Analyze Firepower Firewall Captures to Effectively Troubleshoot Network Issues

Contents

Introduction

This document describes various packet capture analysis techniques that aim to effectively troubleshoot network issues.

Prerequisites

Requirements

Cisco recommends that you have knowledge of these topics:

- Firepower platform architecture
- NGFW logs
- NGFW packet-tracer

Additionally, before you start to analyze packet captures it is highly advisable to meet these requirements:

- **Know the protocol operation** - Do not start to check a packet capture if you do not understand how the captured protocol operates.
- **Know the topology** - You must know the transit devices end-to-end. If this is not possible, you must at least know the upstream and downstream devices.
- **Know the appliance** - You must know how your device handles packets, what are the involved interfaces (ingress/egress), what is the device architecture, and what are the various capture points.
- **Know the configuration** - You must know how a packet flow is supposed to be handled by the device in terms of:
  - Routing/Egress Interface
  - Policies applied
  - Network Address Translation (NAT)
- **Know the available tools** - Along with the captures, it is recommended to be ready to apply other tools and techniques (like logging and tracers) and if needed, correlate them with the captured packets

Components Used

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

- Most of the scenarios are based on FP4140 running FTD software 6.5.x.
- FMC running software 6.5.x.

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

Background Information

Packet capture is one of the most overlooked troubleshoot tools available today. Daily, Cisco TAC solves many problems with analysis of captured data.

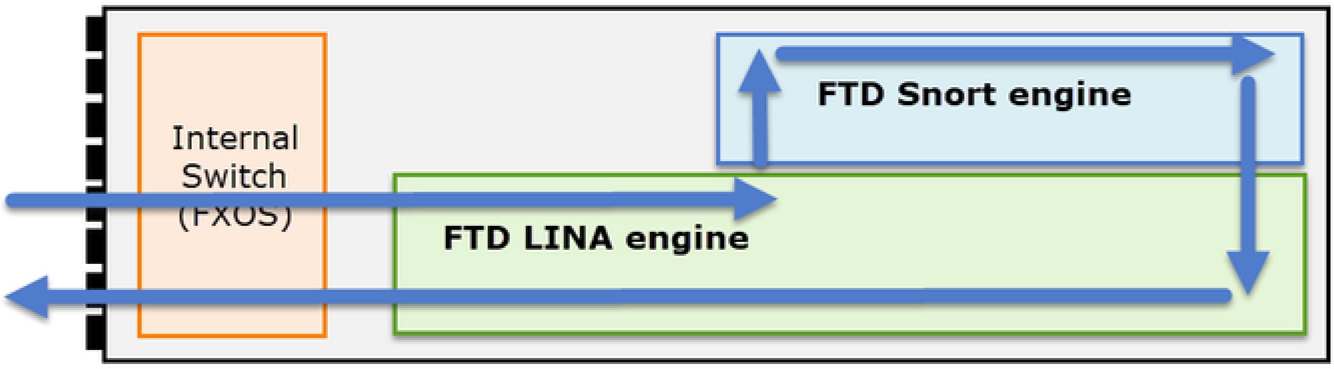
The goal of this document is to help network and security engineers to identify and troubleshoot common network issues based mainly on packet capture analysis.

All the scenarios presented in this document are based on real user cases seen in the Cisco Technical Assistance Center (TAC).

The document covers the packet captures from a Cisco Next-Generation Firewall (NGFW) point of view, but the same concepts are applicable to other device types as well.

How to Collect and Export Captures on the NGFW Product Family?

In the case of a Firepower appliance (1xxx, 21xx, 41xx, 93xx) and a Firepower Threat Defense (FTD) application a packet processing can be visualized as shown in the image.



1. A packet enters the ingress interface and it is handled by the chassis internal switch.
2. The packet enters the FTD Lina engine which does mainly L3/L4 checks.
3. If the policy requires the packet is inspected by the Snort engine (mainly L7 inspection).
4. The Snort engine returns a verdict for the packet.
5. The LINA engine drops or forwards the packet based on Snortâ€™s verdict.
6. The packet egresses the chassis through the internal chassis switch.

Based on the shown architecture, the FTD captures can be taken in three (3) different places:

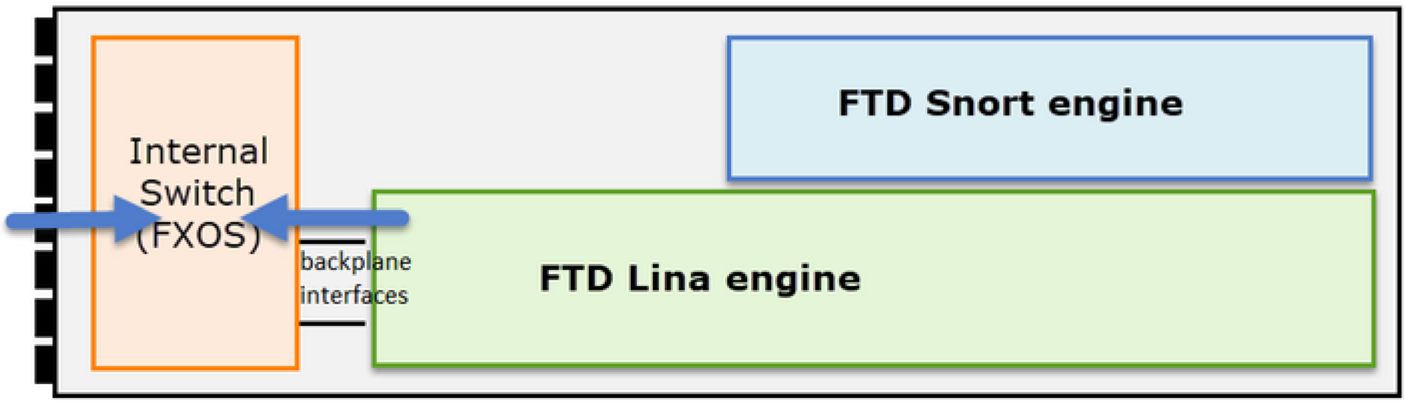
- FXOS
- FTD Lina engine
- FTD Snort engine

Collect FXOS Captures

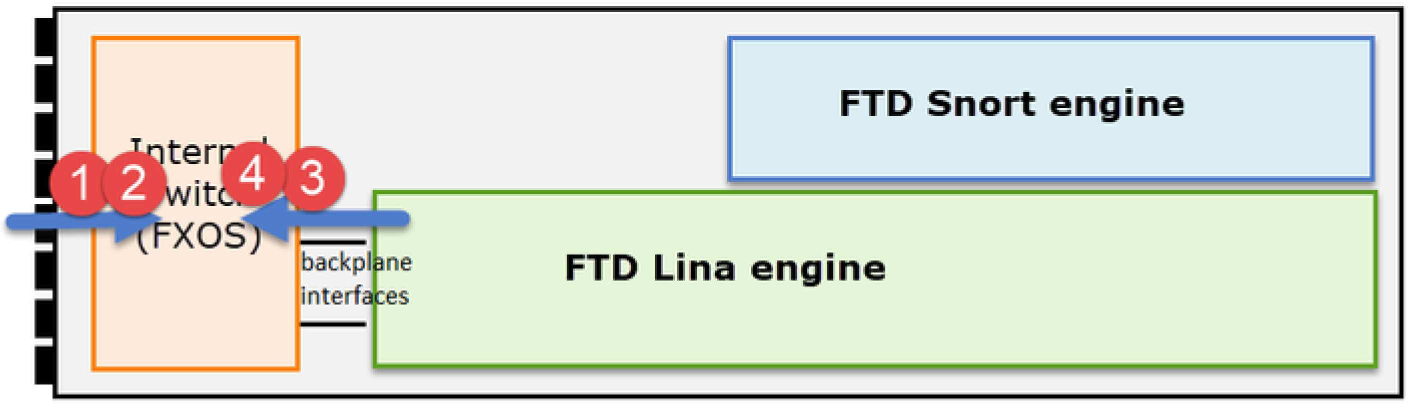
The process is described in this document:

https://www.cisco.com/c/en/us/td/docs/security/firepower/fxos/fxos271/web-guide/b_GUI_FXOS_ConfigGuide_271/troubleshooting.html#concept_E8823CC63C934A909BBC0DF12F301DEI

FXOS captures can be only taken in the ingress direction from the internal switch point of view are shown in the image here.



Shown here, these are two capture points per direction (due to internal switch architecture).



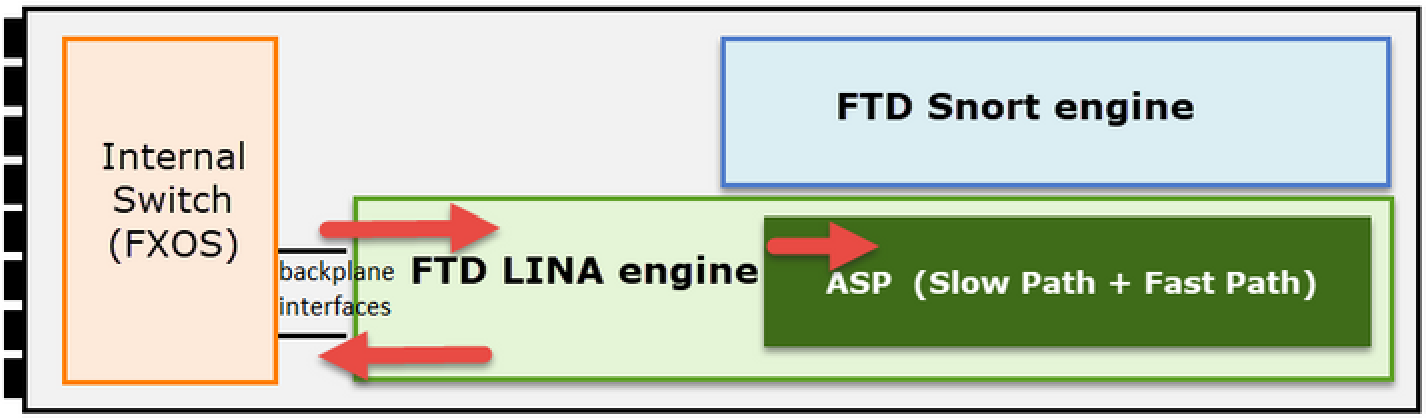
Captured packets in points 2, 3, and 4 have a virtual network tag (VNTag).

