GETVPN Troubleshoot Guide

Contents

Introduction

This document is intended to present a structured troubleshooting methodology and useful tools to help identify and isolate Group Encrypted Transport VPN (GETVPN) problems and to provide possible solutions.

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- GETVPN
  Official GETVPN Configuration Guide
  Official GETVPN Design and Implementation Guide

- Syslog Server Use

Components Used

This document is not restricted to specific software and hardware 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, make sure that you understand the potential impact of any command.

GETVPN Troubleshooting Methodology

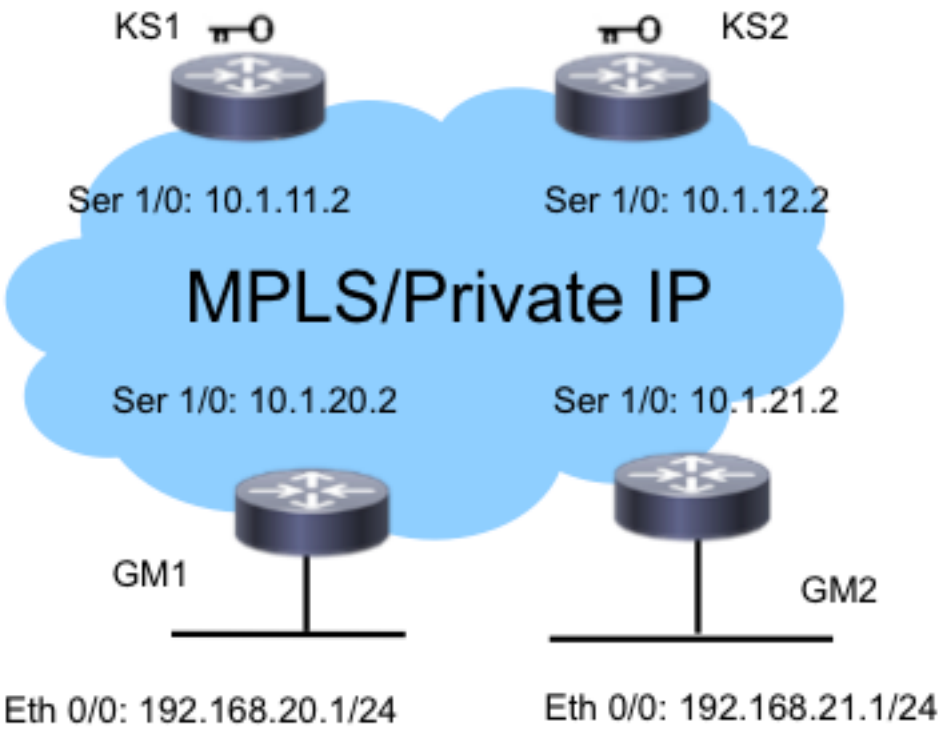
As with most troubleshooting of complex technology problems, the key is to be able to isolate the problem to a specific feature, subsystem, or component. The GETVPN solution is comprised of a number of feature components, specifically:

- Internet Key Exchange (IKE) - Used between Group Member (GM) and Key Server (KS), and amongst Cooperative Protocol (COOP) KSs in order to authenticate and protect the Control Plane.
- Group Domain of Interpretation (GDOI) - Protocol used for the KS in order to distribute group keys and provide key service such as rekey to all the GMs.
- COOP - Protocol used for the KSs in order to communicate with each other and provide redundancy.
- Header Preservation - IPsec in Tunnel mode that preserves the original data packet header for end-to-end traffic delivery.
- Time Based Anti-Replay (TBAR) - Replay detection mechanism used in a group key environment.

It also provides an extensive set of troubleshooting tools in order to ease the troubleshoot process. It is important to understand which of these tools are available, and when they are appropriate for each troubleshooting task. When troubleshooting, it is always a good idea to start with the least intrusive methods so that the production environment is not negatively impacted. The key to this structured troubleshooting is to be able to break the problem down to either a control or data plane issue. You can do this if you follow the protocol or data flow and use the various tools presented here in order to checkpoint them.

Reference Topology

This GETVPN topology and addressing scheme is used throughout the rest of this troubleshooting document.



Reference Configurations

- **KS1**

```
crypto gdoi group G1
identity number 3333
server local
rekey authenmypubkeyrsa get
rekey transport unicast
sa ipsec 1
profile gdoi-p
match address ipv4ENCPOL
address ipv4 10.1.11.2
redundancy
local priority 10
peer address ipv4 10.1.12.2
```

- **GM1**

```
crypto gdoi group G1
identity number 3333
server address ipv4 10.1.11.2
server address ipv4 10.1.12.2
!
crypto map gm_map 10 gdoi
set group G1
!
interface Serial1/0
crypto map gm_map
```

Note: The KS2 and GM2 configurations are not included here for brevity.

Terminology

- **KS** - Key Server
- **GM** - Group Member
- **COOP** - Cooperative Protocol
- **TBAR** - Time Based Anti-Replay
- **KEK** - Key Encryption Key
- **TEK** - Traffic Encryption Key

Logging Facility Preparation and Other Best Practices

Before you begin to troubleshoot, ensure that you have prepared the logging facility as described here. Some best practices are also listed here:

- Check the router amount of free memory, and configure **logging buffered debugging** to a large value (10 MB or more if possible).

- Disable logging to the console, monitor, and syslog servers.

- Retrieve the logging buffer content with the **show log** command at regular intervals, every 20 mins to an hour, in order to prevent log loss due to buffer reuse.

- Whatever happens, enter the **show tech** command from affected GMs and KSs, and examine the output of the **show ip route** command in global and each Virtual Routing and Forwarding (VRF) involved, if any are required.

- Use Network Time Protocol (NTP) in order to sync the clock between all devices that are debugged. Enable millisecond (msec) timestamps for both debug and log messages:

```
service timestamps debug datetime msec
service timestamps log datetime msec
```

- Make sure the show command outputs are timestamped.

```
Router#terminal exec prompt timestamp
```

- When you collect show command outputs for control plane events or data plane counters, always collect multiple iterations of the same output.

Troubleshoot GETVPN Control Plane Issues

Control plane means all the protocol events that led up to the policy and Security Association (SA) creation on the GM so that they are ready to encrypt and decrypt data plane traffic. Some of the key checkpoints in the GETVPN control plane are:

Control Plane Debugging Best Practices

These troubleshooting best practices are not GETVPN specific; they apply to almost any control plane debugging. It is critical to follow these best practices in order to ensure the most effective troubleshooting:

- Turn off console logging and use the logging buffer or syslog in order to collect the debugs.
- Use NTP in order to sync router clocks on all the devices that are debugged.
- Enable msec timestamping for debug and log messages:

```
service timestamp debug datetime msec
service timestamp log datetime msec
```

- Make sure the show command outputs are timestamped so that they can be correlated with the debug output:

```
terminal exec prompt timestamp
```

- Use conditional debugging in a scale environment if possible.

GETVPN Control Plane Troubleshooting Tools

GETVPN Show Commands

As a general rule, these are the command outputs you should collect for almost all GETVPN problems.

KS

```
show crypto gdoi
show crypto gdoi ks coop
show crypto gdoi ks members
show crypto gdoi ks rekey
show crypto gdoi ks policy
```

GM

```
show crypto eli
show crypto gdoi rekey sa
show crypto gdoi
show crypto gdoi gm
show crypto gdoi gm rekey
```

GETVPN Syslog Messages

GETVPN provides an extensive set of syslog messages for significant protocol events and error conditions. The syslog should always be the first place to look when you perform GETVPN troubleshooting.

