Message dropped

IP=10.0.0.100 VLAN=10 MAC=10b3.d5a9.bd9f

I/F=Gi1/0/1

 P=ARP Reason=Packet accepted but not forwarded

%SISF-4-PAK_DROP: Message dropped

IP=10.0.0.100 VLAN=10 MAC=10b3.d5a9.bd9f

I/F=Gi1/0/1

P=ARP Reason=Packet accepted but not forwarded
%SISF-4-PAK_DROP: Message dropped

IP=10.0.0.100 VLAN=10 MAC=10b3.d5a9.bd9f

I/F=Gi1/0/1

P=ARP Reason=Packet accepted but not forwarded
<omitted output>
%SISF-4-PAK_DROP: Message dropped

IP=10.0.0.2 VLAN=10 MAC=10b3.d5a9.bd9f

I/F=Gi1/0/2

P=ARP Reason=Packet accepted but not forwarded
%SISF-4-MAC_THEFT:

MAC Theft IP=10.0.0.2 VLAN=10 MAC=10b3.d5a9.bd9f IF=Gi1/0/1 New I/F=Gi1/0/2

%SISF-4-PAK_DROP: Message dropped

IP=10.0.0.2 VLAN=10 MAC=10b3.d5a9.bd9f

I/F=Gi1/0/2

P=ARP Reason=Packet accepted but not forwarded
%SISF-4-MAC_THEFT:

MAC Theft IP=10.0.0.2 VLAN=10 MAC=10b3.d5a9.bd9f IF=Gi1/0/1 New I/F=Gi1/0/2

%SISF-4-PAK_DROP: Message dropped

IP=10.0.0.2 VLAN=10 MAC=10b3.d5a9.bd9f

I/F=Gi1/0/2

P=ARP Reason=Packet accepted but not forwarded
%SISF-4-MAC_THEFT:

MAC Theft IP=10.0.0.2 VLAN=10 MAC=10b3.d5a9.bd9f IF=Gi1/0/1 New I/F=Gi1/0/2

%SISF-4-PAK_DROP: Message dropped

IP=10.0.0.2 VLAN=10 MAC=10b3.d5a9.bd9f

I/F=Gi1/0/2

P=ARP Reason=Packet accepted but not forwarded

```
%SISF-4-MAC_THEFT:
```

MAC Theft IP=10.0.0.2 VLAN=10 MAC=10b3.d5a9.bd9f IF=Gi1/0/1 New I/F=Gi1/0/2

```
%SISF-4-PAK_DROP: Message dropped
```

IP=10.0.0.2 VLAN=10 MAC=10b3.d5a9.bd9f

I/F=Gi1/0/2

```
 P=ARP Reason=Packet accepted but not forwarded
%SISF-4-MAC_THEFT:
```

MAC Theft IP=10.0.0.2 VLAN=10 MAC=10b3.d5a9.bd9f IF=Gi1/0/1 New I/F=Gi1/0/2

Solution

Option 1: Remove the IPDT policy from the port allows ARP packets and affected devices become reachable

<#root>

```
switch(config)#interface gigabitEthernet 1/0/1
switch(config-if)#
```

no device-tracking attach-policy IPDT_POLICY

```
switch(config-if)#interface gigabitEthernet 1/0/2
switch(config-if)#
```

no device-tracking attach-policy IPDT_POLICY

Option 2: Remove protocol arp gleaning from the device-tracking policy.

<#root>

```
switch(config)#device-tracking policy IPDT_POLICY
switch(config-device-tracking)#
```

no protocol arp

Option 3: Change the security-level of IPDT_POLICY to glean.

<#root>

```
switch(config)#device-tracking policy IPDT_POLICY
switch(config-device-tracking)#
```

security-level glean

Related Information

- Security Configuration Guide, Cisco IOS XE Bengaluru 17.6.x (Catalyst 9300 Switches): Configuring Switch Integrated Security Features
- Security Configuration Guide, Cisco IOS XE Cupertino 17.9.x (Catalyst 9300 Switches): Configuring Switch Integrated Security Features
- Cisco Catalyst 9000 Family Switch Integrated Security Features (SISF) White Paper
- Cisco bug ID CSCvx75602 - SISF memory leak in AR-relay and ND-suppression
- Cisco bug ID CSCwf33293 - [EVPN SISF] Custom method required to modify the limit address values for IPv4/V6 with EVPN + DHCP
- Cisco bug ID CSCvq22011 - IOS-XE drops ARP reply when IPDT gleans from ARP
- Cisco bug ID CSCwc20488 - 255 pseudo-ports limitation per vlan/evi
- Cisco bug ID CSCwh52315 - 9300 switch drops ARP reply when having an IPDT policy in the port
- Cisco bug ID CSCvd51480 - Unbinding ip dhcp snooping and device-tracking