Note: FXOS chassis-level captures are only available on FP41xx and FP93xx platforms. FP1xxx and FP21xx do not provide this capability.

Enable and Collect FTD Lina Captures

Main capture points:

- Ingress interface
- Egress interface
- Accelerated Security Path (ASP)



You can use either Firepower Management Center User Interface (FMC UI) or FTD CLI to enable and collect the FTD Lina captures.

Enable capture from CLI on the INSIDE interface:

<#root>

```
firepower#
```

```
capture CAPI interface INSIDE match icmp host 192.168.103.1 host 192.168.101.1
```

This capture matches the traffic between IPs 192.168.103.1 and 192.168.101.1 in both directions.

Enable ASP capture to see all packets dropped by the FTD Lina engine:

<#root>

```
firepower#
```

```
capture ASP type asp-drop all
```

Export an FTD Lina capture to an FTP server:

<#root>

```
firepower#
```

```
copy /pcap capture:CAPI ftp://ftp_username:ftp_password@192.168.78.73/CAPI.pcap
```

Export an FTD Lina capture to a TFTP server:

<#root>

```
firepower#
```

```
copy /pcap capture:CAPI tftp://192.168.78.73
```

As from FMC 6.2.x version you can enable and collect FTD Lina captures from FMC UI.

Another way to collect FTD captures from an FMC-managed firewall is this.

Step 1

In case of LINA or ASP capture copy the capture to the FTD disk.

<#root>

```
firepower#
```

`copy /pcap capture:capin disk0:capin.pcap`

```
Source capture name [capin]?

Destination filename [capin.pcap]?
!!!!
```

Step 2

Navigate to expert mode, locate the saved capture, and copy it to the /ngfw/var/common location:

<#root>

```
firepower#

Console connection detached.

>
```

`expert`

```
admin@firepower:~$
```

`sudo su`

```
Password:
root@firepower:/home/admin#
```

` cd /mnt/disk0`

```
root@firepower:/mnt/disk0#
```

`ls -al | grep pcap`

```
-rwxr-xr-x 1 root root     24 Apr 26 18:19 CAPI.pcap
-rwxr-xr-x 1 root root  30110 Apr  8 14:10
```

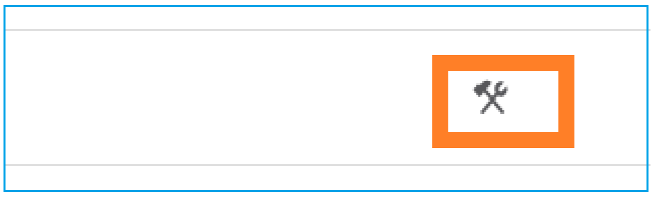
`capin.pcap`

```
-rwxr-xr-x 1 root root   6123 Apr  8 14:11 capin2.pcap
root@firepower:/mnt/disk0#
```

`cp capin.pcap /ngfw/var/common`

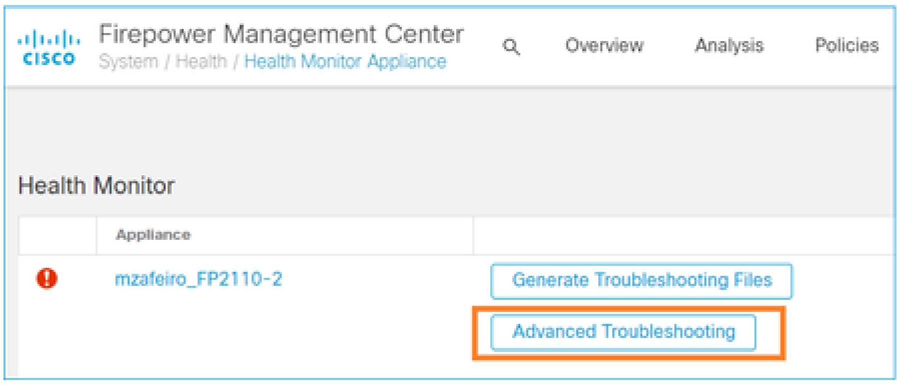
Step 3

Login to the FMC that manages the FTD and navigate to **Devices** > **Device Management.** Locate the FTD device and select the **Troubleshoot** icon:



Step 4

Select **Advanced Troubleshooting:**

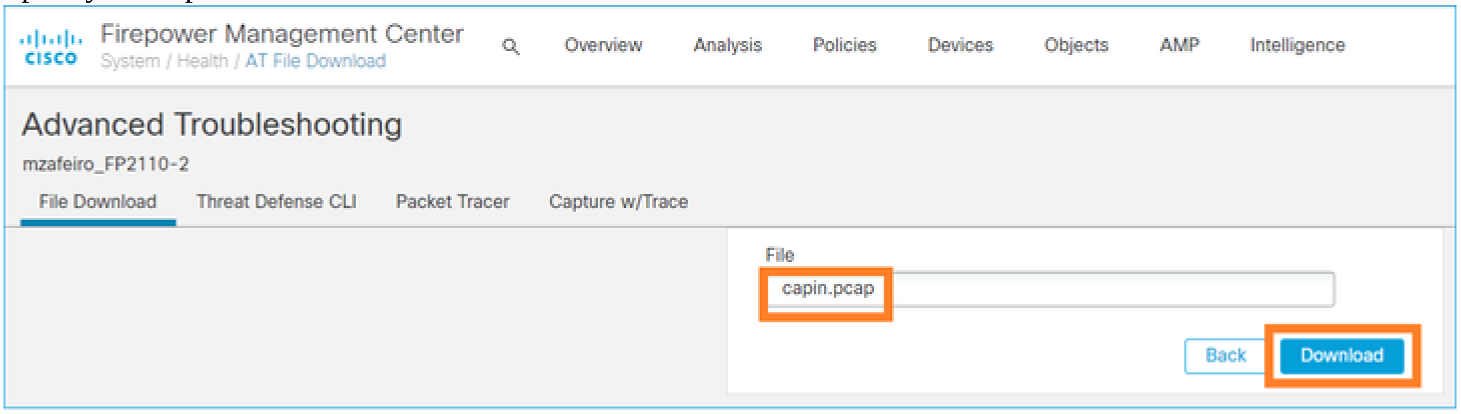


Firepower Management Center
System / Health / Health Monitor Appliance

Health Monitor

	Appliance	
●	mzafeiro_FP2110-2	Generate Troubleshooting Files Advanced Troubleshooting

Specify the capture file name and select **Download:**



Firepower Management Center
System / Health / AT File Download

Advanced Troubleshooting
mzafeiro_FP2110-2

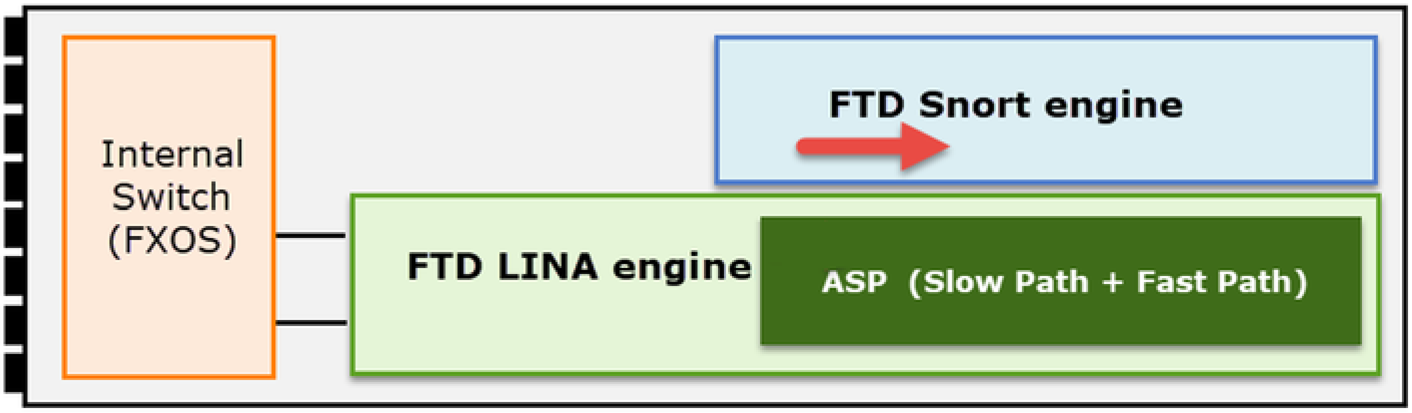
File: capin.pcap

For more examples on how to enable/collect captures from the FMC UI check this document:

https://www.cisco.com/c/en/us/support/docs/security/firepower-ngfw/212474-working-with-firepower-threat-defense-f.html

Enable and Collect FTD Snort Captures

The capture point is shown in the image here.



Enable Snort-level capture:

```
<#root>

>

capture-traffic


Please choose domain to capture traffic from:
  0 - br1
  1 - Router

Selection?

1


Please specify tcpdump options desired.
(or enter '?' for a list of supported options)
Options:

-n host 192.168.101.1
```

To write the capture to a file with name capture.pcap and copy it via FTP to a remote server:

```
<#root>

>

capture-traffic


Please choose domain to capture traffic from:
  0 - br1
  1 - Router

Selection?

1


Please specify tcpdump options desired.
(or enter '?' for a list of supported options)
Options:
```

```
-w capture.pcap host 192.168.101.1


CTRL + C <- to stop the capture


>

file copy 10.229.22.136 ftp / capture.pcap

Enter password for ftp@10.229.22.136:
Copying capture.pcap
Copy successful.

>
```