Common KS Syslog Messages

Syslog Messages	Explanation
COOP_CONFIG_MISMATCH	The configuration between the primary key server and secondary key serv is mismatched.
COOP_KS_ELECTION	The local key server has entered the election process in a group.
COOP_KS_REACH	The reachability between the configured cooperative key servers is restore
COOP_KS_TRANS_TO_PRI	**The local key server transitioned to a primary role from being a seco server in a group.**
COOP_KS_UNAUTH	An authorized remote server tried to contact the local key server in a grou which could be considered a hostile event.
COOP_KS_UNREACH	**The reachability between the configured cooperative key servers is l which could be considered a hostile event.**
KS_GM_REVOKED	During rekey protocol, an unauthorized member tried to join a group, whic could be considered a hostile event.
KS_SEND_MCAST_REKEY	**Sending multicast rekey.**
KS_SEND_UNICAST_REKEY	**Sending unicast rekey.**
KS_UNAUTHORIZED	During GDOI registration protocol, an unauthorized member tried to join a group, which could be considered a hostile event.
UNAUTHORIZED_IPADDR	The registration request was dropped because the requesting device was authorized to join the group.

Common GM Syslog Messages

Syslog Messages	Explanation
GM_CLEAR_REGISTER	The **clear crypto gdoi** command has been executed by the local group member.
GM_CM_ATTACH	A crypto map has been attached for the local group member.
GM_CM_DETACH	A crypto map has been detached for the local group member.&
GM_RE_REGISTER	**IPsec SA created for one group might have been expired or cleare Need to reregister to the key server.**
GM_RECV_REKEY	**Rekey received.**
GM_REGS_COMPL	**Registration complete.**
GM_REKEY_TRANS_2_MULTI	Group member has transitioned from using a unicast rekey mechanism using a multicast mechanism.
GM_REKEY_TRANS_2_UNI	Group member has transitioned from using a multicast rekey mechanis using a unicast mechanism.
PSEUDO_TIME_LARGE	A group member has received a pseudotime with a value that is largely different from its own pseudotime.
REPLAY_FAILED	A group member or key server has failed an anti-replay check.

Note: The messages highlighted in red are the most common or significant messages seen in a GETVPN environment.

Global Crypto and GDOI Debugs

GETVPN debugs are divided:

1. First by the device on which you are troubleshooting.
```
F340.06.15-2900-18#debug cry gdoi ?
 all-features  All features in GDOI
 condition     GDOI Conditional Debugging
 gm            Group Member
 ks            Key Server
```
2. Second by the type of problem you are troubleshooting.
```
GM1#debug cry gdoi gm ?
 all-features    All Group Member features
 infrastructure  GM Infrastructure
 registration    GM messages related to registration
 rekey           GM messagese related to Re-Key
 replay          Anti Replay
```
3. Third by the level of debugging that needs to be enabled. In Version 15.1(3)T and later, all GDOI feature debugs were standardized to have these debug levels. This was designed in order to help troubleshoot large-scale GETVPN environments with enough debugging granularity. When you debug GETVPN problems, it is important to use the appropriate debug level. As a general rule, start with the lowest debug level, that is the error level, and increase the debugging granularity when needed.
```
GM1#debug cry gdoi gm all-features ?
 all-levels  All levels
 detail      Detail level
 error       Error level
 event       Event level
 packet      Packet level
 terse       Terse level
```

GDOI Conditional Debugging

In Cisco IOS® Version 15.1(3)T and later, GDOI conditional debugging was added in order to help troubleshoot GETVPN in a large-scale environment. So all Internet Security Association and Key Management Protocol (ISAKMP) and GDOI debugs can now be triggered with a conditional filter based on the group or peer IP address. For most GETVPN problems, it is good to enable both ISAKMP and GDOI debugs with the appropriate conditional filter, since GDOI debugs only show GDOI-specific operations. In order to use ISAKMP and GDOI conditional debugs, complete these two simple steps:

1. Set the conditional filter.
2. Enable the relevant ISAKMP and GDOI as usual.

For example:

```
KS1# debug crypto gdoi condition peer add ipv4 10.1.20.2
% GDOI Debug Condition added.

KS1#
KS1# show crypto gdoi debug-condition
GDOI Conditional Filters:
Peer Address 10.1.20.2
Unmatched NOT set

KS1#debug crypto gdoi ks registration all-levels
```

```
GDOI Key Server Registration Debug level: (Packet, Detail, Event, Terse, Error)
```

> **Note**: With both ISAKMP and GDOI conditional debugs, in order to catch debug messages that might not have the conditional filter information, for example the IP address in the debug path, the **unmatched** flag can be enabled. However, this must be used with caution because it can produce a large amount of debug information.

GDOI Event Traces

This was added in Version 15.1(3)T. Event tracing offers light-weight, always-on tracing for significant GDOI events and errors. There is also exit-path tracing with traceback enabled for exception conditions. Event traces can provide more GETVPN event history information than traditional syslogs.

GDOI event traces are enabled by default and can be retrieved from the trace buffer with the **show monitor even-trace** command.

```
GM1#show monitor event-trace gdoi ?
all Show all the traces in current buffer
back Show trace from this far back in the past
clock Show trace from a specific clock time/date
coop GDOI COOP Event Traces
exit GDOI Exit Traces
from-boot Show trace from this many seconds after booting
infra GDOI INFRA Event Traces
latest Show latest trace events since last display
merged Show entries in all event traces sorted by time
registration GDOI Registration event Traces
rekey GDOI Rekey event Traces

GM1#show monitor event-trace gdoi rekey all
*Nov 6 15:55:16.117: GDOI_REKEY_EVENT: ACK_SENT: From 10.1.12.2 to 10.1.13.2
with seq no 1 for the group G1
*Nov 6 15:55:16.117: GDOI_REKEY_EVENT: REKEY_RCVD: From 10.1.12.2 to 10.1.13.2
with seq no 1 for the group G1
*Nov 6 16:11:01.125: GDOI_REKEY_EVENT: ACK_SENT: From 10.1.12.2 to 10.1.13.2
with seq no 1 for the group G1
*Nov 6 16:11:01.125: GDOI_REKEY_EVENT: REKEY_RCVD: From 10.1.12.2 to 10.1.13.2
with seq no 1 for the group G1
```

The exit path trace provides detailed information about exit path, that is exception and error conditions, with the traceback option enabled by default. The tracebacks can then be used in order to decode the exact code sequence that has led to the exit path condition. Use the **detail** option in order to retrieve the tracebacks from the trace buffer:

```
GM1#show monitor event-trace gdoi exit all detail
*Nov 6 15:15:25.611: NULL_VALUE_FOUND:Invalid GROUP Name
-Traceback= 0xCA51318z 0xCA1F4DBz 0xC9B2707z 0xCA1ED4Ez 0x97EB018z
0x97EA960z 0x97E8D62z 0x97F3706z 0x97F3361z 0xA02684Ez
*Nov 6 15:15:25.611: MAP_NOT_APPLIED_IN_ANY_INTERFACE:
-Traceback= 0xCA51318z 0xCA46718z 0xCA1EF79z 0x97EB018z 0x97EA960z
0x97E8D62z 0x97F3706z 0x97F3361z 0xA02684Ez 0xA01FD52z
*Nov 6 15:15:25.650: NULL_VALUE_FOUND:NULL Parameters passed idb or ipaddress
when idb ipaddress is changed
-Traceback= 0xCA51318z 0xCA22430z 0xA09A8DCz 0xA09D8F6z 0xA0F280Fz
0xBA1D1F4z 0xBA1CACCz 0xBA1C881z 0xBA1C5BBz 0xA0F494Az
```

The default trace buffer size is 512 entries, and this might not be enough if the problem is

intermittent. In order to increase this default trace entry size, the event trace configuration parameters can be changed like shown here:

```
GM1#show monitor event-trace gdoi rekey parameters
Trace has 512 entries
Stacktrace is disabled by default

GM1#
GM1#config t
Enter configuration commands, one per line. End with CNTL/Z.
GM1(config)#monitor event-trace gdoi rekey size ?
<1-1000000> Number of entries in trace
```

GETVPN Control Plane Checkpoints and Common Issues

Here are some of the common control plane issues for GETVPN. To re-iterate, the Control Plane is defined as all of the GETVPN feature components required in order to enable dataplane encryption and decryption on the GMs. At a high level, this requires successful GM registration, security policy and SA download/install, and subsequent KEK/TEK rekey.

