Document ID: 117721

Updated: Dec 10, 2015

Contributed by Rama Darbha, Namit Agarwal, and Olivier Pelerin, Cisco TAC Engineers.

Related Products

- Cisco IOS Firewall

Contents

Introduction

This document describes how to best troubleshoot the Zone Based Firewall (ZBFW) feature on the Aggregation Services Router (ASR) ® programs the hardware ASICs (quantum flow processor (QFP) in order to perform feature forwarding functionalities. This allows for higher throughput and better performance. The drawback to this is that it presents a greater challenge to troubleshoot. Traditional Cisco IOS commands used to poll current sessions and drop counters via Zone-Based Firewall (ZBFW) are no longer valid as the drops are no longer in software.

Links and Documentation

Command References

- Cisco ASR 1000 Series Aggregation Services Routers Command References
- Cisco IOS XE 3S Command References

Datapath Troubleshoot Steps

In order to troubleshoot the datapath, you must identify whether traffic is properly passed through the ASR and Cisco IOS-XE code. Specific to firewall features, the datapath troubleshooting follows these steps:

1. **Verify Configuration** - Gather the configuration and examine the output in order to verify the connection.

2. **Verify Connection State** - If traffic passes properly, the Cisco IOS-XE opens up a connection on the ZBFW feature. This connection tracks the traffic and state information between a client and server.

3. **Verify Drop Counters** - When traffic does not pass properly, Cisco IOS-XE logs a drop counter for any dropped packets. Check this output in order to isolate the cause of the traffic failure.

4. **Logging** - Gather syslogs in order to provide more granular information on connection builds and packet drops.

5. **Packet Trace Dropped Packets** - Use packet tracing in order to catch dropped packets.

6. **Debugs** - Gather debugs is the most verbose option. Debugs can be obtained conditionally in order to confirm the exact forwarding path for the packets.

Verify Configuration

The output of **show tech support firewall** is summarized here:

```
-------- show clock --------
-------- show version -------
-------- show running-config --------
-------- show parameter-map type inspect --------
-------- show policy-map type inspect -------
-------- show class-map type inspect --------
-------- show zone security --------
-------- show zone-pair security --------
-------- show policy-firewall stats global --------
-------- show policy-firewall stats zone --------
-------- show platform hardware qfp active feature firewall datapath <submode> --------
-------- show platform software firewall RP <submode> --------
```

Verify Connection State

Connection information can be obtained so that all connections on ZBFW are listed. Enter this command:

```
ASR#show policy-firewall sessions platform
--show platform hardware qfp active feature firewall datapath scb any any any any any all any --
[s=session  i=imprecise channel  c=control channel  d=data channel]
 14.38.112.250 41392 14.36.1.206 23 proto 6 (0:0)    [sc]
```

It shows a TCP telnet connection from 14.38.112.250 to 14.36.1.206.

> **Note**: Be aware that if you run this command, it will take a long time if there are lots of connections on the device. Cisco recommends that you run this command with specific filters as outlined here.

The connection table can be filtered down to a specific source or destination address. Use filters after **platform** submode. The options to filter are:

```
radar-ZBFW1#show policy-firewall sessions platform ?
 all                     detailed information
 destination-port        Destination Port Number
 detail                  detail on or off
 icmp                    Protocol Type ICMP
 imprecise               imprecise information
 session                 session information
 source-port             Source Port
 source-vrf              Source Vrf ID
 standby                 standby information
 tcp                     Protocol Type TCP
 udp                     Protocol Type UDP
 v4-destination-address  IPv4 Desination Address
 v4-source-address       IPv4 Source Address
 v6-destination-address  IPv6 Desination Address
 v6-source-address       IPv6 Source Address
 |                       Output modifiers
 <cr>
```

This connection table is filtered so only connections sourced from 14.38.112.250 are displayed:

```
ASR#show policy-firewall sessions platform v4-source-address 14.38.112.250
--show platform hardware qfp active feature firewall datapath scb 14.38.112.250
any any any any all any --
[s=session  i=imprecise channel  c=control channel  d=data channel]
 14.38.112.250 41392 14.36.1.206 23 proto 6 (0:0)    [sc]
```

