Troubleshoot High CPU on Catalyst 9000 Caused by SISF Process

Contents

Introduction

Prerequisites

- Requirements
- Components Used

Background Information

Problem

- Step 1: Check CPU Utilization
- Step 2: Check Device Tracking Database
- Step 3: Check Etherchannels
- Step 3: Check CDP Neighbor

Solution

- Step 1: Configure Device Tracking Policy
- Step 2: Attach the Policy to the Trunk Interface

Related Information

Introduction

This document describes the high CPU utilization on Cisco Catalyst 9000 series switches caused by Switch Integrated Security Features process.

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- Basic understanding of LAN switching technology
- Knowledge of Cisco Catalyst 9000 series switches
- Familiarity with Cisco IOS® XE command-line interface (CLI)
- Familiarity with Device Tracking feature

Components Used

The information in this document is based on these software and hardware versions:

- Cisco Catalyst 9000 series switches
- Software Version: All versions

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.

Background Information

Switch Integrated Security Features (SISF) is a framework developed to optimize security in Layer 2 domains. It merges the IP Device Tracking (IPDT) and *certain* IPv6 first-hop security (FHS) functionality, to simplify the migration from IPv4 to IPv6 stack or a dual-stack.

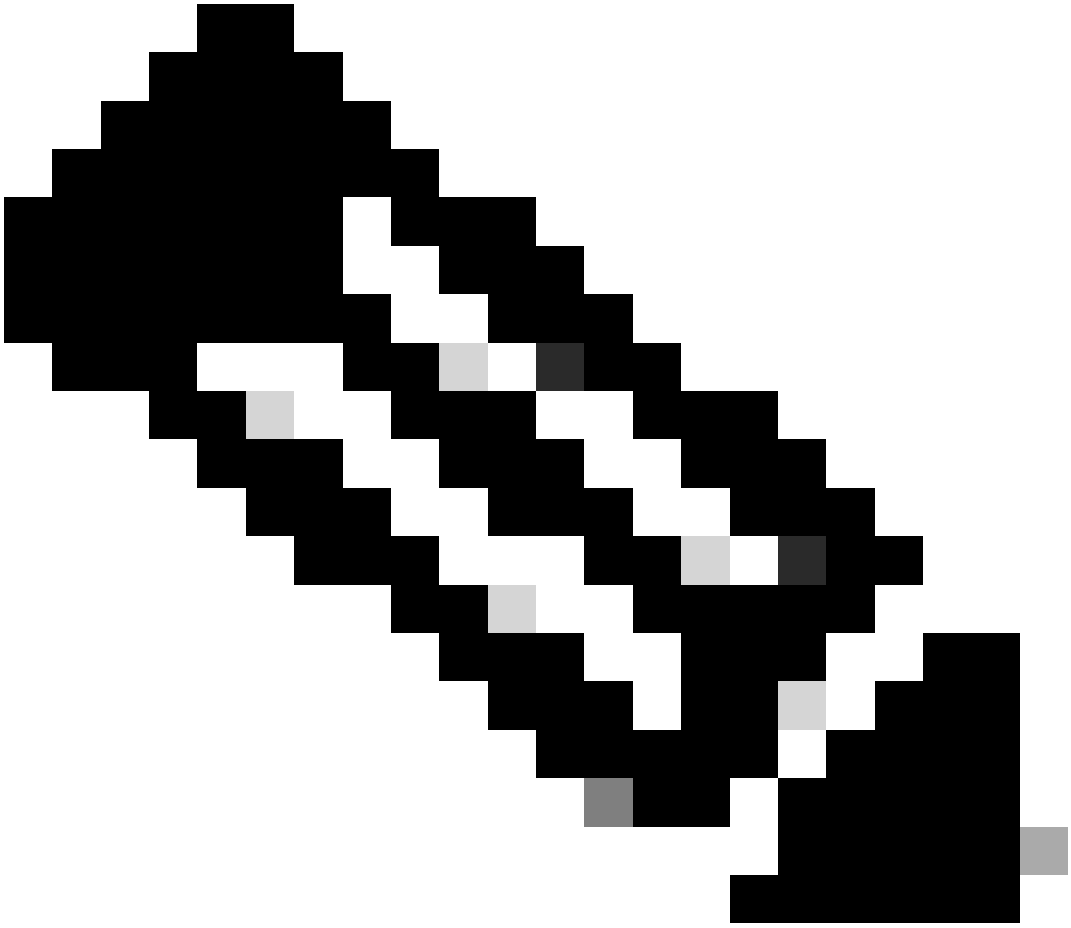
This section provides an overview of the high CPU utilization issue observed on Cisco Catalyst 9000 series switches caused by SISF process. The issue is identified through specific CLI commands and is related to device tracking on trunk interfaces.