COOP Setup and Policy Creation

In order to check and verify that the KS has successfully created the security policy and the associated KEK/TEK, enter:

```
KS1#show crypto gdoi ks policy
Key Server Policy:
For group G1 (handle: 2147483650) server 10.1.11.2 (handle: 2147483650):

For group G1 (handle: 2147483650) server 10.1.12.2 (handle: 2147483651):

# of teks : 1 Seq num : 10
KEK POLICY (transport type : Unicast)
spi : 0x18864836BA888BCD1126671EEAFEB4C7
management alg : disabled encrypt alg : 3DES
crypto iv length : 8 key size : 24
orig life(sec): 1200 remaining life(sec): 528
sig hash algorithm : enabled sig key length : 162
sig size : 128
sig key name : key1

TEK POLICY (encaps : ENCAPS_TUNNEL)
spi : 0x91E3985A
access-list : ENCPOL
transform : esp-null esp-sha-hmac
alg key size : 0 sig key size : 20
orig life(sec) : 900 remaining life(sec) : 796
tek life(sec) : 2203 elapsed time(sec) : 1407
override life (sec): 0 antireplay window size: 4

Replay Value 442843.29 secs
```

One common problem with the KS policy setup is when there are different policies configured between the primary and secondary KSs. This can result in unpredictable KS behavior and this error will be reported:

```
KS1#show crypto gdoi ks policy
```

```
Key Server Policy:
For group G1 (handle: 2147483650) server 10.1.11.2 (handle: 2147483650):

For group G1 (handle: 2147483650) server 10.1.12.2 (handle: 2147483651):

# of teks : 1 Seq num : 10
KEK POLICY (transport type : Unicast)
spi : 0x18864836BA888BCD1126671EEAFEB4C7
management alg : disabled encrypt alg : 3DES
crypto iv length : 8 key size : 24
orig life(sec): 1200 remaining life(sec): 528
sig hash algorithm : enabled sig key length : 162
sig size : 128
sig key name : key1

TEK POLICY (encaps : ENCAPS_TUNNEL)
spi : 0x91E3985A
access-list : ENCPOL
transform : esp-null esp-sha-hmac
alg key size : 0 sig key size : 20
orig life(sec) : 900 remaining life(sec) : 796
tek life(sec) : 2203 elapsed time(sec) : 1407
override life (sec): 0 antireplay window size: 4

Replay Value 442843.29 secs
```

Currently there is no automatic configuration sync between primary and secondary KSs, so these must be manually rectified.

Because COOP is a critical (and almost always mandatory) configuration for GETVPN, it is key to make sure COOP works correctly and the COOP KS roles are correct:

```
KS1#show crypto gdoi ks coop
Crypto Gdoi Group Name :G1
Group handle: 2147483650, Local Key Server handle: 2147483650

Local Address: 10.1.11.2
Local Priority: 200
 Local KS Role: Primary   , Local KS Status: Alive
Local KS version: 1.0.4
Primary Timers:
Primary Refresh Policy Time: 20
Remaining Time: 10
Antireplay Sequence Number: 40

Peer Sessions:
Session 1:
Server handle: 2147483651
Peer Address: 10.1.12.2
Peer Version: 1.0.4
Peer Priority: 100
Peer KS Role: Secondary , Peer KS Status: Alive
Antireplay Sequence Number: 0

IKE status: Established
Counters:
Ann msgs sent: 31
Ann msgs sent with reply request: 2
Ann msgs recv: 64
Ann msgs recv with reply request: 1
Packet sent drops: 7
Packet Recv drops: 0
```

```
Total bytes sent: 20887
Total bytes recv: 40244
```

In a functional COOP setup, this protocol flow should be observed:

IKE Exchange > ANN with COOP priorities exchanged > COOP Election > ANN from primary to secondary KS (policy, GM database, and keys)

When COOP does not work correctly, or if there is a COOP split, such as multiple KSs become the primary KS, these debugs must be collected for troubleshooting:

```
debug crypto isakmp
debug crypto gdoi ks coop all-levels
show crypto isakmp sa
show crypto gdoi ks coop
```

IKE Setup

Successful IKE exchange is required for GETVPN in order to secure the control channel for the subsequent policy and SA download. At the end of the successful IKE exchange, a GDOI_REKEY sa is created.

In versions earlier than Cisco IOS 15.4(1)T, the GDOI_REKEY can be shown with the **show crypto isakmp sa** command:

```
GM1#show crypto isakmp sa
IPv4 Crypto ISAKMP SA
dst src state conn-id status
10.1.13.2 10.1.11.2 GDOI_REKEY        1075 ACTIVE
10.1.11.2 10.1.13.2 GDOI_IDLE         1074 ACTIVE

IPv6 Crypto ISAKMP SA

GM1#
```

In Cisco IOS 15.4(1)T and later, this GDOI_REKEY sa is shown with the **show crypto gdoi rekey sa** command:

```
GM1#show crypto gdoi rekey sa
GETVPN REKEY SA
dst             src             conn-id         status
10.1.13.2       10.1.11.2       1114            ACTIVE
```

> **Note**: Once the initial IKE exchange completes, subsequent policies and keys will be **pushed** from the KS to the GM with the use of the GDOI_REKEY SA. So there is no rekey for the GDOI_IDLE SA when they expire; they disappear when their lifetimes expire. However, there should always be GDOI_REKEY SA on the GM in order for it to receive rekeys.

The IKE exchange for GETVPN is no different from the IKE used in traditional point-to-point IPsec tunnels, so the troubleshooting method remains the same. These debugs must be collected in order to troubleshoot IKE authentication issues:

```
debug crypto isakmp
```

debug crypto isakmp error
debug crypto isakmp detail (hidden command, if detailed isakmp exchange information is needed)
debug crypto isakmp packet (hidden command, if packet level isakmp information is needed)

Registration, Policy Download, and SA Install

Once IKE authentication succeeds, GM registers with the KS. These syslog messages are expected to be seen when this occurs correctly:

```
%GDOI-5-GM_REKEY_TRANS_2_UNI: Group G1 transitioned to Unicast Rekey.
%GDOI-5-SA_KEK_UPDATED: SA KEK was updated
%GDOI-5-SA_TEK_UPDATED: SA TEK was updated
%GDOI-5-GM_REGS_COMPL: Registration to KS 10.1.12.2 complete for group G1 using
address 10.1.13.2
%GDOI-5-GM_INSTALL_POLICIES_SUCCESS: SUCCESS: Installation of Reg/Rekey policies
from KS 10.1.12.2 for group G1 & gm identity 10.1.13.2
```

