Troubleshoot IPsec Issues for Service Tunnels on vEdges with IKEv2

Contents

Introduction

This document describes how to troubleshoot the most common issues for Internet Protocol security (IPsec) tunnels to third-party devices with Internet Key Exchange version 2 (IKEv2) configured. Most commonly referenced as Service/ Transport Tunnels on Cisco SD-WAN documentation. This document also explains how to enable and read IKE debugs and associate them to the packet exchange to understand the point of failure on an IPsec negotiation.

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- IKEv2
- IPsec negotiation
- Cisco SD-WAN

Components Used

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.

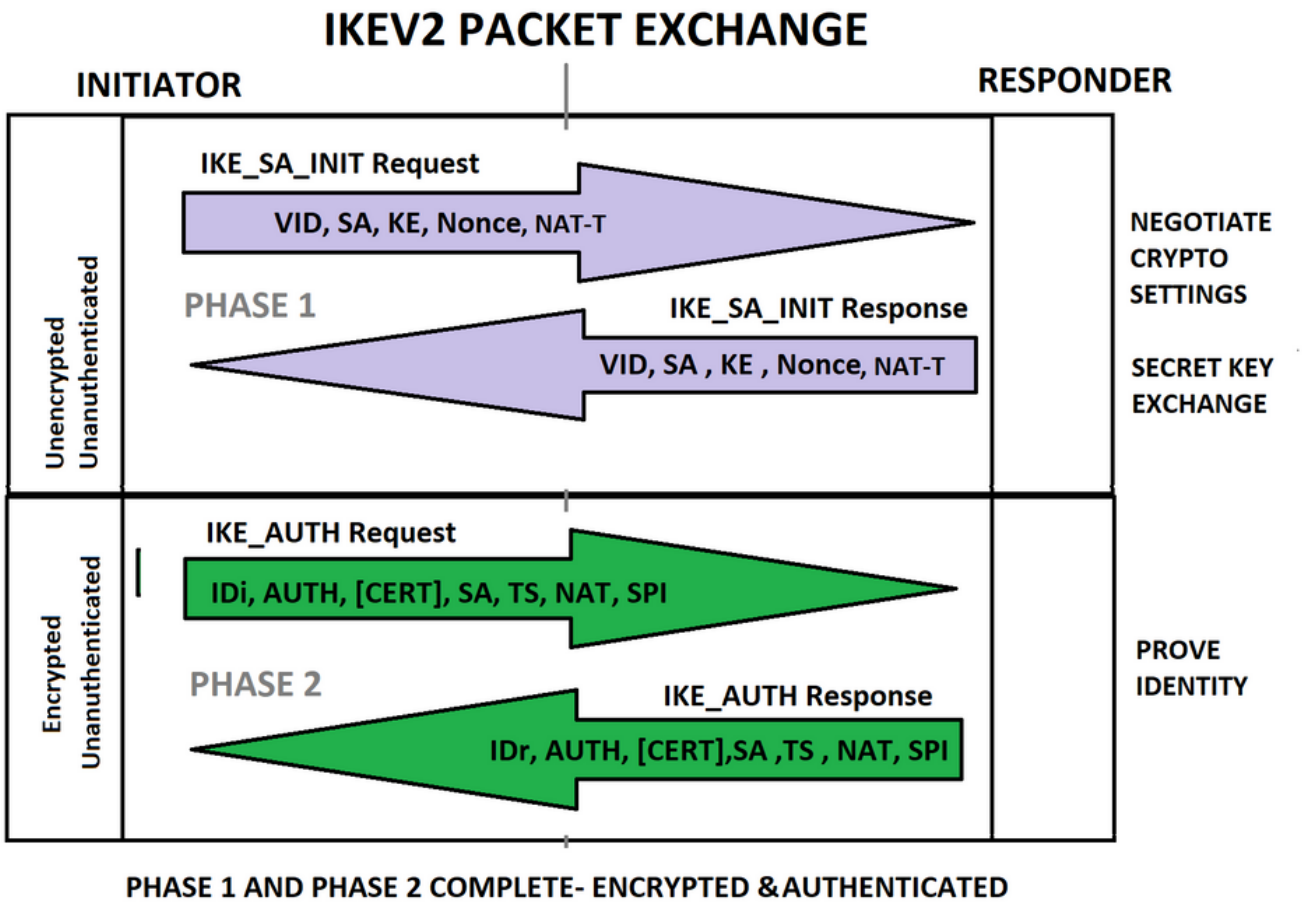
Background Information

IKE Glossary

- **Internet Protocol security (IPsec)** is a standard suite of protocols between 2 communication points across the IP network that provide data authentication, integrity, and confidentiality.
- **Internet Key Exchange version 2 (IKEv2) is the protocol used to set up a security association (SA) in the IPsec protocol suite.**
- A **security association** (**SA**) is the establishment of shared security attributes between two network entities to support secure communication. An SA can include attributes such as cryptographic algorithm and mode; traffic encryption key; and parameters for the network data to be passed over the connection.
- The **vendor IDs** (**VID**) are used to identify peer devices with the same vendor implementation in order to support vendor-specific features.
- **Nonce**: random values created in the exchange to add randomness and prevent replay attacks.
- **Key-exchange (KE)** information for the Diffie-Hellman (DH) secure key-exchange process.
- **Identity Initiator/responder (IDi/IDr)** is used to send out authentication information to the peer. This information is transmitted under the protection of the common shared secret.
- The IPSec shared key can be derived with the use of DH again to ensure **Perfect Forward Secrecy (PFS)** or with a refresh of the shared secret derived from the original DH exchange.
- **Diffie–Hellman (DH) key exchange is a method of securely cryptographic algorithms exchange over a public channel.**
- **Traffic Selectors (TS)** are the proxy identities or traffic exchanged on the IPsec negotiation to pass through the tunnel encrypted.

IKEv2 Packet Exchange

Each IKE packet contains payload information for the tunnel establishment. The IKE glossary explains the abbreviations shown on this image as part of the payload content for the packet exchange.