Once the connection table is filtered, the detailed connection information can be obtained for a more comprehensive anlaysis. In order to display this output, use the **detail** keyword.

```
ASR#show policy-firewall sessions platform v4-source-address 14.38.112.250 detail
--show platform hardware qfp active feature firewall datapath scb 14.38.112.250
any any any any all any detail--
[s=session  i=imprecise channel  c=control channel  d=data channel]
 14.38.112.250 41426 14.36.1.206 23 proto 6 (0:0)       [sc]
 pscb : 0x8c5d4f20,  bucket : 64672, fw_flags: 0x204 0x20419441,
       scb state: active, scb debug: 0
 nxt_timeout: 360000, refcnt: 1,  ha nak cnt: 0,  rg: 0, sess id: 117753
 hostdb: 0x0, L7: 0x0, stats: 0x8e118e40, child: 0x0
 l4blk0: 78fae7a7 l4blk1: e36df99c l4blk2: 78fae7ea l4blk3: 39080000
 l4blk4: e36df90e l4blk5: 78fae7ea l4blk6: e36df99c l4blk7: fde0000
 l4blk8: 0 l4blk9: 1
 root scb: 0x0 act_blk: 0x8e1115e0
 ingress/egress intf: GigabitEthernet0/0/2 (1021), GigabitEthernet0/0/0 (131065)
 current time 34004163065573 create tstamp: 33985412599209 last access: 33998256774622
```

```
nat_out_local_addr:port: 0.0.0.0:0 nat_in_global_addr:port: 0.0.0.0:0
syncookie fixup: 0x0
halfopen linkage: 0x0 0x0
cxsc_cft_fid: 0x0
tw timer: 0x0 0x0 0x372ba 0x1e89c181
Number of simultaneous packet per session allowed: 25
  bucket 125084 flags 1 func 1 idx 8 wheel 0x8ceb1120
```

Check Firewall Drop Counters

The drop counter output changed during XE 3.9. Before XE 3.9, the firewall drop reasons were very generic. After XE 3.9, the firewall drop reasons were extended to become more granular.

In order to verify drop counters, perform two steps:

1. Confirm the global drop counters in Cisco IOS-XE. These counters show what feature has dropped the traffic. Examples of features include Quality of Service (QoS), Network Address Translation (NAT), Firewall, and so on.

2. Once the subfeature has been identified, query the granular drop counters offered by the subfeature. In this guide, the subfeature being analyzed is the Firewall feature.

Global Drop Counters on QFP

The basic command to rely on provides all the drops across the QFP:

```
Router#show platform hardware qfp active statistics drop
```

This command shows you the generic drops globally across the QFP. These drops can be on any feature. Some example features are:

```
Ipv4Acl
Ipv4NoRoute
Ipv6Acl
Ipv6NoRoute
NatIn2out
VfrErr
...etc
```

In order to see all drops, include counters that have a value of zero, use the command:

```
show platform hardware qfp active statistics drop all
```

In order to clear the counters, use this command. It clears the output after showing it to the screen. This command is clear on read, so the output is reset to zero **after** it is displayed to the screen.

```
show platform hardware qfp active statistics drop clear
```

Below is a list of QFP global firewall drop counters and explanation:

Firewall Global Drop Reason	Explanation
FirewallBackpressure	Packet drop due to backpressure by logging mechanism.
FirewallInvalidZone	No security zone configured for interface.
FirewallL4Insp	L4 policy check failure. See the table below for more granular drop reasons (Firewall feature drop reasons).
FirewallNoForwardingZone	Firewall is uninitialized, and no traffic is allowed to pass.
FirewallNonsession	Session creation fails. It could be due to max session limit has reached or mem

	allocation failure.
FirewallPolicy	The configured Firewall policy is drop.
FirewallL4	L4 inspection failure. See the table below for more granular drop reasons (Fire Feature drop reasons).
FirewallL7	Packet drop due to L7 inspection. See below for a list of more granular L7 dro reasons (Firewall feature drop reasons).
FirewallNotInitiator	Not a session initiator for either TCP, UDP, or ICMP. No session is created. F example, for ICMP the first packet received is not ECHO or TIMESTAMP. For TCP, it is not a SYN.
FirewallNoNewSession	This could happen in normal packet processing or imprecise channel processi Firewall High Availability does not allow new sessions.
FirewallSyncookieMaxDst	In order to provide host-based SYN flood protection, there is a per-destination SYN rate as SYN flood limit. When the number of destination entries reaches limit, new SYN packets are dropped.
FirewallSyncookie	SYNCOOLIE logic is triggered. This indicates SYN/ACK with SYN cookie was sent, and the original SYN packet is dropped.
FirewallARStandby	Asymmetric Routing is not enabled and Redundancy Group is not in active st