The policy and keys can be verified with this command:

```
GM1#show crypto gdoi
GROUP INFORMATION

Group Name : G1
Group Identity : 3333
Crypto Path : ipv4
Key Management Path : ipv4
Rekeys received : 1
IPSec SA Direction : Both

Group Server list : 10.1.11.2
10.1.12.2

Group member : 10.1.13.2 vrf: None
Version : 1.0.4
Registration status : Registered
Registered with : 10.1.12.2
Re-registers in       : 139 sec
Succeeded registration: 1
Attempted registration: 1
Last rekey from : 10.1.11.2
Last rekey seq num : 0
Unicast rekey received: 1
Rekey ACKs sent : 1
Rekey Rcvd(hh:mm:ss)  : 00:05:20
allowable rekey cipher: any
allowable rekey hash : any
allowable transformtag: any ESP

Rekeys cumulative
Total received : 1
After latest register : 1
Rekey Acks sents : 1

ACL Downloaded From KS 10.1.11.2:
access-list deny icmp any any
access-list deny eigrp any any
access-list deny ip any 224.0.0.0 0.255.255.255
access-list deny ip 224.0.0.0 0.255.255.255 any
access-list deny udp any port = 848 any port = 848
```

```
access-list permit ip any any

KEK POLICY:
Rekey Transport Type : Unicast
Lifetime (secs) : 878
Encrypt Algorithm : 3DES
Key Size : 192
Sig Hash Algorithm : HMAC_AUTH_SHA
Sig Key Length (bits) : 1024

TEK POLICY for the current KS-Policy ACEs Downloaded:
Serial1/0:
IPsec SA:
spi: 0x8BF147EF(2347845615)
transform: esp-3des esp-sha-hmac
sa timing:remaining key lifetime (sec): (200)
Anti-Replay(Time Based) : 4 sec interval


GM1#
GM1#
GM1#show crypto ipsec sa

interface: Serial1/0
Crypto map tag: gm1map, local addr 10.1.13.2

protected vrf: (none)
local ident (addr/mask/prot/port): (0.0.0.0/0.0.0.0/0/0)
remote ident (addr/mask/prot/port): (0.0.0.0/0.0.0.0/0/0)
current_peer 0.0.0.0 port 848
PERMIT, flags={}
#pkts encaps: 0, #pkts encrypt: 0, #pkts digest: 0
#pkts decaps: 0, #pkts decrypt: 0, #pkts verify: 0
#pkts compressed: 0, #pkts decompressed: 0
#pkts not compressed: 0, #pkts compr. failed: 0
#pkts not decompressed: 0, #pkts decompress failed: 0
#send errors 0, #recv errors 0

local crypto endpt.: 10.1.13.2, remote crypto endpt.: 0.0.0.0
path mtu 1500, ip mtu 1500, ip mtu idb Serial1/0
current outbound spi: 0x0(0)
PFS (Y/N): N, DH group: none

local crypto endpt.: 10.1.13.2, remote crypto endpt.: 0.0.0.0
path mtu 1500, ip mtu 1500, ip mtu idb Serial1/0
current outbound spi: 0x8BF147EF(2347845615)
PFS (Y/N): N, DH group: none

inbound esp sas:
spi: 0x8BF147EF(2347845615)
transform: esp-3des esp-sha-hmac ,
in use settings ={Tunnel, }
conn id: 1, flow_id: SW:1, sibling_flags 80000040, crypto map: gm1map
sa timing: remaining key lifetime (sec): (192)
Kilobyte Volume Rekey has been disabled
IV size: 8 bytes
replay detection support: Y replay window size: 4
Status: ACTIVE(ACTIVE)

inbound ah sas:

inbound pcp sas:

outbound esp sas:
```

```
spi: 0x8BF147EF(2347845615)
transform: esp-3des esp-sha-hmac ,
in use settings ={Tunnel, }
conn id: 2, flow_id: SW:2, sibling_flags 80000040, crypto map: gm1map
sa timing: remaining key lifetime (sec): (192)
Kilobyte Volume Rekey has been disabled
IV size: 8 bytes
replay detection support: Y replay window size: 4
Status: ACTIVE(ACTIVE)

outbound ah sas:

outbound pcp sas:
GM1#
```

Note: With GETVPN, inbound and outbound SAs use the same SPI.

With GETVPN registration and policy install type of problems, these debugs are needed in order to troubleshoot:

```
debug crypto isakmp (KS and GM)
debug crypto gdoi ks registration all-levels (KS)
debug crypto gdoi gm registration all-level (GM)
debug crypto engine (GM only)
show crypto eli detail (multiple iterations on GM)
```

Note: Additional debugs may be required depending on the outcome of these outputs.

Since GETVPN registration typically occurs immediately after the GM reload, this EEM script might be helpful in order to collect these debugs:

```
debug crypto isakmp (KS and GM)
debug crypto gdoi ks registration all-levels (KS)
debug crypto gdoi gm registration all-level (GM)
debug crypto engine (GM only)
show crypto eli detail (multiple iterations on GM)
```

Rekey

Once the GMs are registered to the KS and the GETVPN network is properly set up, the primary KS is responsible for sending rekey messages to all the GMs registered to it. The rekey messages are used in order to synchronize all the policies, keys, and pseudotimes on the GMs. The rekey messages can be sent through a unicast or a multicast method.

This syslog message is seen on the KS when the rekey message is sent:

```
debug crypto isakmp (KS and GM)
debug crypto gdoi ks registration all-levels (KS)
debug crypto gdoi gm registration all-level (GM)
debug crypto engine (GM only)
show crypto eli detail (multiple iterations on GM)
```

On the GMs, this is the syslog that is seen when it receives the rekey:

```
debug crypto isakmp (KS and GM)
```

```
debug crypto gdoi ks registration all-levels (KS)
debug crypto gdoi gm registration all-level (GM)
debug crypto engine (GM only)
show crypto eli detail (multiple iterations on GM)
```

RSA Key Pair Requirement for Rekey on KS

Rekey functionality requires the presence of RSA keys on the KS. The KS provides the public key of the RSA key pair to the GM through this secure channel during registration. The KS then signs the GDOI messages sent to the GM with the private RSA key in the GDOI SIG payload. The GM receives the GDOI messages and uses the public RSA key in order to verify the message. The messages between the KS and the GM are encrypted with the KEK, which is also distributed to the GM during registration. Once the registration is complete, subsequent rekeys are encrypted with the KEK and signed with the private RSA key.

If the RSA key is no present on the KS during GM registration, this message appears on the syslog:

```
debug crypto isakmp (KS and GM)
debug crypto gdoi ks registration all-levels (KS)
debug crypto gdoi gm registration all-level (GM)
debug crypto engine (GM only)
show crypto eli detail (multiple iterations on GM)
```

When the keys are not present on the KS, the GM registers for the first time, but the next rekey fails from the KS. Eventually the existing keys on the GM expire, and it reregisters again.

```
debug crypto isakmp (KS and GM)
debug crypto gdoi ks registration all-levels (KS)
debug crypto gdoi gm registration all-level (GM)
debug crypto engine (GM only)
show crypto eli detail (multiple iterations on GM)
```

Since the RSA key pair is used in order to sign the rekey messages, they **MUST** be the same between the primary and all secondary KSs. This ensures that during a primary KS failure, the rekeys sent by a secondary KS (the new primary KS) can still be properly validated by the GMs. When it generates the RSA key pair on the primary KS, the key pair must be created with the **exportable** option so that they can be exported to all the secondary KSs in order to meet this requirement.