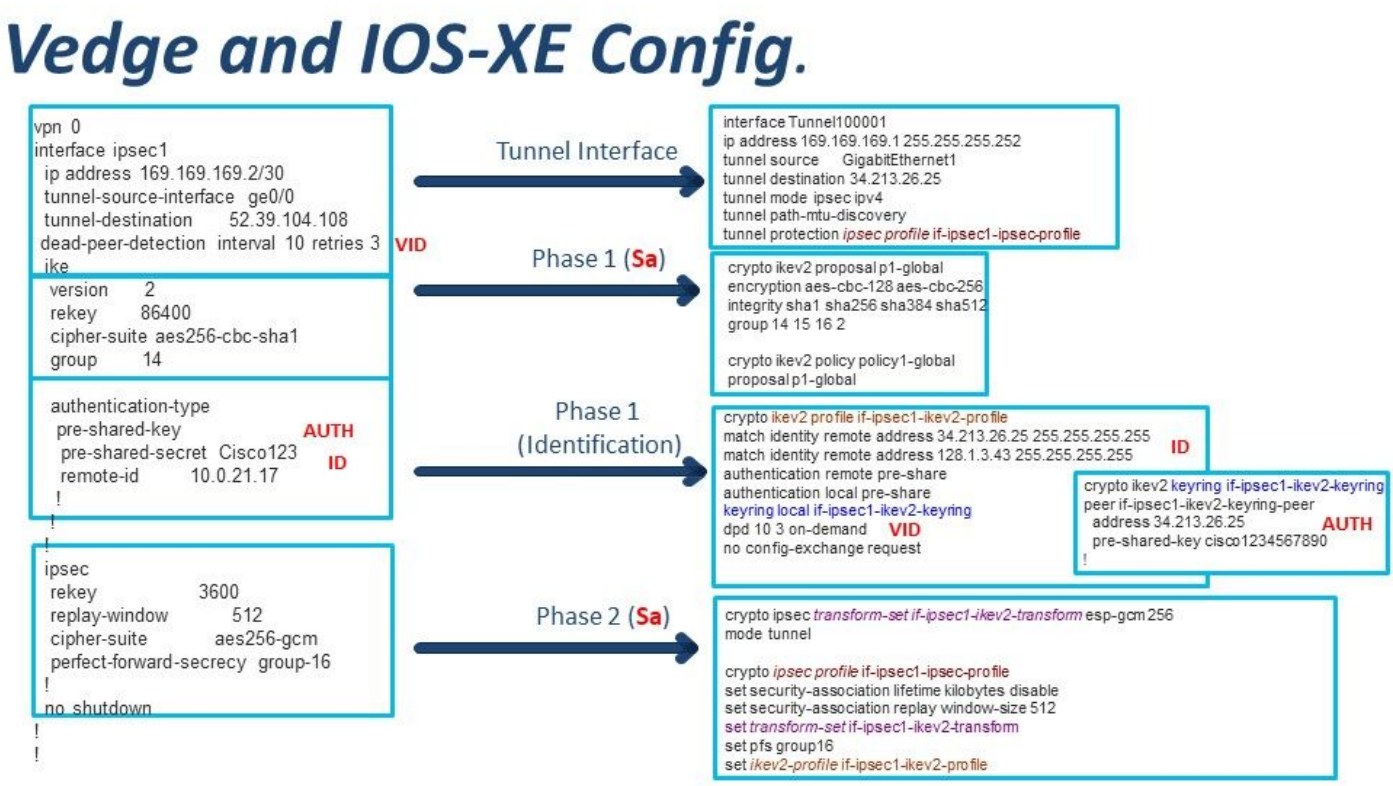


IKEV2-Exchange

> **Note**: It is important to verify on what packet exchange of the IKE negotiation the IPsec Tunnel fails to quickly analyze what configuration is involved to address the issue effectively.

> **Note:** This document does not describe deeper the IKEv2 Packet exchange. For more references, navigate to IKEv2 Packet Exchange and Protocol Level Debugging

It is needed to correlate the vEdge configuration against the Cisco IOS® XE config. Also, it is useful to match the IPsec concepts and the payload content for IKEv2 packet exchanges as shown in the image.



Note: Each part of the configuration modifies an aspect of the IKE negotiation exchange. It is important to correlate the commands to the protocol negotiation of IPsec.

Troubleshoot

Enable IKE debugs

On vEdges **debug iked** enables debug level information either IKEv1 or IKEv2.

```
debug iked misc high
debug iked event high
```

It is possible to display the current debug information within **vshell** and run the command **tail -f <debug path>**.

```
vshell
tail -f /var/log/message
```

In CLI is also possible to display the current logs/debug information for the path specified.

```
monitor start /var/log/messages
```

Tips to Start the Troubleshoot Process for IPsec Issues

It is possible to separate three different IPsec scenarios. It is a good point of reference to identify the symptom to have a better approach to know how to start.

1. IPsec tunnel does not establish.
2. IPsec tunnel went down and it re-established on its own. (Flapped)

3. IPsec tunnel went down and it stays on a downstate.

For the IPsec tunnel does not establish symptoms, it is needed to debug in real-time to verify what is the current behavior on the IKE negotiation.

For IPsec tunnel went down and it re-established on its own symptoms, most commonly known as tunnel Flapped and the root cause analysis (RCA) is needed. It is indispensable to know the timestamp when the tunnel went down or have an estimated time to look at the debugs.

For IPsec tunnel went down and it stays on downstate symptoms, it means the tunnel worked before but for any reason, it came down and we need to know the teardown reason and the current behavior that prevents the tunnel to be successfully established again.

Identify the points before the troubleshoot starts:

1. IPsec tunnel (Number) with issues and configuration.
2. The timestamp when the tunnel went down (if applicable).
3. IPsec peer IP address (Tunnel destination).