Firewall Feature Drop Counters on QFP

The limitation with the QFP global drop counter is that there is no granularity in the drop reasons, and some of the drop reasons such as **FirewallL4** get so overloaded to the point that it is of little use for troubleshooting. This has since been enhanced in Cisco IOS-XE 3.9 (15.3(2)S), where Firewall feature drop counters were added. This gives a much more granular set of drop reasons:

```
ASR#show platform hardware qfp active feature firewall drop all
-----------------------------------------------------------------------------
Drop Reason Packets
-----------------------------------------------------------------------------
Invalid L4 header 0
Invalid ACK flag 0
Invalid ACK number 0
....
```

Below is a list of Firewall feature drop reasons and explanations:

Firewall Feature Drop Reason	**Explanation**
Invalid Header length	The datagram is so small that it could not contain the layer 4TCP,UDP, or ICMP header. could be caused by: 1. TCP header length < 20 2. UDP/ICMP header length < 8
Invalid UDP data length	The UDP datagram length does not match the length specified in the UDP header.
Invalid ACK Number	This drop could be caused by one of these reasons: 1. ACK not equals to the next_seq# of the TCP peer. 2. ACK is greater than the most recent SEQ# sent by the TCP peer.
Invalid ACK Flag	In TCP SYNSENT and SYNRCVD state, it is expected the ACK# is equal to ISN+1 but it This drop could be caused by one of these reasons: 1. Expecting ACK flag but not set in different TCP state. 2. Other than the ACK flag, other flag (like RST) is also set.
Invalid TCP Initiator	This happens when: 1. The first packet from a TCP initiator is not a SYN (Non-initial TCP segment is receiv without a valid session).

	2. The initial SYN packet has the ACK flag set.
SYN with data	The SYN packet contains payload. This is not supported.
Invalid TCP Flags	Invalid TCP flags can be caused by: 1. TCP initial SYN packet has flags other than SYN. 2. In TCP listen state, a TCP peer receives a RST or an ACK. 3. Other responder's packet is received before SYN/ACK. 4. Expected SYN/ACK is not received from the responder.
Invalid Segment in SYNSENT state	An invalid TCP segment in SYNSENT state is caused by: 1. SYN/ACK has payload. 2. SYN/ACK has other flags (PSH, URG, FIN) set. 3. Receive a transit SYN with payload. 4. Receive a non-SYN packet from initiator.
Invalid Segment in SYNRCVD state	An invalid TCP segment in SYNRCVD state could be caused by: 1. Receive a retransit SYN with payload from initiator. 2. Receive an invalid segment which is not SYN/ACK, RST, or FIN from the responder
Invalid SEQ	This occurs in the SYNRCVD state when segments comes from the initiator. It is caused 1. Seq# is less than ISN. 2. If receiver rcvd window size is 0 and: Segment has payload, or Out of order segment (seq# is greater than receiver LASTACK. 3. If receiver rcvd window size is 0 and seq# falls beyond the window. 4. Seq# equals to ISN but not a SYN packet.
Invalid Window Scale Option	Invalid TCP window scale option is caused by incorrect window scale option byte length.
TCP out of window	Packet is too old - one window behind the other side's ACK. This could happen in ESTABLISHED, CLOSEWAIT and LASTACK state.
TCP extra payload after FIN sent	Payload received after FIN sent. This could happen in CLOSEWAIT state.
TCP Window Overflow	This occurs when incoming segment size overflows receiver's window. However, if vTCP enabled, this condition is allowed because firewall needs to buffer the segment for ALG to consume later.
Retran with Invalid Flags	A retransmitted packet was already acknowledged by the receiver.
TCP out-of-order Segment	The Out-Of-Order packet is about to be delivered to L7 for inspection. If L7 does not allow OOO segment, this packet will be dropped.
SYN Flood	Under a TCP SYN flood attack. Under certain conditions when the current connections to host exceeds the configured half-open value the firewall will reject any new connections t IP address for a period of time. As a result the packets will be dropped.
Internal Err - synflood check alloc Failed	During synflood check, allocation of hostdb fails. Recommended action: check "show platform hardware qfp active feature firewall memory check the memory status.
Synflood blackout drop	If configured half-open connections is exceeded and blackout time is configured, all new connection to this IP address are dropped.
Half-Open Session Limit Exceed	Packet dropped due to the allowed half-opened sessions exceeded. Also check the settings of "max-incomplete high/low" and "one-minute high/low" to make the # of half-opened sessions are not being throttled by these configurations.
Too Many Pkt per flow	The maximum number of inspectable packets allowed per flow is exceeded. The max nu is 25.
Too many ICMP error	The maximum number of ICMP error packets allowed per flow is exceeded. The maximu number is 3.