Rekey Troubleshooting

KEK/TEK rekey failure is one of the most common GETVPN problems encountered in customer deployments. Troubleshooting rekey issues should follow the rekey steps as outlined here:

1. Did the rekeys get sent by the KS?

   This can be checked by an observion of the %GDOI-5-KS_SEND_UNICAST_REKEY syslog message or more accurately with this command:

```
KS1#show crypto gdoi ks rekey
Group G1 (Unicast)
Number of Rekeys sent               : 341
Number of Rekeys retransmitted      : 0
```

```
KEK rekey lifetime (sec) : 1200
Remaining lifetime (sec) : 894
Retransmit period : 10
Number of retransmissions : 5
IPSec SA 1 lifetime (sec) : 900
Remaining lifetime (sec) : 405
```

The number of rekeys retransmitted is indicative of rekey acknowledgment packets not received by the KS and therefore possible rekey issues. Keep in mind that the GDOI rekey uses UDP as an unreliable transport mechanism, so some rekey drops might be expected depending on the reliability of the underlying transport network, but a trend of increasing rekey retransmissions should always be investigated.

More detailed per-GM rekey statistics can also be obtained. This is typically the first place to look for potential rekey issues.

```
KS1#show crypto gdoi ks members

Group Member Information :

Number of rekeys sent for group G1 : 346

Group Member ID : 10.1.14.2 GM Version: 1.0.4
Group ID : 3333
Group Name : G1
Key Server ID : 10.1.11.2
 Rekeys sent       : 346
Rekeys retries : 0
Rekey Acks Rcvd : 346
Rekey Acks missed : 0

Sent seq num : 2 1 2 1
Rcvd seq num : 2 1 2 1

Group Member ID : 10.1.13.2 GM Version: 1.0.4
Group ID : 3333
Group Name : G1
Key Server ID : 10.1.12.2
 Rekeys sent       : 340
Rekeys retries : 0
Rekey Acks Rcvd : 340
Rekey Acks missed : 0

Sent seq num : 2 1 2 1
Rcvd seq num : 2 1 2 1
```

2. Did the rekey packets get delivered in the underlying infrastructure network?

   Standard IP troubleshooting along the rekey forwarding path should be followed in order to ensure the rekey packets are not dropped in the transit network between KS and GM. Some common troubleshooting tools used here are input/output Access Control Lists (ACLs), Netflow, and packet capture in the transit network.

3. Did the rekey packets reach the GDOI process for rekey processing?

   Check the GM rekey statistics:

```
GM1#show crypto gdoi gm rekey
Group G1 (Unicast)
Number of Rekeys received (cumulative) : 340
Number of Rekeys received after registration : 340
Number of Rekey Acks sent : 340
```

4. Did the rekey acknowledgement packet return to the KS?

   Follow Steps 1 through 3 in order to trace the rekey acknowledgement packet from the GM back to the KS.

Muticast Rekey

Multicast rekey is different from unicast rekey in these aspects:

- Since multicast is used in order to transport these rekey packets from the KS to the GMs, the KS does not need to replicate the rekey packets itself. The KS only sends one copy of the rekey packet, and they are replicated in the multicast-enabled network.
- There is no acknowledgement mechanism for multicast rekey, so if a GM were not to receive the rekey packet, the KS would have no knowledge of it, and therefore will never remove a GM from its GM database. And because there is no acknowledgement, the KS will always retransmit the rekey packets based on its rekey retransmission configuration.

The most commonly seen multicast rekey problem is when the rekey is not received on the GM. There could be a number of possible causes for this, such as:

- Packet delivery issue within the multicast routing infrastructure
- End-to-end multicast routing is not enabled within the network

The first step to troubleshoot an issue with multicast rekey is to see if rekey works when switched from the multicast to the unicast method.

Once you identify that the issue is specific to multicast rekey, verify that KS sends the rekey to the multicast address specified.

```
GM1#show crypto gdoi gm rekey
Group G1 (Unicast)
Number of Rekeys received (cumulative) : 340
Number of Rekeys received after registration : 340
Number of Rekey Acks sent : 340
```

Test multicast connectivity between the KS and GM with an Internet Control Message Protocol (ICMP) request to the multicast address. All the GMs that are part of the multicast group should reply to the ping. Ensure that ICMP is excluded from the KS encryption policy for this test.

```
KS1#ping 226.1.1.1

Type escape sequence to abort.
Sending 1, 100-byte ICMP Echos to 226.1.1.1, timeout is 2 seconds:

Reply to request 0 from 10.1.21.2, 44 ms
```

If the multicast ping test fails, then multicast troubleshooting must be performed, which is outside of the scope of this document.

Control Plane Relay Check

Symptom

When customers upgrade their GM to a new Cisco IOS version, they might experience KEK rekey failures with this message observed in the syslog:

```
KS1#ping 226.1.1.1

Type escape sequence to abort.
Sending 1, 100-byte ICMP Echos to 226.1.1.1, timeout is 2 seconds:

Reply to request 0 from 10.1.21.2, 44 ms
```

This behavior is caused by an interoperability issue introduced with the anti-replay check that is added for control plane messages. Specifically, a KS that runs the older code will reset the KEK rekey sequence number to 1, and this will be dropped by the GM that runs the new code when it interprets that as a replayed rekey packet. For more details, see Cisco bug ID CSCta05809 (GETVPN: GETVPN control-plane sensible to replay), and GETVPN Configuration Restrictions.

Background

With GETVPN, the Control Plane messages can carry time-sensitive information in order to provide the time-based anti-replay check service. Therefore, these messages require anti-replay protection themselves in order to ensure time accruracy. These messages are:

- **Rekey Messages** from KS to GM
- **COOP Announcement Messages** between KSs

As part of this anti-replay protection implementation, sequence number checks were added in order to protect replayed messages, as well as a pseudotime check when TBAR is enabled.

Solution

In order to resolve this issue, both the GM and KS must be upgraded to Cisco IOS versions after the Control Plane replay check feature. With the new Cisco IOS code, KS does not reset the sequence number back to 1 for a KEK rekey, but instead it continues to use the current sequence number and only resets the sequence number for TEK rekeys.

These Cisco IOS versions have the Replay Check features:

- 12.4(15)T10
- 12.4(22)T3
- 12.4(24)T2
- 15.0(1)M and later

Other Replay Related Issues

- COOP failure due to ANN messages failing replay check (Cisco bug ID CSCtc52655)

Debug Control Plane Replay Failures

For other Control Plane Replay failures, collect this information and make sure the times are synched between the KS and GM.

- Syslog from both GM and KS
- ISAKMP debugs
- GDOI debugs (rekey and replay) from both KS and GM

Control Plane Packet Fragmentation Issues

With GETVPN, Control Plane Packet fragmentation is a common issue, and it can manifest itself in one of these two scenarios when the Control Plane packets are large enough that they will require IP fragmentation:

- GETVPN COOP Announcement packets
- GETVPN rekey packets

COOP Announcement Packets

The COOP Announcement packets carry the GM database information, and thus can grow big in a large GETVPN deployment. From past experience, a GETVPN network that consists of 1500+ GMs will produce Announcement packets larger than 18024 bytes, which is the Cisco IOS default Huge buffer size. When this happens, the KS fails to allocate a buffer large enough to transmit the ANN packets with this error:

```
KS1#ping 226.1.1.1

Type escape sequence to abort.
Sending 1, 100-byte ICMP Echos to 226.1.1.1, timeout is 2 seconds:

Reply to request 0 from 10.1.21.2, 44 ms
```

In order to rectify this condition, this buffer tuning is recommended:

```
KS1#ping 226.1.1.1

Type escape sequence to abort.
Sending 1, 100-byte ICMP Echos to 226.1.1.1, timeout is 2 seconds:

Reply to request 0 from 10.1.21.2, 44 ms
```

Rekey Packets

GETVPN rekey packets can also exceed the typical 1500 IP Maximum Transition Unit (MTU) size when the encryption policy is large, such as a policy that consists of 8+ lines of Access Control Entries (ACEs) in the encryption ACL.