All the debugs and logs are saved on **/var/log/messages** files, for the current logs, they are saved on messages file but for this specific symptom the flap could be identified hours/days after the issue, most probably debugs related would be on messages1,2,3..etc. It is important to know the timestamp to look at the right message file and analyze the debugs (charon) for the IKE negotiation of the IPsec Tunnel related.

Most of the debugs do not print the number of the IPsec tunnel. The most frequent way to identify the negotiation and packets is with the IP address of the remote peer and the IP address where the tunnel is sourced on the vedge. Some examples of IKE debugs printed:

```
Jun 18 00:31:22 vedge01 charon: 09[CFG] vici initiate 'child_IPsec2_1'
Jun 18 00:31:22 vedge01 charon: 16[IKE] initiating IKE_SA ipsec2_1[223798] to 10.10.10.1
Jun 18 00:31:22 vedge01 charon: 16[IKE] initiating IKE_SA ipsec2_1[223798] to 10.10.10.1
```

The debugs for the IKE INIT negotiation show the IPsec Tunnel number, However, the subsequent information for packet exchange only uses the IPsec tunnel IP addresses.

```
Jun 18 00:31:22 vedge01 charon: 09[CFG] vici initiate 'child_ipsec2_1'
Jun 18 00:31:22 vedge01 charon: 16[IKE] initiating IKE_SA ipsec2_1[223798] to 10.10.10.1
Jun 18 00:31:22 vedge01 charon: 16[IKE] initiating IKE_SA ipsec2_1[223798] to 10.10.10.1
Jun 18 00:31:22 vedge01 charon: 16[ENC] generating IKE_SA_INIT request 0 [ SA KE No N(NATD_S_IP)
N(NATD_D_IP) N(FRAG_SUP) N(HASH_ALG) N(REDIR_SUP) ]
Jun 18 00:31:22 vedge01 charon: 16[NET] sending packet: from 10.132.3.92[500] to 10.10.10.1[500]
(464 bytes)
Jun 18 00:31:22 vedge01 charon: 12[NET] received packet: from 10.10.10.1[500] to
10.132.3.92[500] (468 bytes)
Jun 18 00:31:22 vedge01 charon: 12[ENC] parsed IKE_SA_INIT response 0 [ SA KE No N(NATD_S_IP)
N(NATD_D_IP) N(HTTP_CERT_LOOK) N(FRAG_SUP) V ]
Jun 18 00:31:22 vedge01 charon: 12[ENC] received unknown vendor ID:
4f:85:58:17:1d:21:a0:8d:69:cb:5f:60:9b:3c:06:00
Jun 18 00:31:22 vedge01 charon: 12[IKE] local host is behind NAT, sending keep alives
```

IPsec tunnel configuration:

```
interface ipsec2 ip address 192.168.1.9/30 tunnel-source 10.132.3.92 tunnel-destination
10.10.10.1 dead-peer-detection interval 30 ike version 2 rekey 86400 cipher-suite aes256-cbc-
sha1 group 14 authentication-type pre-shared-key pre-shared-secret
$8$wgrs/Cw6tX0na34yF4Fga0B62mGBpHFdOzFaRmoYfnBioWVO3s3efFPBbkaZqvoN ! ! ! ipsec rekey 3600
replay-window 512 cipher-suite aes256-gcm perfect-forward-secrecy group-14 !
```

Symptom 1. IPsec Tunnel Does Not Get Established

As the issue can be the first implementation for the tunnel, it has not been up and the IKE debugs are the best option.

Symptom 2. IPsec Tunnel Went Down and It Was Re-established on Its Own

As previously mentioned, usually this symptom is addressed to know the root cause of why the tunnel went down. With the root cause analysis known, sometimes, the network's admin prevents further issues.