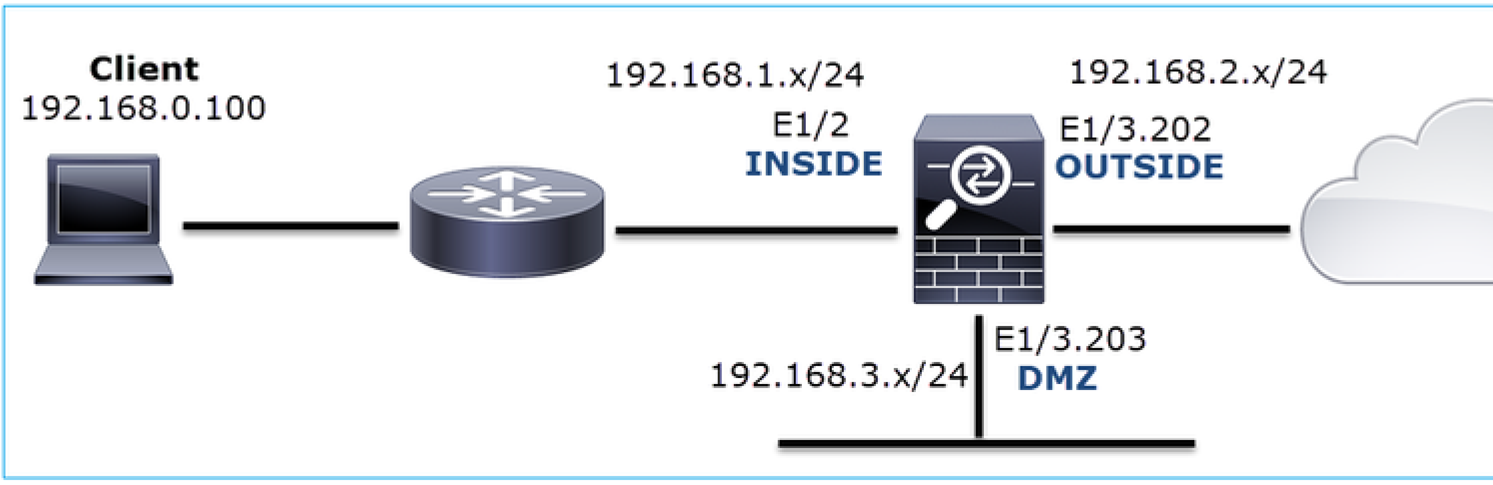
For more Snort-level capture examples that include different capture filters check this document:

https://www.cisco.com/c/en/us/support/docs/security/firepower-ngfw/212474-working-with-firepower-threat-defense-f.html

Troubleshoot

Case 1. No TCP SYN on Egress Interface

The topology is shown in the image here:



Problem Description: HTTP does not work

Affected Flow:

Src IP: 192.168.0.100

Dst IP: 10.10.1.100

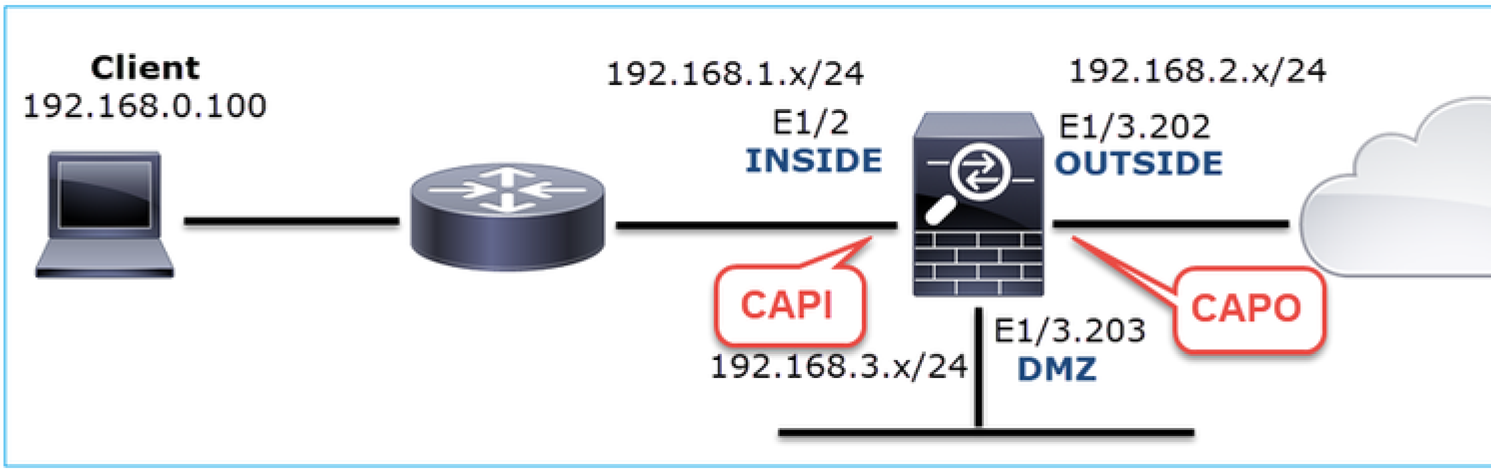
Protocol: TCP 80

Capture Analysis

Enable captures on the FTD LINA engine:

<#root>

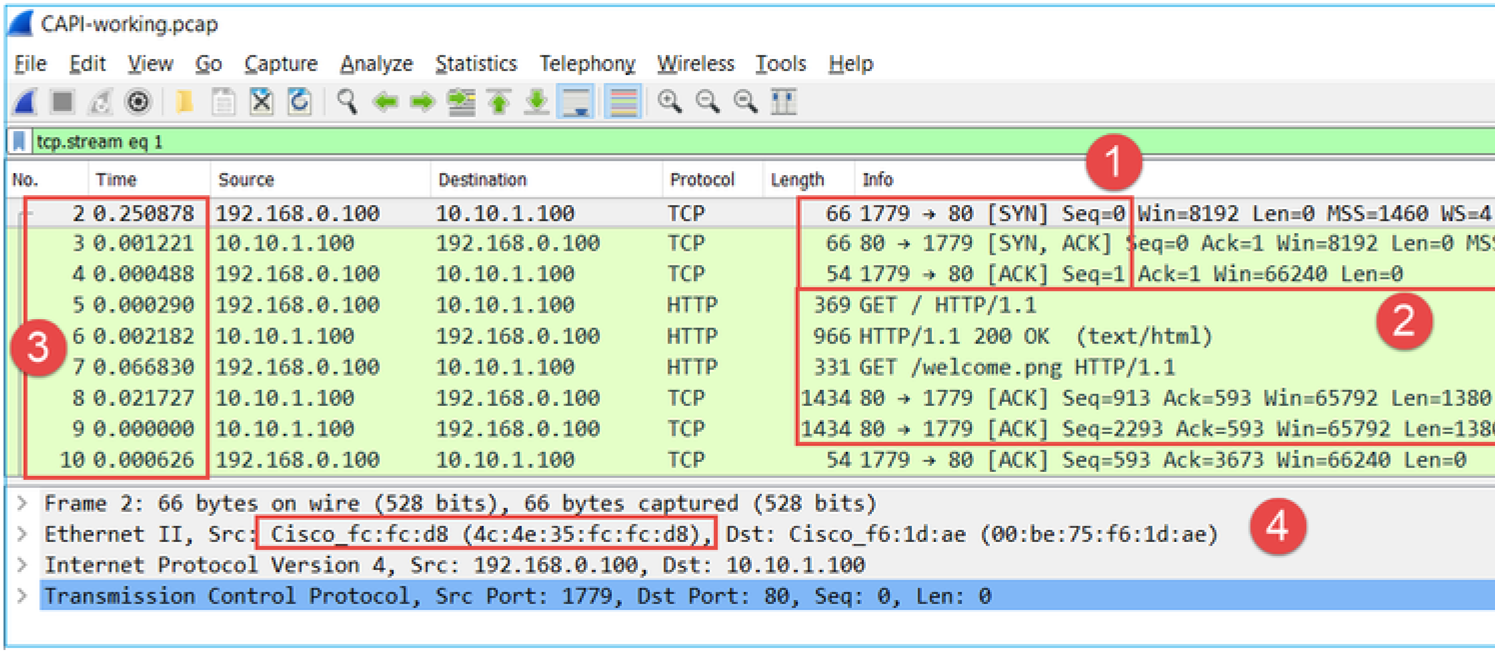
```
firepower#
capture CAPI int INSIDE match ip host 192.168.0.100 host 10.10.1.100
firepower#
capture CAPO int OUTSIDE match ip host 192.168.0.100 host 10.10.1.100
```



Captures - Functional Scenario:

As a baseline, it is always very useful to have captures from a functional scenario.

Capture taken on NGFW INSIDE interface, is as shown in the image:

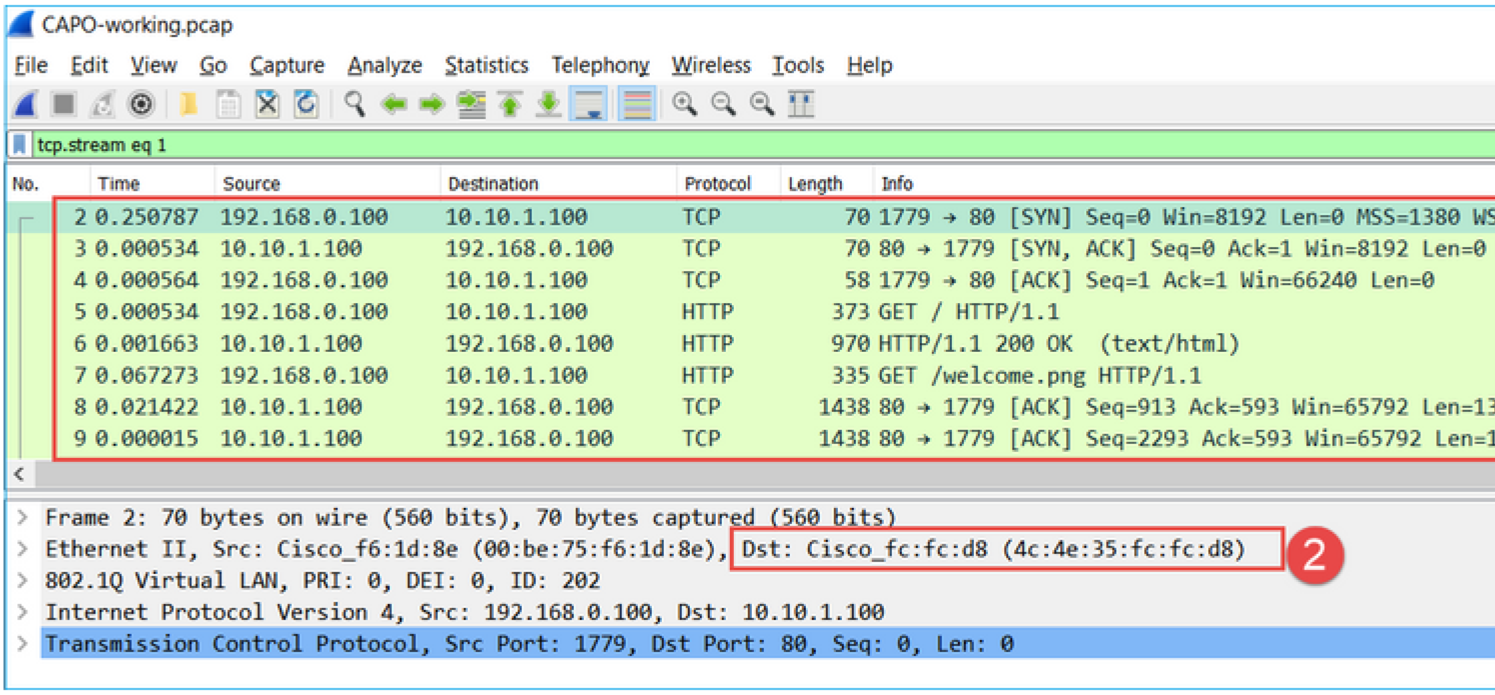

CAPI-working.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
tcp.stream eq 1
1
2
3
4

```
No.  Time       Source          Destination     Protocol  Length  Info
   2 0.250878   192.168.0.100   10.10.1.100     TCP         66 1779 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4
   3 0.001221   10.10.1.100     192.168.0.100   TCP         66 80 → 1779 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MS
   4 0.000488   192.168.0.100   10.10.1.100     TCP         54 1779 → 80 [ACK] Seq=1 Ack=1 Win=66240 Len=0
   5 0.000290   192.168.0.100   10.10.1.100     HTTP       369 GET / HTTP/1.1
   6 0.002182   10.10.1.100     192.168.0.100   HTTP       966 HTTP/1.1 200 OK  (text/html)
   7 0.066830   192.168.0.100   10.10.1.100     HTTP       331 GET /welcome.png HTTP/1.1
   8 0.021727   10.10.1.100     192.168.0.100   TCP       1434 80 → 1779 [ACK] Seq=913 Ack=593 Win=65792 Len=1380
   9 0.000000   10.10.1.100     192.168.0.100   TCP       1434 80 → 1779 [ACK] Seq=2293 Ack=593 Win=65792 Len=138
  10 0.000626   192.168.0.100   10.10.1.100     TCP         54 1779 → 80 [ACK] Seq=593 Ack=3673 Win=66240 Len=0

> Frame 2: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
> Ethernet II, Src: Cisco_fc:fc:d8 (4c:4e:35:fc:fc:d8), Dst: Cisco_f6:1d:ae (00:be:75:f6:1d:ae)
> Internet Protocol Version 4, Src: 192.168.0.100, Dst: 10.10.1.100
> Transmission Control Protocol, Src Port: 1779, Dst Port: 80, Seq: 0, Len: 0
```

Key Points:

1. TCP 3-way handshake.
2. Bidirectional data exchange.
3. No delays between the packets (based on the time difference between the packets)
4. Source MAC is the correct downstream device.

Capture taken on NGFW OUTSIDE interface, is shown in the image here:


CAPO-working.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
2

```
tcp.stream eq 1

No.  Time      Source         Destination    Protocol  Length Info
   2 0.250787  192.168.0.100  10.10.1.100    TCP         70 1779 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1380 WS
   3 0.000534  10.10.1.100    192.168.0.100  TCP         70 80 → 1779 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0
   4 0.000564  192.168.0.100  10.10.1.100    TCP         58 1779 → 80 [ACK] Seq=1 Ack=1 Win=66240 Len=0
   5 0.000534  192.168.0.100  10.10.1.100    HTTP       373 GET / HTTP/1.1
   6 0.001663  10.10.1.100    192.168.0.100  HTTP       970 HTTP/1.1 200 OK  (text/html)
   7 0.067273  192.168.0.100  10.10.1.100    HTTP       335 GET /welcome.png HTTP/1.1
   8 0.021422  10.10.1.100    192.168.0.100  TCP       1438 80 → 1779 [ACK] Seq=913 Ack=593 Win=65792 Len=13
   9 0.000015  10.10.1.100    192.168.0.100  TCP       1438 80 → 1779 [ACK] Seq=2293 Ack=593 Win=65792 Len=1

> Frame 2: 70 bytes on wire (560 bits), 70 bytes captured (560 bits)
> Ethernet II, Src: Cisco_f6:1d:8e (00:be:75:f6:1d:8e), Dst: Cisco_fc:fc:d8 (4c:4e:35:fc:fc:d8)
> 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 202
> Internet Protocol Version 4, Src: 192.168.0.100, Dst: 10.10.1.100
> Transmission Control Protocol, Src Port: 1779, Dst Port: 80, Seq: 0, Len: 0
```

Key Points:

1. Same data as in the CAPI capture.
2. Destination MAC is the correct upstream device.

Captures - Non-functional scenario

From the device CLI the captures look like this:

<#root>

```
firepower#

show capture

capture CAPI type raw-data interface INSIDE

[Capturing - 484 bytes]

  match ip host 192.168.0.100 host 10.10.1.100
capture CAPO type raw-data interface OUTSIDE

[Capturing - 0 bytes]

  match ip host 192.168.0.100 host 10.10.1.100
```

CAPI contents:

<#root>

```
firepower#
```

```
show capture CAPI
```

```
6 packets captured

   1: 11:47:46.911482   192.168.0.100.3171 > 10.10.1.100.80:
```

```
S
```

```
 1089825363:1089825363(0) win 8192 <mss 1460,nop,wscale 2,nop,nop,sackOK>
   2: 11:47:47.161902   192.168.0.100.3172 > 10.10.1.100.80:
```

```
S
```

```
 3981048763:3981048763(0) win 8192 <mss 1460,nop,wscale 2,nop,nop,sackOK>
   3: 11:47:49.907683   192.168.0.100.3171 > 10.10.1.100.80:
```

```
S
```

```
 1089825363:1089825363(0) win 8192 <mss 1460,nop,wscale 2,nop,nop,sackOK>
   4: 11:47:50.162757   192.168.0.100.3172 > 10.10.1.100.80:
```

```
S
```

```
 3981048763:3981048763(0) win 8192 <mss 1460,nop,wscale 2,nop,nop,sackOK>
   5: 11:47:55.914640   192.168.0.100.3171 > 10.10.1.100.80:
```

```
S
```

```
 1089825363:1089825363(0) win 8192 <mss 1460,nop,nop,sackOK>
   6: 11:47:56.164710   192.168.0.100.3172 > 10.10.1.100.80:
```

```
S
```

```
 3981048763:3981048763(0) win 8192 <mss 1460,nop,nop,sackOK>
```

<#root>

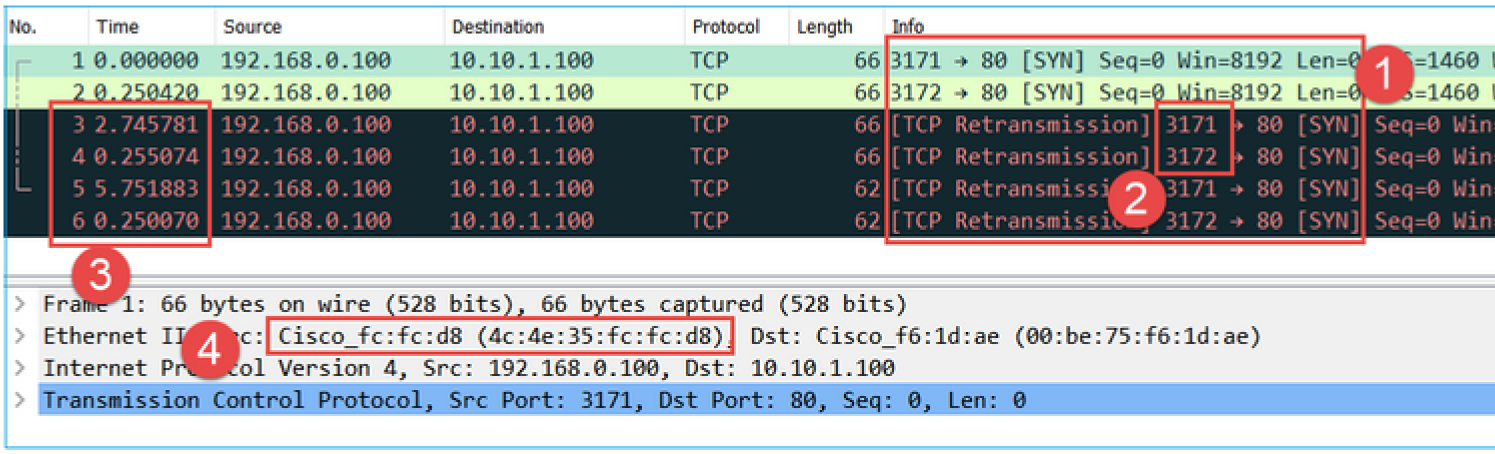
```
firepower#
```

```
show capture CAPO
```

```
0 packet captured
```

```
0 packet shown
```

This is the image of CAPI capture in Wireshark:



```
No.  Time      Source          Destination   Protocol  Length  Info
  1 0.000000  192.168.0.100   10.10.1.100   TCP           66  3171 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460
  2 0.250420  192.168.0.100   10.10.1.100   TCP           66  3172 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460
  3 2.745781  192.168.0.100   10.10.1.100   TCP           66  [TCP Retransmission] 3171 → 80 [SYN] Seq=0 Win
  4 0.255074  192.168.0.100   10.10.1.100   TCP           66  [TCP Retransmission] 3172 → 80 [SYN] Seq=0 Win
  5 5.751883  192.168.0.100   10.10.1.100   TCP           62  [TCP Retransmission] 3171 → 80 [SYN] Seq=0 Win
  6 0.250070  192.168.0.100   10.10.1.100   TCP           62  [TCP Retransmission] 3172 → 80 [SYN] Seq=0 Win

> Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
> Ethernet II, Src: Cisco_fc:fc:d8 (4c:4e:35:fc:fc:d8), Dst: Cisco_f6:1d:ae (00:be:75:f6:1d:ae)
> Internet Protocol Version 4, Src: 192.168.0.100, Dst: 10.10.1.100
> Transmission Control Protocol, Src Port: 3171, Dst Port: 80, Seq: 0, Len: 0
```

Key Points:

1. Only TCP SYN packets are seen (no TCP 3-way handshake).
2. There are 2 TCP sessions (source port 3171 and 3172) that cannot be established. The source client resends the TCP SYN packets. These retransmitted packets are identified by the Wireshark as TCP Retransmissions.
3. The TCP Retransmissions occur every ~3 then 6 etc seconds.
4. The source MAC address is from the correct downstream device.