Fragmentation Problem and Identification

In both of the previous scenarios, GETVPN must be able to properly transmit and receive the fragmented UDP packets in order for COOP or GDOI rekey to work properly. IP fragmentation can be a problem in some network environments. For example, a network that consists of Equal Cost Multi Path (ECMP) forwarding plane, and some devices in the forwarding plane require virtual reassembly of the fragmented IP packets, such as Virtual Fragmentation Reassembly (VFR).

In order to identify the problem, check the reassembly errors on the device where it is suspected that the fragmented UDP 848 packets are not properly received:

```
KS1#show ip traffic | section Frags
Frags: 10 reassembled, 3 timeouts, 0 couldn't reassemble
0 fragmented, 0 fragments, 0 couldn't fragment
```

If the reassembly timeouts continue to increment, use the **debug ip error** command in order to confirm if the drop is part of the rekey/COOP packet flow. Once confirmed, normal IP forwarding troubleshooting should be performed in order to isolate the exact device in the forwarding plane that might have dropped the packets. Some commonly used tools include:

- Packet capture
- Traffic forwarding statistics
- Security feature statistics (Firewall, IPS)
- VFR statistics

GDOI Interoperability Issues

Various interoperability issues have been found with GETVPN over the years, and it is critical to notice the Cisco IOS release versions between KS and GM and amongst the KSs for interoperability issues.

Other well known GETVPN interoperability issues are:

- Control Plane Relay Check
- GETVPN KEK Rekey Behavior Change
- Cisco bug ID CSCub42920 (GETVPN: KS fails to validate hash in rekey ACK from previous GM versions)
- Cisco bug ID CSCuw48400 (GetVPN GM unable to register or rekey fails - sig-hash > default SHA-1)
- Cisco bug ID CSCvg19281 (Multiple GETVPN GM crashes after migration to new KS pair ; if a GM version is earlier than 3.16, and KS is upgraded from an earlier code to 3.16 or later, this issue can happen)

GETVPN IOS Upgrade Procedure

This Cisco IOS upgrade procedure should be followed when a Cisco IOS code upgrade needs to be performed in a GETVPN environment:

1. Upgrade a secondary KS first and wait until COOP KS election is completed.
2. Repeat Step1 for all secondary KSs.
3. Upgrade the primary KS.
4. Upgrade GMs.

Troubleshoot GETVPN Data Plane Issues

Compared to Control Plane problems, GETVPN data plane issues are problems where the GM has the policy and keys to perform dataplane encryption and decryption, but for some reason the

end-to-end traffic flow does not work. Most of the dataplane issues for GETVPN relate to generic IPsec forwarding, and are not GETVPN specific. So most of the troubleshooting approach described here applies to generic IPsec dataplane issues as well.

With encryption problems (both Group-based or pair-wise tunnels), it is important to troubleshoot the problem and isolate the problem to a particular part of the datapath. Specifically, the troubleshooting approach described here is intended to help you answer these questions:

- Which device is the culprit - encrypting router or decrypting router?
- In which direction is the problem happening - ingress or egress?

GETVPN Data Plane Troubleshooting Tools

IPsec dataplane troubleshooting is very different from that for the Control Plane. With the dataplane, there are usually no debugs that you can run, or at least run safely in a production environment. So the troubleshooting relies heavily on different counters and traffic statistics that can help trace the packet along a forwarding path. The idea is to be able to develop a set of checkpoints in order to help isolate where packets might be dropped as shown here:

Here are some data plane debugging tools:

- Access Lists
- IP Precedence Accounting
- Netflow
- Interface Counters
- Crypto Counters
- IP Cisco Express Forwarding (CEF) Global and Per-feature Drop Counters
- Embedded Packet Capture (EPC)
- Data Plane Debugs (IP packet and CEF debugs)

The checkpoints in the datapath in the previous image can be validated with these tools:

Encrypting GM

- Ingress LAN interface
  - Input ACL
  - Ingress netflow
  - Embedded Packet Capture
  - Input precedence accounting

- Crypto engine
  - **show crypto ipsec sa**
  - **show crypto ipsec sa detail**

show crypto engine accelerator statistics

- Egress WAN interface
  Egress netflow
  Embedded Packet Capture
  Output precedence accounting

Decrypting GM

- Ingress WAN interface
  Input ACL
  Ingress netflow
  Embedded Packet Capture
  Input precedence accounting

- Crypto Engine
  show crypto ipsec sa
  show crypto ipsec sa detail
  show crypto engine accelerator statistics

- Egress LAN interface
  Egress netflow
  Embedded packet capture

The return path follows the same traffic flow. The next sections have some examples of these dataplane tools in use.

Encryption/Decryption Counters

The encryption/decryption counters on a router are based on an IPsec flow. Unfortunately this does not work well with GETVPN since GETVPN typically deploys a "permit ip any any" encryption policy that encrypts everything. So if the problem only happens for some of the flows and not all, these counters can be somewhat difficult to use in order to correctly assess if the packets are encrypted or decrypted when there is enough significant background traffic that works.

```
GM1#show crypto ipsec sa | in encrypt|decrypt
#pkts encaps: 100, #pkts encrypt: 100, #pkts digest: 100
#pkts decaps: 100, #pkts decrypt: 100, #pkts verify: 100
```

Netflow

Netflow can be used in order to monitor both the ingress and egress traffic on both GMs. Note with the GETVPN **permit ip any any** policy, the enrypted traffic will be aggregate and does not provide the per-flow information. Per-flow information will then need to be collected with the DSCP/precedence marking described later.

In this example, the netflow for a 100 count ping from a host behind GM1 to a host behind GM2 is shown at the various checkpoints.

Encrypting GM

Netflow configuration:

```
interface Ethernet0/0
description LAN
ip address 192.168.13.1 255.255.255.0
ip flow ingress
ip pim sparse-dense-mode
!
interface Serial1/0
description WAN interface
ip address 10.1.13.2 255.255.255.252
ip flow egress
ip pim sparse-dense-mode
crypto map gm1map
```

Netflow output:

```
GM1#show ip cache flow | be SrcIf
SrcIf SrcIPaddress DstIf DstIPaddress Pr SrcP DstP Pkts
Et0/0 192.168.13.2 Se1/0* 192.168.14.2 32 8DE1 6523   100
Et0/0 192.168.13.2 Se1/0 192.168.14.2 01 0000 0800   100
GM1#
```

> **Note**: In the previous output, * denotes egress traffic. The first line shows egress encrypted traffic (with protocol 0x32 = ESP) out of the WAN inteface, and the second line ingress ICMP traffic hitting the LAN interface.

Decrypting GM

Configuration:

```
interface Ethernet0/0
description LAN interface
ip address 192.168.14.1 255.255.255.0
ip flow egress
ip pim sparse-dense-mode
!
interface Serial1/0
description WAN interface
ip address 10.1.14.2 255.255.255.252
ip flow ingress
ip pim sparse-dense-mode
crypto map gm1map
```

Netflow output:

```
GM2#show ip cache flow | be SrcIf
SrcIf SrcIPaddress DstIf DstIPaddress Pr SrcP DstP Pkts
Se1/0 192.168.13.2 Et0/0 192.168.14.2 32 8DE1 6523   100
Se1/0 192.168.13.2 Et0/0* 192.168.14.2 01 0000 0800   100
GM2#
```

DSCP/IP Precedence Marking

The challenge with troubleshooting an encryption problem is that once the packet is encrypted you lose visibility into the payload, which is what encryption is supposed to do, and that makes it difficult to trace the packet for a particular IP flow. There are two ways to address this limitation when it comes to troubleshooting an IPsec problem:

- Use ESP-NULL as the IPsec transform. IPsec still performs ESP encapsulation but no encryption is applied to the payload, so they are visible in a packet capture.
- Mark an IP flow with a unique Differentiated Services Code Point (DSCP)/precedence marking based on their L3/L4 characteristics.