Problem

The keepalive probe sent by the switch is broadcast out of all ports when SISF 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.

The problem addressed in this document is high CPU utilization on Cisco Catalyst 9000 series switches caused by SISF process.



Note: Be aware that SISF dependent features such as DHCP snooping enables SISF, which can trigger this problem.

Step 1: Check CPU Utilization

To identify the high CPU utilization, use the this command:

<#root>

device#

show processes cpu sorted

```
CPU utilization for five seconds: 93%/6%; one minute: 91%; five minutes: 87%
 PID Runtime(ms)     Invoked      uSecs   5Sec   1Min   5Min TTY Process
 439     3560284      554004       6426 54.81% 52.37% 47.39%   0 SISF Main Thread


 438     2325444      675817       3440 22.67% 25.17% 26.15%   0
```

```
SISF Switcher Th

 104      548861       84846       6468 10.76%  8.17%  7.51%   0 Crimson flush tr
 119      104155      671081        155  1.21%  1.27%  1.26%   0 IOSXE-RP Punt Se
<SNIP>
```

Step 2: Check Device Tracking Database

Use the this command to check the device tracking database:

```
<#root>

device#

show device-tracking database


Binding Table has 2188 entries, 2188 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 192.168.187.204                        c815.4ef1.d457         Po1        602        0005       546
ARP 192.168.186.161                        4c49.6c7b.6722         Po1        602        0005       171
ARP 192.168.186.117                        4c5f.702b.61eb         Po1        602        0005       459
ARP 192.168.185.254                        20c1.9bac.5765         Po1        602        0005       546
ARP 192.168.184.157                        c815.4eeb.3d04         Po1        602        0005       3mn

ARP 192.168.1.2                            0004.76e0.cff8         Gi1/0/19   901        0005       234

ARP 192.168.152.97                         001c.7f3c.fd08         Po1        620        0005       54s
ARP 169.254.242.184                        1893.4125.9c57         Po1        602        0005       209
ARP 169.254.239.56                         4c5f.702b.61ff         Po1        602        0005       142
ARP 169.254.239.4                          8c17.59c8.fff0         Po1        602        0005       223
ARP 169.254.230.139                        70d8.235f.2a08         Po1        600        0005       6mn
ARP 169.254.229.77                         4c5f.7028.4231         Po1        602        0005       107

<SNIP>
```

It is evident that there are multiple MAC addresses tracked in the Po1 interface. This is not expected if this device is acting as an access switch and there is a end device connected to the interface.

You can check the members of the port channel using this command:

Step 3: Check Etherchannels

```
<#root>
```

```
device#
```

show etherchannel summary

```
Flags:  D - down        P - bundled in port-channel
        I - stand-alone s - suspended
        H - Hot-standby (LACP only)
        R - Layer3      S - Layer2
        U - in use      f - failed to allocate aggregator

        M - not in use, minimum links not met
        u - unsuitable for bundling
        w - waiting to be aggregated
        d - default port

        A - formed by Auto LAG


Number of channel-groups in use: 1
Number of aggregators:           1

Group  Port-channel  Protocol    Ports
------+-------------+-----------+-----------------------------------------------
```

1 Po1(SU) LACP Te1/1/1(P) Te2/1/1(P)

Step 3: Check CDP Neighbor

Use the this command to check the CDP neighbor:

```
<#root>

device#
```

show cdp neighbor

```
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
                  S - Switch, H - Host, I - IGMP, r - Repeater, P - Phone,
                  D - Remote, C - CVTA, M - Two-port Mac Relay

Device ID        Local Intrfce     Holdtme    Capability  Platform  Port ID
```

C9500 Ten 2/1/1 132 R S C9500-48Y Twe 2/0/16
C9500 Ten 1/1/1 165 R S C9500-48Y Twe 1/0/16

A Catalyst 9500 switch is visibly connected on the other side. This could be another access devices in daisy chain configuration or a distribution/core switch. In any case, this devices cannot be tracking MAC addresses on a trunk interfaces.

Solution

The high CPU utilization issue is caused by device tracking. Disable device tracking on the trunk interfaces.