Identify the points before the troubleshoot starts:

1. IPsec tunnel (Number) with issues and configuration.
2. The timestamp when the tunnel went down.
3. IPsec peer IP address (Tunnel destination)

DPD Retransmissions

In this example, the tunnel went down on Jun 18 at 00:31:17.

```
Jun 18 00:31:17 vedge01 FTMD[1472]: %Viptela-vedge01-FTMD-6-INFO-1000001: VPN 1 Interface ipsec2
DOWN
Jun 18 00:31:17 vedge01 FTMD[1472]: %Viptela-vedge01-ftmd-6-INFO-1400002: Notification:
interface-state-change severity-level:major host-name:"vedge01" system-ip:4.0.5.1 vpn-id:1 if-
name:"ipsec2" new-state:down
```

> **Note**: The logs for IPsec tunnel down are not part of iked debugs, they are *FTMD* logs. Therefore, neither *charon* nor *IKE* would be printed.

> **Note**: The related logs are not usually together printed, there be more information between them not related to the same process.

Step 1. After the timestamp is identified and the time and the logs are correlated, start to review the logs from bottom to top.

```
Jun 18 00:31:17 vedge01 charon: 11[IKE] giving up after 3 retransmits

Jun 18 00:28:22 vedge01 charon: 08[IKE] retransmit 3 of request with message ID 543 (tries=3,
timeout=30, exchange=37, state=2)
Jun 18 00:28:22 vedge01 charon: 08[NET] sending packet: from 10.132.3.92[4500] to
10.10.10.1[4500] (76 bytes)

Jun 18 00:26:45 vedge01 charon: 06[IKE] retransmit 2 of request with message ID 543 (tries=3,
timeout=30, exchange=37, state=2)
```

```
Jun 18 00:26:45 vedge01 charon: 06[NET] sending packet: from 10.132.3.92[4500] to
10.10.10.1[4500] (76 bytes)

Jun 18 00:25:21 vedge01 charon: 08[IKE] sending DPD request
Jun 18 00:25:21 vedge01 charon: 08[ENC] generating INFORMATIONAL request 543 [ ]
Jun 18 00:25:21 vedge01 charon: 08[NET] sending packet: from 10.132.3.92[4500] to
10.10.10.1[4500] (76 bytes)
Jun 18 00:25:51 vedge01 charon: 05[IKE] retransmit 1 of request with message ID 543 (tries=3,
timeout=30, exchange=37, state=2)
Jun 18 00:25:51 vedge01 charon: 05[NET] sending packet: from 10.132.3.92[4500] to
10.10.10.1[4500] (76 bytes)
```

The last successful DPD packet exchange is described as request # 542.

```
Jun 18 00:24:08 vedge01 charon: 11[ENC] generating INFORMATIONAL request 542 [ ]
Jun 18 00:24:08 vedge01 charon: 11[NET] sending packet: from 10.132.3.92[4500] to
10.10.10.1[4500] (76 bytes)
Jun 18 00:24:08 vedge01 charon: 07[NET] received packet: from 13.51.17.190[4500] to
10.10.10.1[4500] (76 bytes)
Jun 18 00:24:08 vedge01 charon: 07[ENC] parsed INFORMATIONAL response 542 [ ]
```