ESP-NULL require changes on both tunnel end points and often is not allowed based on the customer security policy. Therefore, Cisco typically recommends the use of DSCP/precedence marking instead.

DSCP/Precedence Reference Chart

ToS (hex)	ToS(Decimal)	IP Precedence	DSCP	Binary
0xE0	224	7 Network Control	56 CS7	**111**00000
0xC0	192	6 Internetwork Control	48 CS6	**110**00000
0xB8	184	5 Critical	46 EF	**101**11000
0xA0	160		40 CS5	**101**00000
0x88	136	4 Flash Override	34 AF41	**100**01000
0x80	128		32 CS4	**100**00000
0x68	104	3 Flash	26 AF31	**011**01000
0x60	96		24 CS3	**011**00000
0x48	72	2 Immediate	18 AF21	**010**01000
0x40	64		16 CS2	**010**00000
0x20	32	1 Priority	8 CS1	**001**00000
0x00	0	0 Routine	0 Dflt	**000**00000

Mark Packets with DSCP/Precedence

These methods are typically used in order to mark packets with the specific DSCP/Precedence markings.

PBR

```
interface Ethernet1/0
ip policy route-map mark
!
access-list 150 permit ip host 172.16.1.2 host 172.16.254.2
!
route-map mark permit 10
match ip address 150
set ip precedence flash-override
```

MQC

```
class-map match-all my_flow
match access-group 150
!
policy-map marking
class my_flow
set ip precedence 4
!
interface Ethernet1/0
service-policy input marking
```

Router Ping

```
GM1-host#ping ip
Target IP address: 192.168.14.2
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Extended commands [n]: y
Source address or interface:
Type of service [0]: 136
...
<snip>
```

> **Note**: It is always a good idea to monitor the normal traffic flow and DSCP/precedence profile before you apply marking so that the marked traffic flow is unique.

Monitor Marked Packets

IP Precedence Accounting

```
interface Ethernet0/0
ip address 192.168.1.2 255.255.255.0
ip accounting precedence input
```

```
middle_router#show interface precedence
Ethernet0/0
Input
Precedence 4: 100 packets, 17400 bytes
```

Interface ACL

```
middle_router#show access-list 144
Extended IP access list 144
10 permit ip any any precedence routine
20 permit ip any any precedence priority
30 permit ip any any precedence immediate
40 permit ip any any precedence flash
50 permit ip any any precedence flash-override (100 matches)
60 permit ip any any precedence critical
70 permit ip any any precedence internet (1 match)
80 permit ip any any precedence network
```

Embedded Packet Capture

Embedded Packet Capture (EPC) is a useful tool to capture packets at the interface level in order to identify if a packet has reached a specific device. Remember that EPC works well for clear text traffic, but it can be a challenge when the captured packets are encrypted. Therefore techniques like DSCP/precedence marking discussed previously or other IP characters, such as the length of the IP packet, have to be used together with EPC in order to make the troubleshooting more effective.

Cisco IOS-XE Packet Trace

This is a useful feature to trace the feature forwarding path on all platforms that run Cisco IOS-XE, such as CSR1000v, ASR1000, and ISR4451-X.

GETVPN Data Plane Common Issues

Troubleshooting the IPsec dataplane for GETVPN is mostly no different from troubleshooting traditional point-to-point IPsec dataplane issues, with two exceptions due to these unique dataplane properties of GETVPN.

Time Based Anti-Replay Failure

In a GETVPN network, TBAR failures can often be difficult to troubleshoot since there are no longer pair-wise tunnels. In order to troubleshoot GETVPN TBAR failures, complete these steps:

1. Identify which packet is dropped due to TBAR failure and subsequently identify the encrypting GM.

   Prior to Version 15.3(2)T, the TBAR failure syslog did not print the source address of the failed packet, so this makes it very difficult to identify which packet failed. This has been significantly improved in Version 15.3(2)T and later, where Cisco IOS prints this:

```
middle_router#show access-list 144
Extended IP access list 144
10 permit ip any any precedence routine
20 permit ip any any precedence priority
30 permit ip any any precedence immediate
40 permit ip any any precedence flash
50 permit ip any any precedence flash-override (100 matches)
60 permit ip any any precedence critical
70 permit ip any any precedence internet (1 match)
80 permit ip any any precedence network
```

   A TBAR history was also implemented in this version:

```
GM2#show crypto gdoi gm replay
Anti-replay Information For Group G1:
Timebased Replay:
Replay Value : 621388.66 secs
Input Packets : 0 Output Packets : 0
Input Error Packets : 2 Output Error Packets : 0
Time Sync Error : 0 Max time delta : 0.00 secs


TBAR Error History (sampled at 10pak/min):
19:29:32.081 EST Wed Nov 13 2013: src=192.168.13.2; my_pst=620051.84 secs;
peer_pst=619767.09 secs; win=4
```

   Note: The enhancements mentioned previously have since been implemented in Cisco IOS-XE by Cisco bug ID CSCun49335 and in Cisco IOS by Cisco bug ID CSCub91811.
   For Cisco IOS versions that did not have this feature, **debug crypto gdoi gm replay detail** can also provide this information, although this debug prints the TBAR information for all traffic (not only packets dropped due to TBAR failure), so it might not be feasible to run in a production environment.

```
GM2#show crypto gdoi gm replay
Anti-replay Information For Group G1:
Timebased Replay:
Replay Value : 621388.66 secs
Input Packets : 0 Output Packets : 0
Input Error Packets : 2 Output Error Packets : 0
Time Sync Error : 0 Max time delta : 0.00 secs


TBAR Error History (sampled at 10pak/min):
19:29:32.081 EST Wed Nov 13 2013: src=192.168.13.2; my_pst=620051.84 secs;
peer_pst=619767.09 secs; win=4
```

2. Once the source of the packet is identified, you should be able to find the encrypting GM. Then, the pseudotimestamp on both the encrypting and decrypting GMs should be monitored for any potential pseudotime drift. The best way to do this would be to synchronize both GMs and the KS to NTP and periodically collect the pseudotime information with a reference system clock on all of them in order to determine if the problem is caused by clock skew on the GMs.