packets per flow	
Unexpect TCP payload from Rsp to Init	In SYNRCVD state, TCP receives a packet with payload from responder to initiator direct
Internal Error - Undefined Direction	Packet direction undefined.
SYN inside current window	A SYN packet is seen within the window of an already established TCP connection.
RST inside current window	A RST packet is observed within the window of an already established TCP connection.
Stray Segment	A TCP segment is received that should not have been received through the TCP state machine such as a TCP SYN packet being received in the listen state from the responde
ICMP Internal Error - Missed ICMP NAT info	The ICMP packet is nat'ed but internal NAT info is missing. This is an internal error.
ICMP packet in SCB close state	Received an ICMP packet in SCB CLOSE state.
Missed IP header in ICMP packet	Missing IP header in ICMP packet.
ICMP Error No IP or ICMP	ICMP error packet without IP or ICMP in payload. Probably caused by a malformed pack an attack.
ICMP Err Pkt Too Short	ICMP Error packet is too short.
ICMP Err Exceed Burst Limit	ICMP error pkt exceeds the burst limit of 10.
ICMP Err Unreachable	ICMP error pkt unreachable exceed limit. Only the 1 unreachable packet is allowed to pas through.
ICMP Err Invalid Seq#	Seq# of embedded packet doesn't match the seq# of the packet that originates the ICMP
ICMP Err Invalid Ack	Invalid ACK in the ICMP Error embedded packet.
ICMP action drop	The configured ICMP action is drop.
Zone-pair without policy-map	Policy not present on zone-pair. it could be due to ALG (Application Layer Gateway) not b configured to open pinhole for application data channel, or ALG didn't open pinhole corre or no pinhole is opened due to scalability issues.
Session Missed And Policy Not Present	Session lookup failed and no policy is present to inspect this packet.
ICMP Error And Policy Not Present	ICMP Error with no policy configured on zone-pair.
Classification Failed	Classification failure in a given zone pair when Firewall tries to determine if protocol is inspectable.
Classification Action Drop	Classification action is drop.
Security Policy Misconfig	Failed classification due to security policy misconfiguration. This could also be due to no pinpole for L7 data channel.
Send RST to responder	Send RST to responder in SYNSENT state when ACK# is not equal to ISN+1.
Firewall Policy	Policy action is to drop.

Drop Fragment Drop	Drop remaining fragments when the first fragment is dropped.
ICMP Firwall Policy Drop	Policy action of the ICMP embedded packet is DROP.
L7 inspection returns DROP	L7 (ALG) decides to drop packet. The reason could be found from different ALG statistics
L7 Segment Pkt Not Allow	Received segmented packet when ALG does not honor it.
L7 Fragment Pkt Not Allow	Received fragmented (or VFR) packets when ALG does not honor it.
Unknown L7 Proto Type	Unrecognized protocol type.

Troubleshoot Firewall Drops

Once the drop reason is identified from the above global or firewall feature drop counters, additional troubleshooting steps might be needed if these drops are unexpected. Apart from configuration validation in order to ensure the configuration is correct for the firewall functionalities enabled, it is often required to take packet captures for the traffic flow in question to see if the packets are malformed or if there is any protocol or application implementation issues.

Logging

ASR logging functionality generates syslogs in order to record dropped packets. These syslogs provide more details on why the packet was dropped. There are two types of sysloggings:

1. Local buffered syslogging

2. Remote high speed logging

Local Buffered Syslogging

In order to isolate the cause of the drops, you can use generic ZBFW troubleshooting, such as enabling log drops. There are two ways to configure packet drop logging.