Step 2. Put all the information together in the right order:

```
Jun 18 00:24:08 vedge01 charon: 11[ENC] generating INFORMATIONAL request 542 [ ]
Jun 18 00:24:08 vedge01 charon: 11[NET] sending packet: from 10.132.3.92[4500] to
10.10.10.1[4500] (76 bytes)
Jun 18 00:24:08 vedge01 charon: 07[NET] received packet: from 10.10.10.1[4500] to
10.132.3.92[4500] (76 bytes)
Jun 18 00:24:08 vedge01 charon: 07[ENC] parsed INFORMATIONAL response 542 [ ]

Jun 18 00:25:21 vedge01 charon: 08[IKE] sending DPD request
Jun 18 00:25:21 vedge01 charon: 08[ENC] generating INFORMATIONAL request 543 [ ]
Jun 18 00:25:21 vedge01 charon: 08[NET] sending packet: from 10.132.3.92[4500] to
10.10.10.1[4500] (76 bytes)
Jun 18 00:25:51 vedge01 charon: 05[IKE] retransmit 1 of request with message ID 543 (tries=3,
timeout=30, exchange=37, state=2)
Jun 18 00:25:51 vedge01 charon: 05[NET] sending packet: from 10.132.3.92[4500] to
10.10.10.1[4500] (76 bytes)

Jun 18 00:26:45 vedge01 charon: 06[IKE] retransmit 2 of request with message ID 543 (tries=3,
timeout=30, exchange=37, state=2)
Jun 18 00:26:45 vedge01 charon: 06[NET] sending packet: from 10.132.3.92[4500] to
10.10.10.1[4500] (76 bytes)

Jun 18 00:28:22 vedge01 charon: 08[IKE] retransmit 3 of request with message ID 543 (tries=3,
timeout=30, exchange=37, state=2)
Jun 18 00:28:22 Lvedge01 charon: 08[NET] sending packet: from 10.132.3.92[4500] to
10.10.10.1[4500] (76 bytes)

Jun 18 00:31:17 vedge01 charon: 11[IKE] giving up after 3 retransmits
Jun 18 00:31:17 vedge01 FTMD[1472]: %Viptela-LONDSR01-FTMD-6-INFO-1000001: VPN 1 Interface
ipsec2 DOWN
Jun 18 00:31:17 vedge01 FTMD[1472]: %Viptela-LONDSR01-ftmd-6-INFO-1400002: Notification:
interface-state-change severity-level:major host-name:"LONDSR01" system-ip:4.0.5.1 vpn-id:1 if-
name:"ipsec2" new-state:down
```

For the example described, the tunnel goes down due to vEdge01 does not receive the DPD packets from 10.10.10.1. It is expected after 3 DPD retransmissions the IPsec peer is set as "lost" and the tunnel goes down. There are multiple reasons for this behavior, usually, it is related to the ISP where the packets are lost or dropped in the path. If the issue occurs once, there is no way to track the traffic lost, however, if the issue persists, the packet can be tracked with the use of captures on vEdge, remote IPSec peer, and the ISP.

Symptom 3. IPsec Tunnel Went Down and It Stays on a Downstate

As previously mentioned in this symptom, the tunnel previously worked fine but for any reason, it came down and the tunnel has not been able to successfully established again. In this scenario, there is an affectation to the network.

identify the points before the troubleshoot starts:

1. IPsec tunnel (Number) with issues and configuration.
2. The timestamp when the tunnel went down.
3. IPsec peer IP address (Tunnel destination)

PFS Mismatch

In this example, the troubleshoot does not start with the timestamp when the tunnel goes down. As the issue persists, the IKE debugs are the best option.

```
interface ipsec1 description VWAN_VPN ip address 192.168.0.101/30 tunnel-source-interface ge0/0
tunnel-destination 10.10.10.1 ike version 2 rekey 28800 cipher-suite aes256-cbc-sha1 group 2
authentication-type pre-shared-key pre-shared-secret
"$8$njK2pLLjgKWNQu0KecNtY3+fo3hbTs0/7iJy6unNtersmCGjGB38kIPjsoqqXZdVmtizLu79\naQdjt2POM242Yw=="
! ! ! ipsec rekey 3600 replay-window 512 cipher-suite aes256-cbc-sha1 perfect-forward-secrecy
group-16 ! mtu 1400 no shutdown
```