GM1

```
GM1#show crypto gdoi gm replay
Load for five secs: 0%/0%; one minute: 0%; five minutes: 0%
Time source is hardware calendar, *21:06:26.469 EST Wed Nov 13 2013

Anti-replay Information For Group G1:
Timebased Replay:
Replay Value             : 625866.26 secs
Input Packets : 0 Output Packets : 0
Input Error Packets : 0 Output Error Packets : 0
Time Sync Error : 0 Max time delta : 0.00 secs
```

GM2

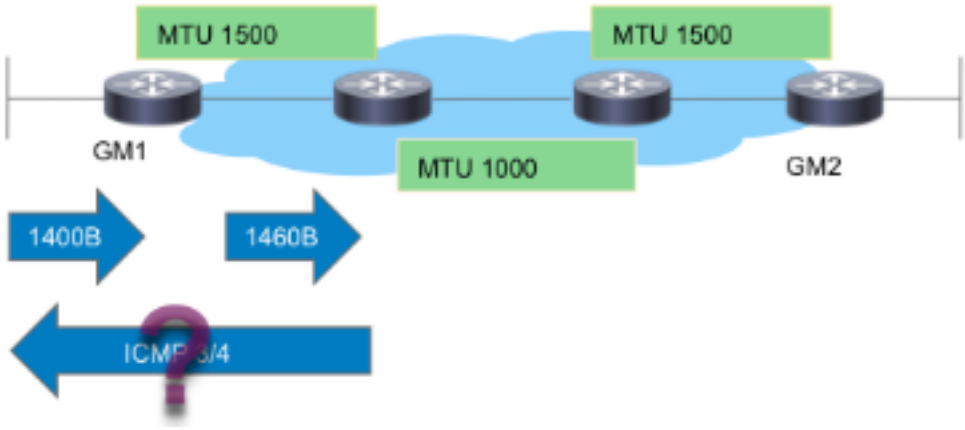
```
GM2#show crypto gdoi gm replay
Load for five secs: 0%/0%; one minute: 0%; five minutes: 0%
Time source is hardware calendar, *21:06:26.743 EST Wed Nov 13 2013

Anti-replay Information For Group G1:
Timebased Replay:
Replay Value             : 625866.51 secs
Input Packets : 4 Output Packets : 4
Input Error Packets : 2 Output Error Packets : 0
Time Sync Error : 0 Max time delta : 0.00 secs
```

In the previous example, if the pseudotime (as indicated by Replay Value) is significantly different between the GMs when the outputs are captured with the same reference time, then the problem can be attributed to clock skew.

> **Note**: On the Cisco Aggregated Services Router 1000 Series platform, due to the platform architecture, the datapath on the Quantum Flow Processor (QFP) actually refers to the wall clock for counting pseudotime ticks. This has created problems with TBAR when the wall clock time changes due to NTP sync. This problem is documented with Cisco bug ID CSCum37911.

PMTUD and GETVPN Header Preservation

With GETVPN, Path MTU Discovery (PMTUD) does not work between the encrypting and decrypting GMs, and large packets with the Don't Fragment (DF) bit set can get blackholed. The reason that this does not work is due to GETVPN Header Preservation where the data source/destination addresses are preserved in the ESP encapsulating header. This is depicted in this image:



As the image shows, PMTUD breaks down with GETVPN with this flow:

1. Large data packet arrives on the encrypting GM1.
2. The post-encryption ESP packet is forwarded out of GM1 and delivered towards the destination.
3. If there is a transit link with IP MTU of 1400 bytes, the ESP packet will be dropped, and an ICMP 3/4 packet too big message will be sent towards the packet source, which is the source of the data packet.
4. The ICMP3/4 packet is either dropped due to ICMP not excluded from the GETVPN encryption policy, or dropped by the end host since it does not know anything about the ESP packet (unauthenticated payload).

In summary, PMTUD does not work with GETVPN today. In order to work around this issue, Cisco recommends these steps:

1. Implement "ip tcp adjust-mss" in order to reduce the TCP packet segment size tin order o accommodate encryption overhead and minimum path MTU in the transit network.
2. Clear the DF bit in the data packet as they arrive on the encrypting GM in order to avoid PMTUD.

Generic IPsec Dataplane Issues

Most of the IPsec dataplane troubleshooting is like troubleshooting traditional point-to-point IPsec tunnels. One of the common issues is %CRYPTO-4-RECVD_PKT_MAC_ERR. See Syslog "%CRYPTO-4-RECVD_PKT_MAC_ERR:" Error Message with Ping Loss Over IPsec Tunnel Troubleshooting for more troubleshooting details.

Known Issues

This message can be generated when an IPsec packet is received that does not match an SPI in the SADB. See Cisco bug ID CSCtd47420 - GETVPN - CRYPTO-4-RECVD_PKT_NOT_IPSEC reported for pkt not matching flow. An example is:

```
GM2#show crypto gdoi gm replay
Load for five secs: 0%/0%; one minute: 0%; five minutes: 0%
Time source is hardware calendar, *21:06:26.743 EST Wed Nov 13 2013

Anti-replay Information For Group G1:
Timebased Replay:
Replay Value             : 625866.51 secs
Input Packets : 4 Output Packets : 4
Input Error Packets : 2 Output Error Packets : 0
Time Sync Error : 0 Max time delta : 0.00 secs
```

This message should be %CRYPTO-4-RECVD_PKT_INV_SPI, which is what gets reported for traditional IPsec as well as on some hardware platforms such as ASR. This cosmetic issue was fixed by Cisco bug ID CSCup80547: Error in reporting CRYPTO-4-RECVD_PKT_NOT_IPSEC for ESP pak.

> **Note**: These messages can sometimes appear due to another GETVPN bug CSCup34371: GETVPN GM stops decrytping traffic after TEK rekey.

In this case, the GM cannot decrypt GETVPN traffic, although it has a valid IPsec SA in the SADB (the SA being rekeyed). The problem disappears as soon as the SA expires and is removed from the SADB. This issue causes significant outage, because TEK rekey is performed in advance. For example, the outage can be 22 minutes in the case of a TEK lifetime of 7200 seconds. See the bug description for the exact condition that should be met in order to encounter this bug.

Troubleshoot GETVPN on Platforms that Run Cisco IOS-XE

Troubleshooting Commands

Platforms that run Cisco IOS-XE have platform-specific implementations, and often require platform-specific debugging for GETVPN issues. Here are a list of commands typically used in order to troubleshoot GETVPN on these platforms:

show crypto eli all

show platform software ipsec policy statistics

show platform software ipsec fp active inventory

show platform hardware qfp active feature ipsec spd all

show platform hardware qfp active statistics drop clear

show platform hardware qfp active feature ipsec data drop clear

show crypto ipsec sa

show crypto gdoi

show crypto ipsec internal

debug crypto ipsec

debug crypto ipsec error

debug crypto ipsec states

debug crypto ipsec message

debug crypto ipsec hw-req

debug crypto gdoi gm infra detail

debug crypto gdoi gm rekey detail

ASR1000 Common Issues

IPsec Policy Install Failure (Continuous Re-registration)

An ASR1000 GM might continue to register to the Key Server if the crypto engine does not support the IPsec policy or algorithm received. For example, on Nitrox based ASR platforms (such as ASR1002), Suite-B or SHA2 policies are not supported and this can cause the continuous re-registration symptoms.

Common Migration/Upgrade Issues

ASR1000 TBAR Limitation

On the ASR1000 platform, the Cisco bug ID CSCum37911 fix introduced a limitation on this platform where TBAR time of less than 20 seconds is not supported. See Restrictions for GETVPN on IOS-XE.

This enhancement bug has been opened to lift this restriction, Cisco bug ID CSCuq25476 - ASR1k needs to support a GETVPN TBAR window size of less than 20 seconds.

Update: This restriction has since been lifted with the fix for Cisco bug ID CSCur57558 , and it is no longer a limitation in XE3.10.5, XE3.13.2 and later code.

Also note, for a GM that runs on Cisco IOS-XE platforms (ASR1k or ISR4k), it is highly recommended that the device runs a version with the fix for this issue if TBAR is enabled; Cisco bug ID CSCut91647 - GETVPN on IOS-XE: GM incorrectly drops packets due to TBAR failure.

ISR4x00 Classification Issue

A regression was found on the ISR4x00 platform where the deny policies are ignored. For details, see Cisco bug ID CSCut14355 - GETVPN - ISR4300 GM ignores deny policy.

Related Information

- **Group Encrypted Transport VPN (GET VPN) - Cisco Systems**
- **Technical Support & Documentation - Cisco Systems**