Based on the 2 captures it can be concluded that:

- A packet of a specific 5-tuple (src/dst IP, src/dst port, protocol) arrives on the firewall on the expected interface (INSIDE).
- A packet does not leave the firewall on the expected interface (OUTSIDE).

Recommended Actions

The actions listed in this section have as a goal to further narrow down the issue.

Action 1. Check the Trace of an Emulated Packet.

Use the packet-tracer tool to see how a packet is supposed to be handled by the firewall. In case the packet is dropped by the firewall Access Policy the trace of the emulated packet looks similar to this output:

<#root>

```
firepower#

packet-tracer input INSIDE tcp 192.168.0.100 11111 10.10.1.100 80


Phase: 1
Type: CAPTURE
Subtype:
Result: ALLOW
Config:
Additional Information:
MAC Access list

Phase: 2
Type: ACCESS-LIST
Subtype:
Result: ALLOW
```

```
Config:
Implicit Rule
Additional Information:
MAC Access list

Phase: 3
Type: ROUTE-LOOKUP
Subtype: Resolve Egress Interface
Result: ALLOW
Config:
Additional Information:
found next-hop 192.168.2.72 using egress ifc  OUTSIDE

Phase: 4
```

Type: ACCESS-LIST

```
Subtype: log
```

Result: DROP

```
Config:
access-group CSM_FW_ACL_ global
access-list CSM_FW_ACL_ advanced deny ip any any rule-id 268439946 event-log flow-start
access-list CSM_FW_ACL_ remark rule-id 268439946: ACCESS POLICY: FTD_Policy - Default
access-list CSM_FW_ACL_ remark rule-id 268439946: L4 RULE: DEFAULT ACTION RULE
Additional Information:

Result:
input-interface: INSIDE
input-status: up
input-line-status: up
output-interface: OUTSIDE
output-status: up
output-line-status: up
Action: drop
```

Drop-reason: (acl-drop) Flow is denied by configured rule, Drop-location: frame 0x00005647a4f4b120 flow

Action 2. Check the traces of live packets.

Enable the packet trace to check how the real TCP SYN packets are handled by the firewall. By default, only the first 50 ingress packets are traced:

<#root>

```
firepower#
```

capture CAPI trace

Clear the capture buffer:

<#root>

```
firepower#
```

`clear capture /all`

In case the packet is dropped by the firewall Access Policy the trace looks similar to this output:

<#root>

```
firepower#
```

`show capture CAPI packet-number 1 trace`

```
6 packets captured

   1: 12:45:36.279740       192.168.0.100.3630 > 10.10.1.100.80: S 2322685377:2322685377(0) win 8192 <ms
Phase: 1
Type: CAPTURE
Subtype:
Result: ALLOW
Config:
Additional Information:
MAC Access list

Phase: 2
Type: ACCESS-LIST
Subtype:
Result: ALLOW
Config:
Implicit Rule
Additional Information:
MAC Access list

Phase: 3
Type: ROUTE-LOOKUP
Subtype: Resolve Egress Interface
Result: ALLOW
Config:
Additional Information:
found next-hop 192.168.2.72 using egress ifc  OUTSIDE

Phase: 4
```

`Type: ACCESS-LIST`

```
Subtype: log
```

`Result: DROP`

```
Config:
access-group CSM_FW_ACL_ global
access-list CSM_FW_ACL_ advanced deny ip any any rule-id 268439946 event-log flow-start
access-list CSM_FW_ACL_ remark rule-id 268439946: ACCESS POLICY: FTD_Policy - Default
access-list CSM_FW_ACL_ remark rule-id 268439946: L4 RULE: DEFAULT ACTION RULE
Additional Information:

Result:
input-interface: INSIDE
input-status: up
input-line-status: up
output-interface: OUTSIDE
```

```
output-status: up
output-line-status: up
Action: drop
```

```
Drop-reason: (acl-drop) Flow is denied by configured rule, Drop-location: frame 0x00005647a4f4b120 flow
```

```
1 packet shown
```

Action 3. Check FTD Lina logs.

To configure Syslog on FTD via FMC check this document:

https://www.cisco.com/c/en/us/support/docs/security/firepower-ngfw/200479-Configure-Logging-on-FTD-via-FMC.html

It is highly recommended to have an external Syslog server configured for FTD Lina logs. If there is no remote Syslog server configured, enable local buffer logs on the firewall while you troubleshoot. The log configuration shown in this example is a good start point:

<#root>

```
firepower#

show run logging

â€¦
logging enable
logging timestamp
logging buffer-size 1000000
logging buffered informational
```

Set the terminal pager to 24 lines in order to control the terminal pager:

<#root>

```
firepower#

terminal pager 24
```

Clear the capture buffer:

<#root>

```
firepower#

clear logging buffer
```

Test the connection and check the logs with a parser filter. In this example the packets are dropped by the firewall Access Policy:

<#root>

```
firepower#
```

```
show logging | include 10.10.1.100
```

```
Oct 09 2019 12:55:51: %FTD-4-106023: Deny tcp src INSIDE:192.168.0.100/3696 dst OUTSIDE:10.10.1.100/80 b
Oct 09 2019 12:55:51: %FTD-4-106023: Deny tcp src INSIDE:192.168.0.100/3697 dst OUTSIDE:10.10.1.100/80 b
Oct 09 2019 12:55:54: %FTD-4-106023: Deny tcp src INSIDE:192.168.0.100/3696 dst OUTSIDE:10.10.1.100/80 b
Oct 09 2019 12:55:54: %FTD-4-106023: Deny tcp src INSIDE:192.168.0.100/3697 dst OUTSIDE:10.10.1.100/80 b
```

Action 4. Check the firewall ASP drops.

If you suspect that the packet is dropped by the firewall you can see the counters of all the packets dropped by the firewall at software level:

<#root>

```
firepower#
```

```
show asp drop
```

```
Frame drop:
  No route to host (no-route)                                                234
  Flow is denied by configured rule (acl-drop)                                71

Last clearing: 07:51:52 UTC Oct 10 2019 by enable_15

Flow drop:

Last clearing: 07:51:52 UTC Oct 10 2019 by enable_15
```

You can enable captures to see all ASP software-level drops:

<#root>

```
firepower#
```

```
capture ASP type asp-drop all buffer 33554432 headers-only
```

> **Tip**: If you are not interested in the packet contents you can capture only the packet headers (headers-only option). This allows you to capture much more many packets in the capture buffer. Additionally, you can increase the size of the capture buffer (by default is 500Kbytes) to a value up 32 Mbytes (buffer option). Finally, as from FTD version 6.3, the file-size option allows you to configure a capture file up to 10GBytes. In that case you can only see the capture contents in a pcap format.

To check the capture contents, you can use a filter to narrow down your search:

<#root>

```
firepower#
```

`show capture ASP | include 10.10.1.100`

```
  18: 07:51:57.823672   192.168.0.100.12410 > 10.10.1.100.80: S 1870382552:1870382552(0) win 8192 <mss 1
  19: 07:51:58.074291   192.168.0.100.12411 > 10.10.1.100.80: S 2006489005:2006489005(0) win 8192 <mss 1
  26: 07:52:00.830370   192.168.0.100.12410 > 10.10.1.100.80: S 1870382552:1870382552(0) win 8192 <mss 1
  29: 07:52:01.080394   192.168.0.100.12411 > 10.10.1.100.80: S 2006489005:2006489005(0) win 8192 <mss 1
  45: 07:52:06.824282   192.168.0.100.12410 > 10.10.1.100.80: S 1870382552:1870382552(0) win 8192 <mss 1
  46: 07:52:07.074230   192.168.0.100.12411 > 10.10.1.100.80: S 2006489005:2006489005(0) win 8192 <mss 1
```

In this case, since the packets are already traced at interface level the reason for the drop is not mentioned in the ASP capture. Remember that a packet can be only traced in one place (ingress interface or ASP drop). In that case, it is recommended to take multiple ASP drops and set a specific ASP drop reason. Here is a recommended approach:

1. Clear the current ASP drop counters:

<#root>

```
firepower#
```

`clear asp drop`

2. Send the flow that you troubleshoot through the firewall (run a test).

3. Check again the ASP drop counters and note down the ones increased.

<#root>

```
firepower#
```

`show asp drop`

```
Frame drop:
  No route to host (
```

`no-route`

```
)                                                 234
  Flow is denied by configured rule (
```

`acl-drop`

```
)                                 71
```

4. Enable ASP capture(s) for the specific drops seen:

<#root>

```
firepower#
```

```
capture ASP_NO_ROUTE type asp-drop no-route

firepower#

capture ASP_ACL_DROP type asp-drop acl-drop
```

5. Send the flow that you troubleshoot through the firewall (run a test).

6. Check the ASP captures. In this case, the packets were dropped due to an absent route:

<#root>

```
firepower#

show capture ASP_NO_ROUTE | include 192.168.0.100.*10.10.1.100

  93: 07:53:52.381663   192.168.0.100.12417 > 10.10.1.100.80: S 3451917925:3451917925(0) win 8192 <mss 1
  95: 07:53:52.632337   192.168.0.100.12418 > 10.10.1.100.80: S 1691844448:1691844448(0) win 8192 <mss 1
 101: 07:53:55.375392   192.168.0.100.12417 > 10.10.1.100.80: S 3451917925:3451917925(0) win 8192 <mss 1
 102: 07:53:55.626386   192.168.0.100.12418 > 10.10.1.100.80: S 1691844448:1691844448(0) win 8192 <mss 1
 116: 07:54:01.376231   192.168.0.100.12417 > 10.10.1.100.80: S 3451917925:3451917925(0) win 8192 <mss 1
 117: 07:54:01.626310   192.168.0.100.12418 > 10.10.1.100.80: S 1691844448:1691844448(0) win 8192 <mss 1
```

Action 5. Check the FTD Lina connection table.