The debug iked is enabled and negotiation is displayed.

```
daemon.info: Apr 27 05:12:56 vedge01 charon: 16[NET] received packet: from 10.10.10.1[4500] to
172.28.0.36[4500] (508 bytes)
daemon.info: Apr 27 05:12:56 vedge01 charon: 16[ENC] parsed CREATE_CHILD_SA request 557 [ SA No
TSi TSr ]
daemon.info: Apr 27 05:12:56 vedge01 charon: 16[CFG] received proposals:
ESP:AES_GCM_16_256/NO_EXT_SEQ, ESP:AES_CBC_256/HMAC_SHA1_96/NO_EXT_SEQ,
ESP:3DES_CBC/HMAC_SHA1_96/NO_EXT_SEQ, ESP:AES_CBC_256/HMAC_SHA2_256_128/NO_EXT_SEQ,
ESP:AES_CBC_128/HMAC_SHA1_96/NO_EXT_SEQ, ESP:3DES_CBC/HMAC_SHA2_256_128/NO_EXT_SEQ
daemon.info: Apr 27 05:12:56 vedge01 charon: 16[CFG] configured proposals:
ESP:AES_CBC_256/HMAC_SHA1_96/MODP_4096/NO_EXT_SEQ
daemon.info: Apr 27 05:12:56 vedge01 charon: 16[IKE] no acceptable proposal found
daemon.info: Apr 27 05:12:56 vedge01 charon: 16[IKE] failed to establish CHILD_SA, keeping
IKE_SA
daemon.info: Apr 27 05:12:56 vedge01 charon: 16[ENC] generating CREATE_CHILD_SA response 557 [
N(NO_PROP) ]
daemon.info: Apr 27 05:12:56 vedge01 charon: 16[NET] sending packet: from 172.28.0.36[4500] to
10.10.10.1[4500] (76 bytes)

daemon.info: Apr 27 05:12:57 vedge01 charon: 08[NET] received packet: from 10.10.10.1[4500] to
172.28.0.36[4500] (76 bytes)
daemon.info: Apr 27 05:12:57 vedge01 charon: 08[ENC] parsed INFORMATIONAL request 558 [ ]
daemon.info: Apr 27 05:12:57 vedge01 charon: 08[ENC] generating INFORMATIONAL response 558 [ ]
daemon.info: Apr 27 05:12:57 vedge01 charon: 08[NET] sending packet: from 172.28.0.36[4500] to
10.10.10.1[4500] (76 bytes)
daemon.info: Apr 27 05:12:58 vedge01 charon: 07[NET] received packet: from 10.10.10.1[4500] to
172.28.0.36[4500] (396 bytes)
daemon.info: Apr 27 05:12:58 vedge01 charon: 07[ENC] parsed CREATE_CHILD_SA request 559 [ SA No
TSi TSr ]
daemon.info: Apr 27 05:12:58 vedge01 charon: 07[CFG] received proposals:
ESP:AES_GCM_16_256/NO_EXT_SEQ, ESP:AES_CBC_256/HMAC_SHA1_96/NO_EXT_SEQ,
ESP:3DES_CBC/HMAC_SHA1_96/NO_EXT_SEQ, ESP:AES_CBC_256/HMAC_SHA2_256_128/NO_EXT_SEQ,
```

```
ESP:AES_CBC_128/HMAC_SHA1_96/NO_EXT_SEQ, ESP:3DES_CBC/HMAC_SHA2_256_128/NO_EXT_SEQ
daemon.info: Apr 27 05:12:58 vedge01 charon: 07[CFG] configured proposals:
ESP:AES_CBC_256/HMAC_SHA1_96/MODP_4096/NO_EXT_SEQ
daemon.info: Apr 27 05:12:58 Avedge01 charon: 07[IKE] no acceptable proposal found
daemon.info: Apr 27 05:12:58 vedge01 charon: 07[IKE] failed to establish CHILD_SA, keeping
IKE_SA
```

> **Note:** CREATE_CHILD_SA packets are exchanged for every rekey or new SA. For more references, navigate to Understanding IKEv2 Packet Exchange

IKE debugs show the same behavior and it is constantly repeated, so it is possible to take a part of the information and analyze it:

CREATE_CHILD_SA means a rekey, with the purpose for the new SPIS to be generated and exchanged between the IPsec endpoints.

- The vedge receives the CREATE_CHILD_SA request packet from 10.10.10.1.
- The vedge processes the request and verifies the proposals (SA) sent by peer 10.10.10.1
- The vedge compares the received proposal sent by the peer against its configured proposals.
- The CREATE_CHILD_SA exchanged fails with " no acceptable proposals found".