Method 1: Use inspect-global parameter-map in order to log all dropped packets.

```
parameter-map type inspect-global     log dropped-packets
```

Method 2: Use custom inspect parameter-map in order to log dropped packets for only specific class.

```
parameter-map type inspect LOG_PARAM
log dropped-packets
!
policy-map type inspect ZBFW_PMAP
class type inspect ZBFW_CMAP
inspect LOG_PARAM
```

These messages are sent to the log or console depending on how the ASR is configured for logging. Here is an example of a drop log message.

```
*Apr  8 13:20:39.075: %IOSXE-6-PLATFORM: F0: cpp_cp: QFP:0.0 Thread:103
```

```
TS:00000605668054540031 %FW-6-DROP_PKT: Dropping tcp pkt from GigabitEthernet0/0/2
14.38.112.250:41433 => 14.36.1.206:23(target:class)-(INSIDE_OUTSIDE_ZP:class-default)
due to Policy drop:classify result with ip ident 11579 tcp flag 0x2, seq 2014580963,
ack 0
```

Limitations of Local Buffered Syslogging

1. These logs are rate limited as per Cisco bug ID CSCud09943.

2. These logs might not print unless specific configuration is applied. For example, packets dropped by class-default packets will not be logged unless the **log** keyword is specified:

```
policy-map type inspect ZBFW_PMAP
class class-default
drop log
```

Remote High Speed Logging

High speed logging (HSL) generates syslogs directly from the QFP and sends it to the configured netflow HSL collector. This is the recommended logging solution for ZBFW on ASR.

For HSL, use this configuration:

```
parameter-map type inspect inspect-global
  log template timeout-rate 1
  log flow-export v9 udp destination 1.1.1.1 5555
```

In order to use this configuration, a netflow collector capable of Netflow Version 9 is required. This is detailed in

Configuration Guide: Zone-Based Policy Firewall, Cisco IOS XE Release 3S (ASR 1000) Firewall High-Speed Logging

Packet Tracing Using Conditional Matching

Turn on conditional debugs in order to enable packet tracing and then enable packet tracing for these features:

```
ip access-list extended CONDITIONAL_ACL
 permit ip host 10.1.1.1 host 192.168.1.1
 permit ip host 192.168.1.1 host 10.1.1.1
!
debug platform condition feature fw dataplane submode all level info
debug platform condition ipv4 access-list CONDITIONAL_ACL both
```

> **Note**: The match condition can use the IP address directly, as an ACL is not necessary. This will match as source or destination which allows for bidirectional traces. This method can be used if you are not allowed to alter the configuration. For example: debug platform condition ipv4 address 192.168.1.1/32.

Turn on the packet-tracing feature:

```
debug platform packet-trace copy packet both
debug platform packet-trace packet 16
```

```
debug platform packet-trace drop
debug platform packet-trace enable
```

There are two ways to use this feature:

1. Enter the **debug platform packet-trace drop** command in order to trace only the dropped packets.

2. Exclusion of the command **debug platform packet-trace drop** will trace any packet that matches the condition, which includes ones that are inspected/passed by the device.

Turn on conditional debugs:

```
debug platform condition start
```

Run the test, then turn off debugs:

```
debug platform condition stop
```

Now the information can be displayed to the screen. In this example, ICMP packets were dropped due to a firewall policy:

```
Router#show platform packet-trace statistics
Packets Summary
 Matched  2
 Traced   2
Packets Received
 Ingress  2
 Inject   0
Packets Processed
 Forward  0
 Punt     0
 Drop     2
   Count       Code  Cause
   2           183   FirewallPolicy
 Consume  0
```

```
 Router#show platform packet-trace summary
Pkt   Input            Output           State  Reason
0     Gi0/0/2          Gi0/0/0          DROP   183 (FirewallPolicy)
1     Gi0/0/2          Gi0/0/0          DROP   183 (FirewallPolicy)
```

```
 Router#show platform packet-trace packet 0
Packet: 0           CBUG ID: 2980
Summary
 Input     : GigabitEthernet0/0/2
 Output    : GigabitEthernet0/0/0
 State     : DROP 183 (FirewallPolicy)
 Timestamp
   Start   : 1207843476722162 ns (04/15/2014 12:37:01.103864 UTC)
   Stop    : 1207843477247782 ns (04/15/2014 12:37:01.104390 UTC)
Path Trace
 Feature: IPV4
   Source      : 10.1.1.1
   Destination : 192.168.1.1
   Protocol    : 1 (ICMP)
 Feature: ZBFW
   Action  : Drop
   Reason  : ICMP policy drop:classify result
   Zone-pair name  : INSIDE_OUTSIDE_ZP
   Class-map name  : class-default
Packet Copy In
 c89c1d51 5702000c 29f9d528 08004500 00540000 40004001 ac640e26 70fa0e24
```

```
 01010800 172a2741 00016459 4d5310e4 0c000809 0a0b0c0d 0e0f1011 12131415
Packet Copy Out
 c89c1d51 5702000c 29f9d528 08004500 00540000 40003f01 ad640e26 70fa0e24
 01010800 172a2741 00016459 4d5310e4 0c000809 0a0b0c0d 0e0f1011 12131415
```