To do this, create a device tracking policy and attach it to the trunk interfaces:

Step 1: Configure Device Tracking Policy

Create a device tracking policy to treat trunk interfaces as trusted ports:

```
<#root>
device#
configure terminal

device(config)#
device-tracking policy DT_trunk_policy

device(config-device-tracking)#
trusted-port

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

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

Step 2: Attach the Policy to the Trunk Interface

```
<#root>
device#
cconfigure terminal

device(config)#
interface Po1

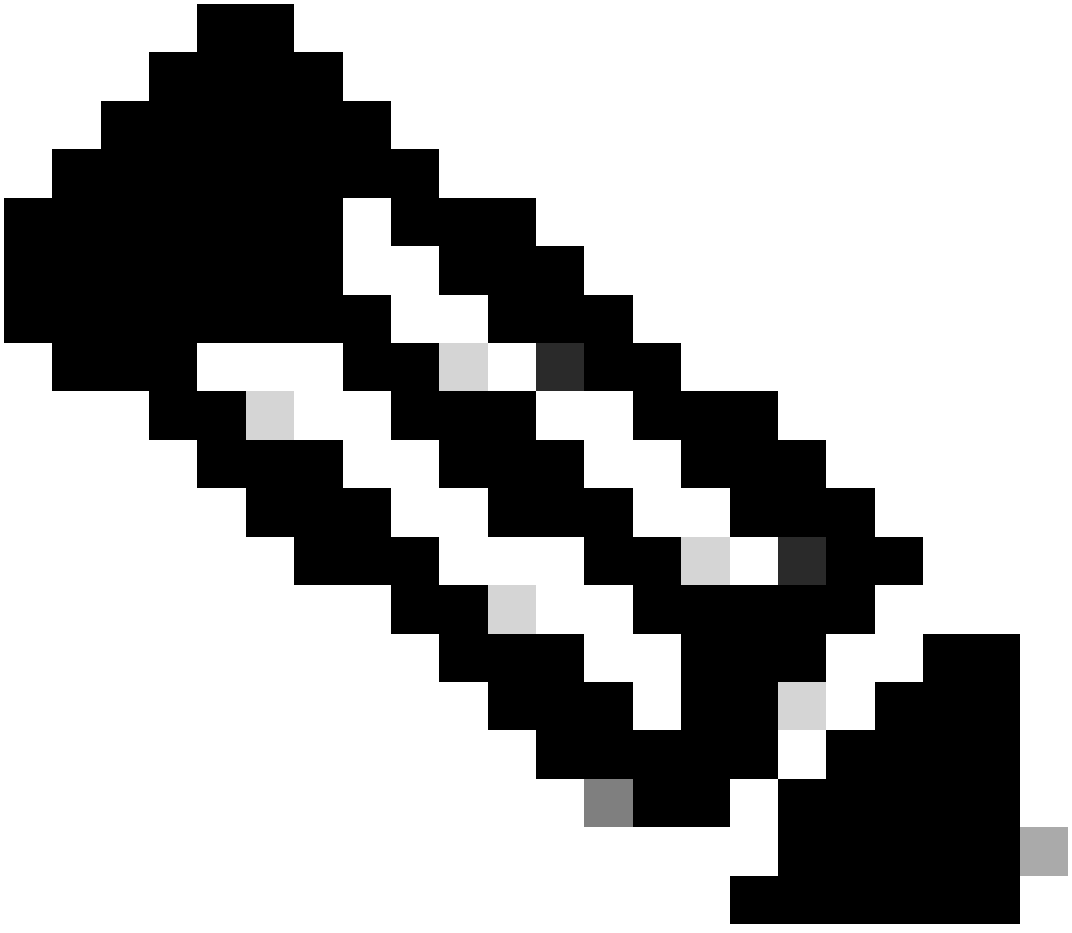
device(config-if)#
device-tracking attach-policy DT_trunk_policy
device(config-if)#
end
```

- The **device-role switch** and **trusted-port** options help you design an efficient and scalable secure

zone. When used together, these two parameters help you achieve an efficient distribution of the creation of entries in the binding table. This keeps the binding tables size under control.

- The **trusted-port** option: Disables the guard function on configured targets. Bindings learned through a trusted-port have preference over bindings learned through any other port. A trusted port is also given preference in case of a collision while making an entry in the table.
- The **device-role** option: Indicates the type of device that is facing the port and this can be a node or a switch. To allow the creation of binding entries for a port, you configure the device as a node. To stop the creation of binding entries, you configure the device as switch.

Configuring the device as a switch is suited to multiple switch set-ups, where the possibility of large device tracking tables is very high. Here, a port facing a device (an uplink trunk port) can be configured to stop creating binding entries, and the traffic arriving at such a port can be trusted, because the switch on the other side of the trunk port has device-tracking enabled and it has checked the validity of the binding entry.



Note: While there are scenarios where configuring only either one of these options can be suitable, the more common use case is for both the trusted-port and device-role switch options to be configured on the port.

Related Information

- Cisco Technical Support & Downloads
- Troubleshoot SISF on Catalyst 9000 Series Switches
- Security Configuration Guide, Cisco IOS XE Dublin 17.12.x (Catalyst 9300 Switches)