At this point, the question is: **Why is there a configuration mismatch if the tunnel worked previously and no changes were done?**

Analyze in deep, there is an extra field on the configured proposals that the peer is not sending.

configured proposals: ESP:AES_CBC_256/HMAC_SHA1_96/MODP_4096/NO_EXT_SEQ

Received proposals:
ESP:AES_GCM_16_256/NO_EXT_SEQ,
ESP:AES_CBC_256/HMAC_SHA1_96/NO_EXT_SEQ,
ESP:3DES_CBC/HMAC_SHA1_96/NO_EXT_SEQ,
ESP:AES_CBC_256/HMAC_SHA2_256_128/NO_EXT_SEQ,
ESP:AES_CBC_128/HMAC_SHA1_96/NO_EXT_SEQ,
ESP:3DES_CBC/HMAC_SHA2_256_128/NO_EXT_SEQ

MODP_4096 is DH group 16, which vedges has configured for PFS (perfect-forward-secrecy) on phase 2 (IPsec section).

PFS is the only mismatch configuration in which the tunnel can be successfully established or not according to who is the initiator or responder in the IKE negotiation. However, when the rekey starts the tunnel is not be able to continue and this symptom can be presented or related to.

vEdge IPSec/Ikev2 Tunnel Not Getting Re-initiated After Being Torn Down Due to a DELETE Event

See Cisco bug ID CSCvx86427 for more information about this behavior.

As the issue perseveres, the IKE debugs are the best options. However, for this particular bug if debugs are enabled no information is displayed neither the terminal nor the message file.

To narrow down this issue and verify if vEdge hits the Cisco bug ID CSCvx86427, it is needed to find the moment when the tunnel goes down.

identify the points before the troubleshoot starts:

1. IPsec tunnel (Number) with issues and configuration.
2. The timestamp when the tunnel went down.
3. IPsec peer IP address (Tunnel destination)

After the timestamp is identified, and the time and logs are correlated, review the logs just before when the tunnel goes down.

```
Apr 13 22:05:21 vedge01 charon: 12[IKE] received DELETE for IKE_SA ipsec1_1[217]
Apr 13 22:05:21 vedge01 charon: 12[IKE] deleting IKE_SA ipsec1_1[217] between
10.16.0.5[10.16.0.5]...10.10.10.1[10.10.10.1]
Apr 13 22:05:21 vedge01 charon: 12[IKE] deleting IKE_SA ipsec1_1[217] between
10.16.0.5[10.16.0.5]...10.10.10.1[10.10.10.1]
Apr 13 22:05:21 vedge01 charon: 12[IKE] IKE_SA deleted
Apr 13 22:05:21 vedge01 charon: 12[IKE] IKE_SA deleted
Apr 13 22:05:21 vedge01 charon: 12[ENC] generating INFORMATIONAL response 4586 [ ]
Apr 13 22:05:21 vedge01 charon: 12[NET] sending packet: from 10.16.0.5[4500] to 10.10.10.1[4500]
(80 bytes)
Apr 13 22:05:21 vedge01 charon: 12[KNL] Deleting SAD entry with SPI 00000e77
Apr 13 22:05:21 vedge01 FTMD[1269]: %Viptela-AZGDSR01-FTMD-6-INFO-1000001: VPN 1 Interface
ipsec1 DOWN
Apr 13 22:05:21 vedge01 FTMD[1269]: %Viptela-AZGDSR01-ftmd-6-INFO-1400002: Notification:
interface-state-change severity-level:major host-name:"vedge01" system-ip:4.1.0.1 vpn-id:1 if-
name:"ipsec1" new-state:down
```

> **Note:** There are multiples DELETES packets on an IPsec negotiation, and the DELETE for CHILD_SA is an expected DELETE for a REKEY process, this issue is seen when a pure IKE_SA DELETE packet is received without any particular IPsec negotiation. That DELETE removes all the IPsec/IKE tunnel.

Related Information

- KEv2 Packet Exchange and Protocol Level Debugging
- The Internet Key Exchange (IKE) - RFC 2409
- IKEv2 - RFC 7296
- Site-to-Site LAN to LAN IPSec Between vEdge and Cisco IOS
- Technical Support & Documentation - Cisco Systems