The **show platform packet-trace packet <num> decode** command decodes the packet header information and contents. This feature was introduced in XE3.11:

```
Router#show platform packet-trace packet all decode
Packet: 0           CBUG ID: 2980
Summary
 Input     : GigabitEthernet0/0/2
 Output    : GigabitEthernet0/0/0
 State     : DROP 183 (FirewallPolicy)
 Timestamp
   Start   : 1207843476722162 ns (04/15/2014 12:37:01.103864 UTC)
   Stop    : 1207843477247782 ns (04/15/2014 12:37:01.104390 UTC)
Path Trace
 Feature: IPV4
    Source      : 10.1.1.1
    Destination : 192.168.1.1
    Protocol    : 1 (ICMP)
  Feature: ZBFW
    Action  : Drop
    Reason  : ICMP policy drop:classify result
    Zone-pair name  : INSIDE_OUTSIDE_ZP
    Class-map name  : class-default
Packet Copy In
 c89c1d51 5702000c 29f9d528 08004500 00540000 40004001 ac640e26 70fa0e24
 01010800 172a2741 00016459 4d5310e4 0c000809 0a0b0c0d 0e0f1011 12131415
 ARPA
    Destination MAC     : c89c.1d51.5702
    Source MAC          : 000c.29f9.d528
   Type                : 0x0800 (IPV4)
 IPv4
   Version             : 4
   Header Length       : 5
   ToS                 : 0x00
   Total Length        : 84
   Identifier          : 0x0000
   IP Flags            : 0x2 (Don't fragment)
   Frag Offset         : 0
   TTL                 : 64
   Protocol            : 1 (ICMP)
   Header Checksum     : 0xac64
   Source Address      : 10.1.1.1
   Destination Address : 192.168.1.1
 ICMP
   Type                : 8 (Echo)
   Code                : 0 (No Code)
   Checksum            : 0x172a
   Identifier          : 0x2741
   Sequence            : 0x0001
Packet Copy Out
 c89c1d51 5702000c 29f9d528 08004500 00540000 40003f01 ad640e26 70fa0e24
 01010800 172a2741 00016459 4d5310e4 0c000809 0a0b0c0d 0e0f1011 12131415
 ARPA
   Destination MAC     : c89c.1d51.5702
   Source MAC          : 000c.29f9.d528
   Type                : 0x0800 (IPV4)
 IPv4
   Version             : 4
   Header Length       : 5
   ToS                 : 0x00
```

```
   Total Length        : 84
   Identifier          : 0x0000
   IP Flags            : 0x2 (Don't fragment)
   Frag Offset         : 0
   TTL                 : 63
   Protocol            : 1 (ICMP)
   Header Checksum     : 0xad64
   Source Address      : 10.1.1.1
   Destination Address : 192.168.1.1
 ICMP
   Type                : 8 (Echo)
   Code                : 0 (No Code)
   Checksum            : 0x172a
   Identifier          : 0x2741
   Sequence            : 0x0001
```

Embedded Packet Capture

Embedded Packet Capture support has been added in Cisco IOS-XE 3.7 (15.2(4)S). For more details, see

Embedded Packet Capture for Cisco IOS and IOS-XE Configuration Example.

Debugs

Conditional Debugs

In XE3.10, conditional debugs will be introduced. Conditional statements can be used in order to ensure the ZBFW feature only logs debug messages that are relevant to the condition. Conditional debugs use ACLs in order to restrict logs that match the ACL elements. Also, prior to XE3.10, the debug messages were more difficult to read. The debug output was improved in XE3.10 to make them easier to understand.

In order to enable these debugs, issue this command:

```
debug platform condition feature fw dataplane submode [detail | policy | layer4 | drop]
debug platform condition ipv4 access-list <ACL_name> both
debug platform condition start
```

Notice that the condition command must be set via an ACL and directionality. The conditional debugs will not be implemented until the they are started with the command **debug platform condition start**. In order to turn off conditional debugs use the command **debug platform condition stop**.

```
debug platform condition stop
```

In order to turn off conditional debugs, **DO NOT** use the command **undebug all**. In order to turn off all conditional debugs, use the command:

```
ASR#clear platform condition all
```

Prior to XE3.14, **ha** and **event** debugs are not conditional. As a result, the command **debug platform condition feature fw dataplane submode all** causes all logs to be created, independent of the condition selected below. This could create additional noise that makes debugging difficult.

By default, the conditional logging level is **info**. In order to increase/decrease the level of logging, use the command:

```
debug platform condition feature fw dataplane submode all [verbose | warning]
```

Gather and View Debugs

Debug files will not print to the console or monitor. All debugs are written to the hard disk of the ASR. Debugs are written to the hard disk under the folder **tracelogs** with the name **cpp_cp_F0-0.log.<date>**. In order to view the file where debugs are written, use the output:

```
ASR# cd harddisk:
ASR# cd tracelogs
ASR# dir cpp_cp_F0*Directory of harddisk:/tracelogs/cpp_cp_F0*

Directory of harddisk:/tracelogs/

3751962 -rwx 1048795 Jun 15 2010 06:31:51 +00:00
cpp_cp_F0-0.log.5375.20100615063151
3751967 -rwx 1048887 Jun 15 2010 02:18:07 +00:00
cpp_cp_F0-0.log.5375.20100615021807
39313059840 bytes total (30680653824 bytes free)
```

Each debug file will be stored as a **cpp_cp_F0-0.log.<date>** file. These are regular text files that can be copied off the ASR with TFTP. The log file maximum on the ASR is 1Mb. After 1Mb, the debugs are written to a new log file. That is why each log file is timestamped in order to indicate the start of the file.

Log files might exist in these locations:

```
harddisk:/tracelogs/
bootflash:/tracelogs/
```

Since log files are only displayed after they are rotated, the log file can be manually rotated with this command:

```
ASR#test platform software trace slot f0 cpp-control-process rotate
```

This immediately creates a "cpp_cp" log file and starts a new one on the QFP. For example:

```
ASR#test platform software trace slot f0 cpp-control-process rotate
 Rotated file from: /tmp/fp/trace/stage/cpp_cp_F0-0.log.7311.20140408134406,
Bytes: 82407, Messages: 431

ASR#more tracelogs/cpp_cp_F0-0.log.7311.20140408134406
04/02 10:22:54.462 : btrace continued for process ID 7311 with 159 modules
04/07 16:52:41.164 [cpp-dp-fw]: (info): QFP:0.0 Thread:110 TS:00000531990811543397
:FW_DEBUG_FLG_HA:[]: HA[1]: Changing HA state to 9
04/07 16:55:23.503 [cpp-dp-fw]: (info): QFP:0.0 Thread:120 TS:00000532153153672298
:FW_DEBUG_FLG_HA:[]: HA[1]: Changing HA state to 10
04/07 16:55:23.617 [buginf]: (debug): [system] Svr HA bulk sync CPP(0) complex(0)
epoch(0) trans_id(26214421) rg_num(1)
```

This command allows the debug files to be merged into a single file for easier processing. It merges all files in the directory and interlaces them based on time. This can help when the logs are very verbose and are created across multiple files:

```
ASR#request platform software trace slot rp active merge target bootflash:MERGED_OUTPUT.log
Creating the merged trace file: [bootflash:MERGED_OUTPUT.log]
 including all messages

Done with creation of the merged trace file: [bootflash:MERGED_OUTPUT.log]
```

Was this document helpful? Yes No

Thank you for your feedback.

Open a Support Case (Requires a Cisco Service Contract.)

Related Cisco Support Community Discussions

The Cisco Support Community is a forum for you to ask and answer questions, share suggestions, and collaborate with your peers.

Refer to Cisco Technical Tips Conventions for information on conventions used in this document.