There can be cases where you expect the packet to egress interface 'X', but for whatever reasons it egresses interface 'Y'. The firewall egress interface determination is based on this order of operation:

1. Established Connection Lookup
2. Network Address Translation (NAT) lookup - UN-NAT (destination NAT) phase takes precedence over PBR and route lookup.
3. Policy-Based Routing (PBR)
4. Routing Table lookup

To check the FTD connection table:

<#root>

```
firepower#

show conn

2 in use, 4 most used
Inspect Snort:
        preserve-connection: 2 enabled, 0 in effect, 4 most enabled, 0 most in effect

TCP

DMZ

 10.10.1.100:

80


INSIDE
```

```
 192.168.0.100:
```

11694

```
, idle 0:00:01, bytes 0, flags
```

aA N1

```
TCP
```

DMZ

```
 10.10.1.100:80
```

INSIDE

```
 192.168.0.100:
```

11693

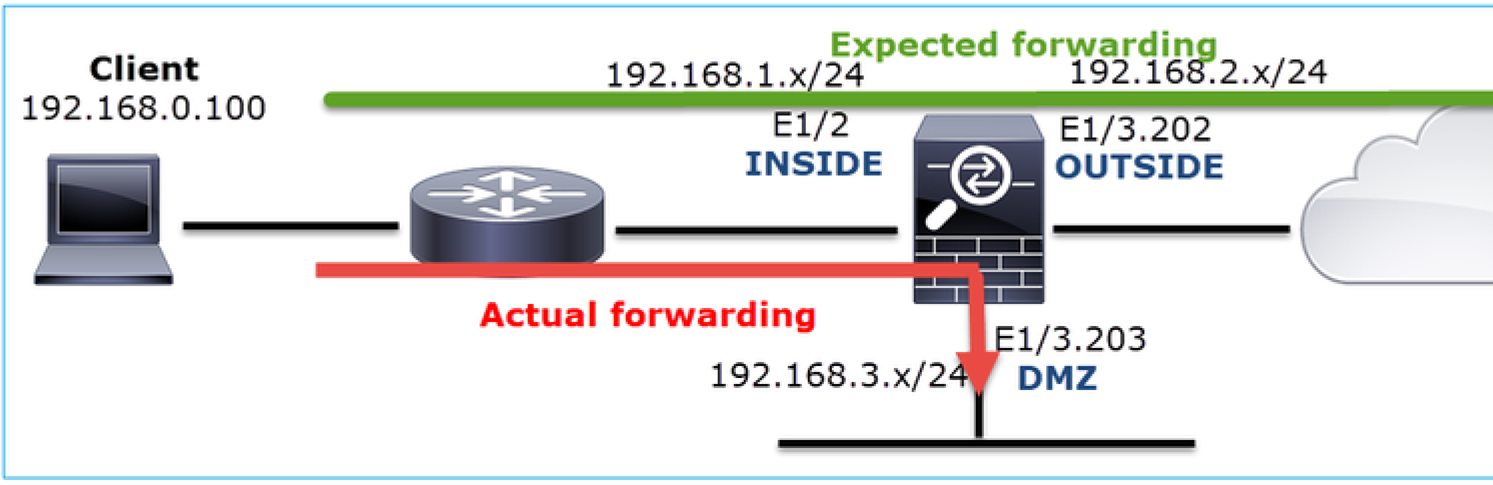
```
, idle 0:00:01, bytes 0, flags
```

aA N1

Key Points:

- Based on the flags (Aa) the connection is embryonic (half-opened - only TCP SYN was seen by the firewall).
- Based on the source/destination ports the ingress interface is INSIDE and the egress interface is DMZ.

This can be visualized in the image here:



> **Note**: Since all FTD interfaces have a Security Level of 0 the interface order in the **show conn** output is based on the interface number. Specifically, the interface with higher vpif-num (virtual platform interface number) is selected as inside while the interface with lower vpif-num is selected as outside. You can see the interface vpif value with the **show interface detail** command. Related enhancement, Cisco bug ID CSCvi15290 ENH: FTD shows the connection directionality in FTD 'show conn' output

---

<#root>

```
firepower#
```

`show interface detail | i Interface number is|Interface [P|E].*is up`

```
...
Interface Ethernet1/2 "INSIDE", is up, line protocol is up
        Interface number is
```

19

```
Interface Ethernet1/3.202 "OUTSIDE", is up, line protocol is up
        Interface number is
```

20

```
Interface Ethernet1/3.203 "DMZ", is up, line protocol is up
        Interface number is
```

22

Note: As from Firepower software release 6.5, ASA release 9.13.x the show conn long and show conn detail command outputs provide information about the connection initiator and responder

---

Output 1:

<#root>

```
firepower#
```

show conn long

```
...
TCP OUTSIDE: 192.168.2.200/80 (192.168.2.200/80) INSIDE: 192.168.1.100/46050 (192.168.1.100/46050), flag
```

Initiator: 192.168.1.100, Responder: 192.168.2.200

```
  Connection lookup keyid: 228982375
```

Output 2:

<#root>

```
firepower#
```

show conn detail

```
...
TCP OUTSIDE: 192.168.2.200/80 INSIDE: 192.168.1.100/46050,
    flags aA N1, idle 4s, uptime 11s, timeout 30s, bytes 0
```

Initiator: 192.168.1.100, Responder: 192.168.2.200

```
  Connection lookup keyid: 228982375
```

Additionally, the **show conn long** displays the NATed IPs within a parenthesis in case of a Network Address Translation:

<#root>

```
firepower#
```

```
show conn long
```

```
...
TCP OUTSIDE: 192.168.2.222/80 (192.168.2.222/80) INSIDE: 192.168.1.100/34792 (192.168.2.150/34792), flag
  Initiator: 192.168.1.100, Responder: 192.168.2.222
  Connection lookup keyid: 262895
```

Action 6. Check the firewall Address Resolution Protocol (ARP) cache.

If the firewall cannot resolve the next hop, the firewall silently drops the original packet (TCP SYN in this case) and continuously sends ARP Requests until it resolves the next hop.

In order to see the firewall ARP cache, use the command:

<#root>

```
firepower#
```

```
show arp
```

Additionally, to check if there are unresolved hosts you can use the command:

<#root>

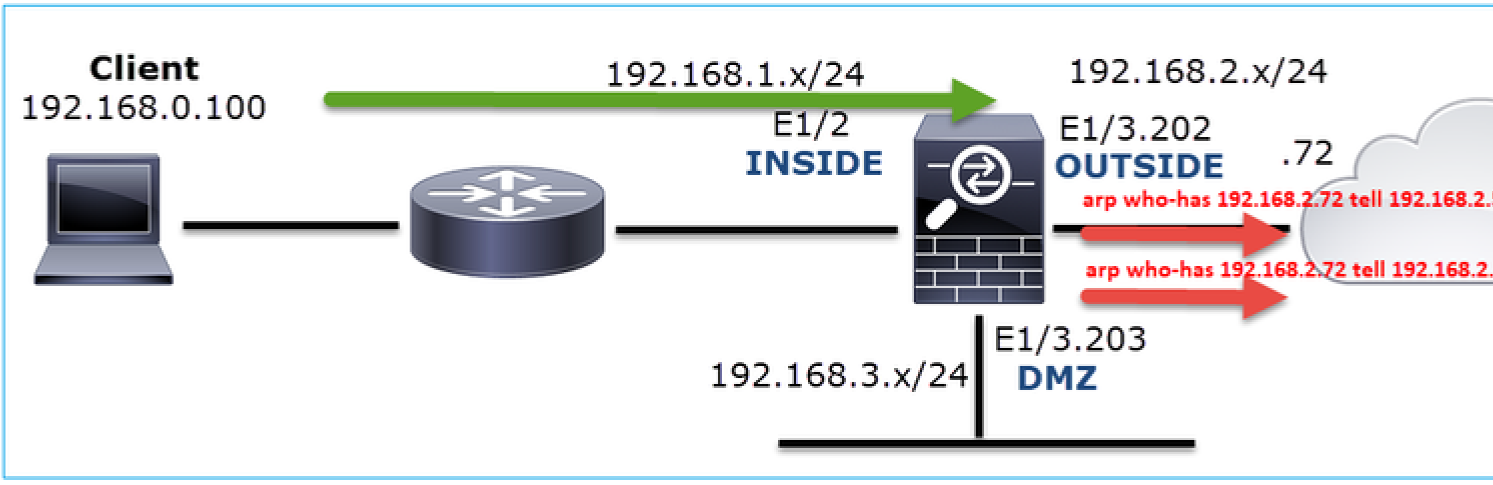
```
firepower#
```

```
 show arp statistics

        Number of ARP entries in ASA: 0

        Dropped blocks in ARP: 84
        Maximum Queued blocks: 3
        Queued blocks: 0
        Interface collision ARPs Received: 0
        ARP-defense Gratuitous ARPS sent: 0
        Total ARP retries:

182           < indicates a possible issue for some hosts

        Unresolved hosts:

1



< this is the current status

        Maximum Unresolved hosts: 2
```

If you want to check further the ARP operation you can enable an ARP-specific capture:

```
<#root>

firepower#

capture ARP ethernet-type arp interface OUTSIDE

firepower#

show capture ARP

...
   4: 07:15:16.877914       802.1Q vlan#202 P0 arp

who-has 192.168.2.72 tell 192.168.2.50

   5: 07:15:18.020033       802.1Q vlan#202 P0 arp who-has 192.168.2.72 tell 192.168.2.50
```

In this output, the firewall (192.168.2.50) tries to resolve the next-hop (192.168.2.72), but there is no ARP reply



The output here shows a functional scenario with proper ARP resolution:

```
<#root>

firepower#

show capture ARP


2 packets captured

   1: 07:17:19.495595       802.1Q vlan#202 P0

arp who-has 192.168.2.72 tell 192.168.2.50

   2: 07:17:19.495946       802.1Q vlan#202 P0

arp reply 192.168.2.72 is-at 4c:4e:35:fc:fc:d8

2 packets shown


<#root>
```

```
firepower#
```

```
show arp
```

```
        INSIDE 192.168.1.71 4c4e.35fc.fcd8 9
        OUTSIDE 192.168.2.72 4c4e.35fc.fcd8 9
```

In case there is no ARP entry in place a trace of a live TCP SYN packet shows:

<#root>

```
firepower#
```

```
show capture CAPI packet-number 1 trace
```

```
6 packets captured

   1: 07:03:43.270585
```

```
192.168.0.100.11997 > 10.10.1.100.80
```

```
: S 4023707145:4023707145(0) win 8192 <mss 1460,nop,wscale 2,nop,nop,sackOK>
Phase: 1
Type: CAPTURE
Subtype:
Result: ALLOW
Config:
Additional Information:
MAC Access list

Phase: 2
Type: ACCESS-LIST
Subtype:
Result: ALLOW
Config:
Implicit Rule
Additional Information:
MAC Access list

Phase: 3
Type: ROUTE-LOOKUP
Subtype: Resolve Egress Interface
Result: ALLOW
Config:
Additional Information:
found next-hop 192.168.2.72 using egress ifc  OUTSIDE
â€¦
Phase: 14
Type: FLOW-CREATION
Subtype:
Result: ALLOW
Config:
Additional Information:
New flow created with id 4814, packet dispatched to next module
â€¦
Phase: 17
Type: ROUTE-LOOKUP
Subtype: Resolve Egress Interface
Result: ALLOW
Config:
```

```
Additional Information:
found next-hop 192.168.2.72 using egress ifc  OUTSIDE

Result:
input-interface: INSIDE
input-status: up
input-line-status: up
```

```
output-interface: OUTSIDE
```

```
output-status: up
output-line-status: up
```

```
Action: allow
```

As can be seen in the output, the trace shows **Action: allow** even when the next hop is not reachable and the packet is silently dropped by the firewall! In this case, the packet-tracer tool must be also checked since it provides a more accurate output:

<#root>

```
firepower#
```

```
packet-tracer input INSIDE tcp 192.168.0.100 1111 10.10.1.100 80
```

```
Phase: 1
Type: CAPTURE
Subtype:
Result: ALLOW
Config:
Additional Information:
MAC Access list

Phase: 2
Type: ACCESS-LIST
Subtype:
Result: ALLOW
Config:
Implicit Rule
Additional Information:
MAC Access list

Phase: 3
Type: ROUTE-LOOKUP
Subtype: Resolve Egress Interface
Result: ALLOW
Config:
Additional Information:
found next-hop 192.168.2.72 using egress ifc  OUTSIDE
â€¦

Phase: 14
Type: FLOW-CREATION
Subtype:
Result: ALLOW
Config:
Additional Information:
New flow created with id 4816, packet dispatched to next module
â€¦
Phase: 17
```

```
Type: ROUTE-LOOKUP
Subtype: Resolve Egress Interface
Result: ALLOW
Config:
Additional Information:
found next-hop 192.168.2.72 using egress ifc  OUTSIDE

Result:
input-interface: INSIDE
input-status: up
input-line-status: up
output-interface: OUTSIDE
output-status: up
output-line-status: up
Action: drop
```

`Drop-reason: (no-v4-adjacency) No valid V4 adjacency, Drop-location: frame 0x00005647a4e86109 flow (NA)/`

In recent ASA/Firepower versions, the previous message has been optimized to:

<#root>

```
Drop-reason: (no-v4-adjacency) No valid V4 adjacency.
```

`Check ARP table (show arp) has entry for nexthop`

```
., Drop-location: f
```

Possible Causes and Recommended Actions Summary

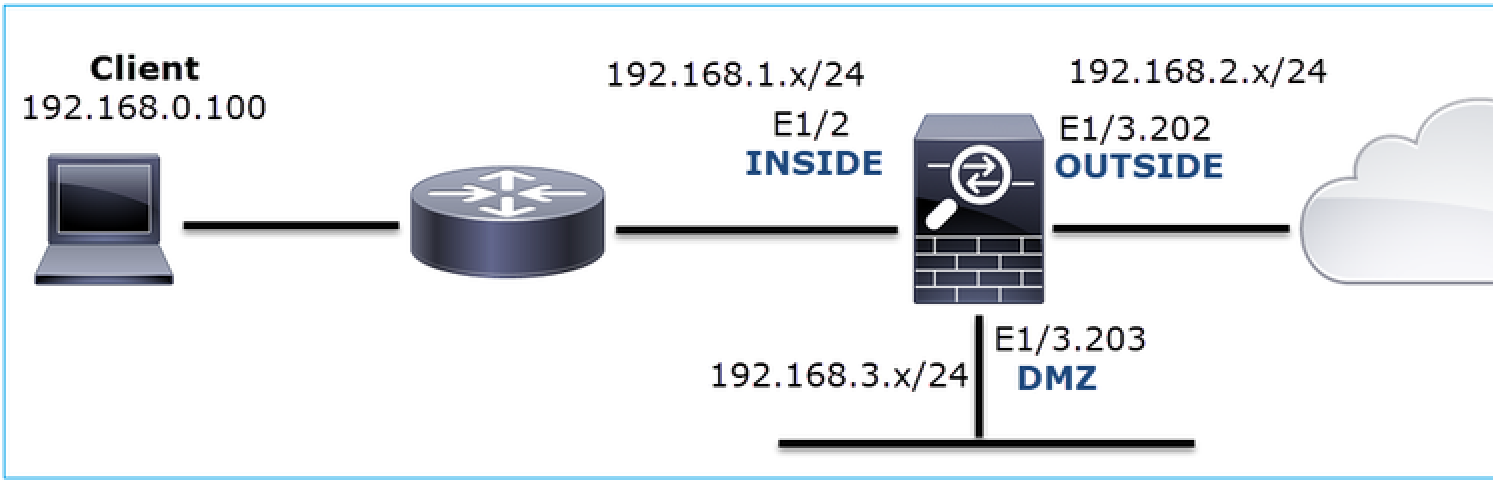
If you only see a TCP SYN packet on the ingress interfaces, but no TCP SYN packet sent out of the expected egress interface some possible causes are:

Possible Cause	**Recommended Actions**
The packet is dropped by the firewall access-policy.	• Use **packet-tracer** or **capture w/trace** to see how to firewall handles the packet. • Check the firewall logs. • Check the firewall ASP drops **(show asp drop** or **capture type asp-drop).** • Check FMC Connection Events. This assumes that the rule has logging enabled.
The capture filter is wrong.	• Use **packet-tracer** or **capture w/trace** to see if there is NAT translation that modifies the source or destination IP. In that case, adjust your capture filter. • **show conn long** command output shows the NATed IPs.

The packet is sent to a different egress interface.	• Use **packet-tracer** or **capture w/trace** to see how the firewall handles the packet. Remember the order of operations which regard the egress interface determination, current connection, UN-NAT, PBR and Routing table lookup. • Check the firewall logs. • Check the firewall connection table (**show conn**). If the packet is sent to a wrong interface because it matches a current connection use the command **clear conn address** and specify the 5-tuple of the connection that you want to clear.
There is no route towards the destination.	• Use **packet-tracer** or **capture w/trace** to see how to firewall handles the packet. • Check the firewall ASP drops **(show asp drop)** for **no-route** drop reason.
There is no ARP entry on the egress interface.	• Check the firewall ARP cache (**show arp**). • Use **packet-tracer** to see if there is a valid adjacency.
The egress interface is down.	Check the output of the **show interface ip brief** command on the firewall and verify the interface status.

Case 2. TCP SYN from Client, TCP RST from Server

This image shows the topology:



Problem Description: HTTP does not work

Affected Flow:

Src IP: 192.168.0.100

Dst IP: 10.10.1.100

Protocol: TCP 80

Capture Analysis

Enable captures on the FTD LINA engine.

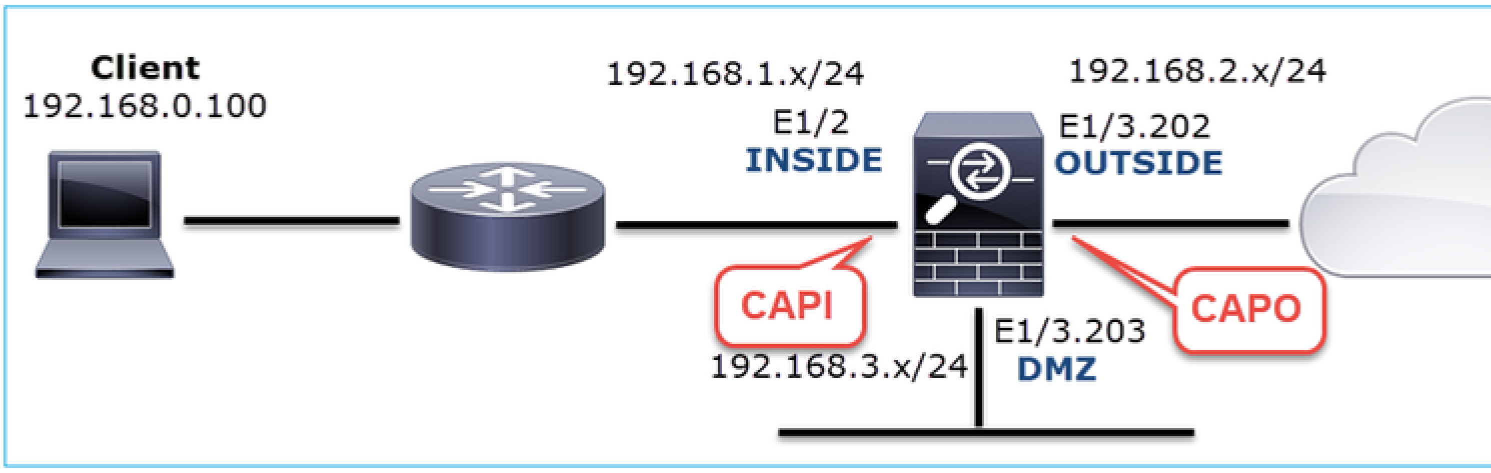
<#root>

```
firepower#
```

```
capture CAPI int INSIDE match ip host 192.168.0.100 host 10.10.1.100
```

```
firepower#
```

```
capture CAPO int OUTSIDE match ip host 192.168.0.100 host 10.10.1.100
```



Captures - Non-functional scenario:

From the device CLI the captures look as follows:

<#root>

```
firepower#
```

```
show capture
```

```
capture CAPI type raw-data trace interface INSIDE [Capturing -
```

```
834 bytes
```

```
]
  match ip host 192.168.0.100 host 10.10.1.100
capture CAPO type raw-data interface OUTSIDE [Capturing -
```

```
878 bytes
```

```
]
  match ip host 192.168.0.100 host 10.10.1.100
```

CAPI contents:

<#root>

```
firepower#
```

```
show capture CAPI
```

```
   1: 05:20:36.654217   192.168.0.100.22195 > 10.10.1.100.80:
```

```
S
```

```
 1397289928:1397289928(0) win 8192 <mss 1460,nop,wscale 2,nop,nop,sackOK>
   2: 05:20:36.904311   192.168.0.100.22196 > 10.10.1.100.80:
```

```
S
```

```
 2171673258:2171673258(0) win 8192 <mss 1460,nop,wscale 2,nop,nop,sackOK>
   3: 05:20:36.905043   10.10.1.100.80 > 192.168.0.100.22196:
```

```
R
```

```
 1850052503:1850052503(0) ack 2171673259 win 0
   4: 05:20:37.414132   192.168.0.100.22196 > 10.10.1.100.80:
```

```
S
```

```
 2171673258:2171673258(0) win 8192 <mss 1460,nop,wscale 2,nop,nop,sackOK>
   5: 05:20:37.414803   10.10.1.100.80 > 192.168.0.100.22196:
```

```
R
```

```
 31997177:31997177(0) ack 2171673259 win 0
   6: 05:20:37.914183   192.168.0.100.22196 > 10.10.1.100.80:
```

```
S
```

```
 2171673258:2171673258(0) win 8192 <mss 1460,nop,nop,sackOK>
...
```

CAPO contents:

<#root>

```
firepower#
```

```
show capture CAPO
```

```
   1: 05:20:36.654507   802.1Q vlan#202 P0 192.168.0.100.22195 > 10.10.1.100.80:
```

```
S
```

```
 2866789268:2866789268(0) win 8192 <mss 1380,nop,wscale 2,nop,nop,sackOK>
   2: 05:20:36.904478   802.1Q vlan#202 P0 192.168.0.100.22196 > 10.10.1.100.80:
```

```
S
```

```
 4785344:4785344(0) win 8192 <mss 1380,nop,wscale 2,nop,nop,sackOK>
   3: 05:20:36.904997   802.1Q vlan#202 P0 10.10.1.100.80 > 192.168.0.100.22196:
```

```
R
```

```
 0:0(0) ack 4785345 win 0
   4: 05:20:37.414269   802.1Q vlan#202 P0 192.168.0.100.22196 > 10.10.1.100.80:
```

S

```
 4235354730:4235354730(0) win 8192 <mss 1380,nop,wscale 2,nop,nop,sackOK>
   5: 05:20:37.414758   802.1Q vlan#202 P0 10.10.1.100.80 > 192.168.0.100.22196:
```

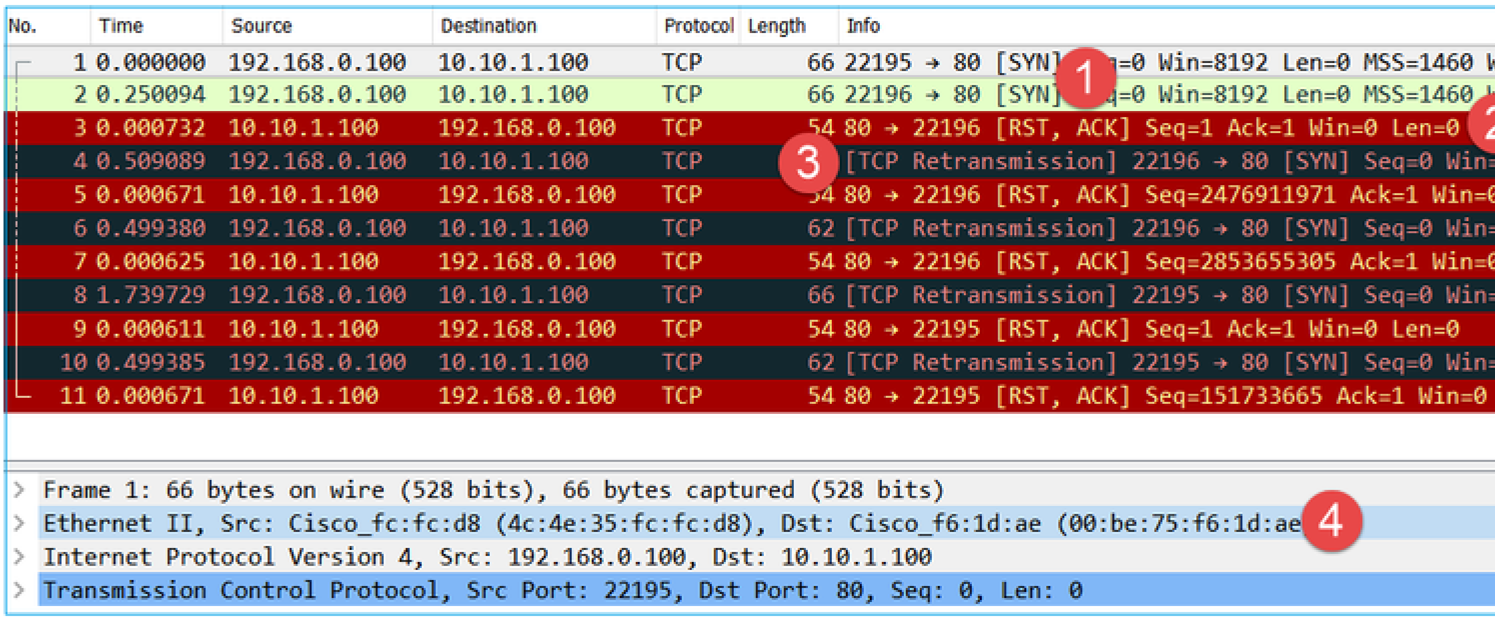
R

```
 0:0(0) ack 4235354731 win 0
   6: 05:20:37.914305   802.1Q vlan#202 P0 192.168.0.100.22196 > 10.10.1.100.80:
```

S

```
 4118617832:4118617832(0) win 8192 <mss 1380,nop,nop,sackOK>
```

This image shows the capture of CAPI in Wireshark.

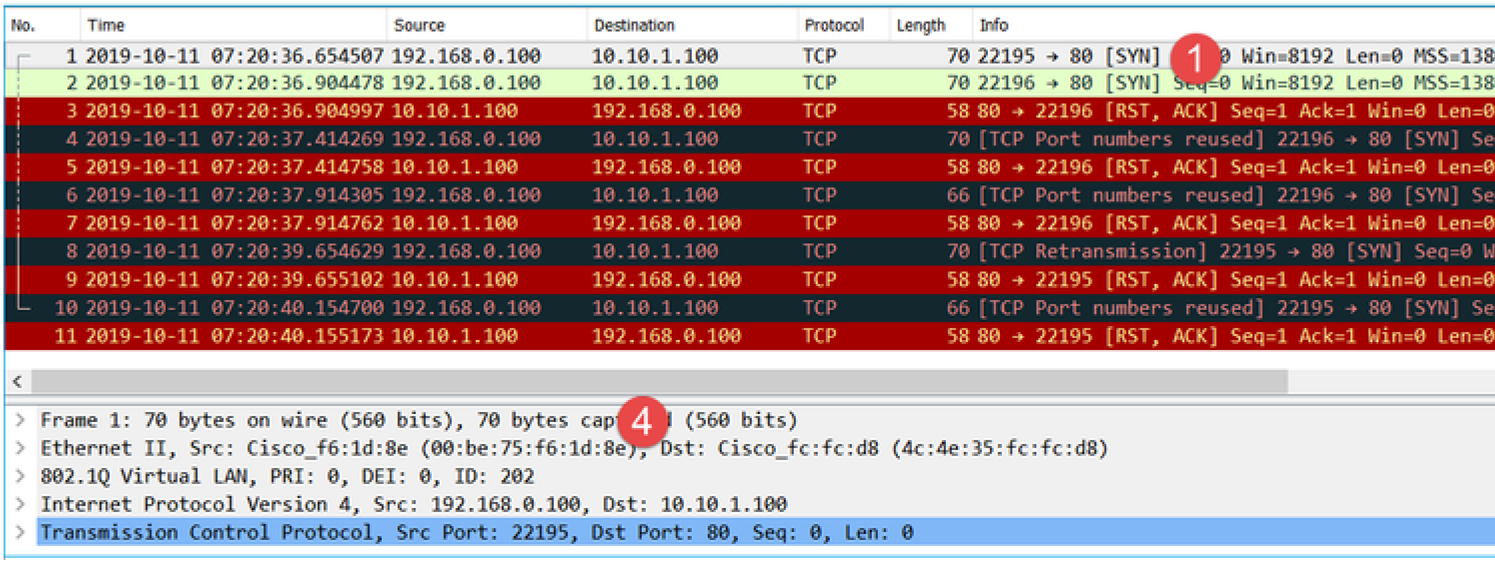


```
No.  Time      Source         Destination    Protocol Length Info
  1 0.000000  192.168.0.100  10.10.1.100    TCP       66 22195 → 80 [SYN] q=0 Win=8192 Len=0 MSS=1460 W
  2 0.250094  192.168.0.100  10.10.1.100    TCP       66 22196 → 80 [SYN] q=0 Win=8192 Len=0 MSS=1460 W
  3 0.000732  10.10.1.100    192.168.0.100  TCP       54 80 → 22196 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
  4 0.509089  192.168.0.100  10.10.1.100    TCP          [TCP Retransmission] 22196 → 80 [SYN] Seq=0 Win=
  5 0.000671  10.10.1.100    192.168.0.100  TCP       4 80 → 22196 [RST, ACK] Seq=2476911971 Ack=1 Win=0
  6 0.499380  192.168.0.100  10.10.1.100    TCP       62 [TCP Retransmission] 22196 → 80 [SYN] Seq=0 Win=
  7 0.000625  10.10.1.100    192.168.0.100  TCP       54 80 → 22196 [RST, ACK] Seq=2853655305 Ack=1 Win=0
  8 1.739729  192.168.0.100  10.10.1.100    TCP       66 [TCP Retransmission] 22195 → 80 [SYN] Seq=0 Win=
  9 0.000611  10.10.1.100    192.168.0.100  TCP       54 80 → 22195 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
 10 0.499385  192.168.0.100  10.10.1.100    TCP       62 [TCP Retransmission] 22195 → 80 [SYN] Seq=0 Win=
 11 0.000671  10.10.1.100    192.168.0.100  TCP       54 80 → 22195 [RST, ACK] Seq=151733665 Ack=1 Win=0

> Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
> Ethernet II, Src: Cisco_fc:fc:d8 (4c:4e:35:fc:fc:d8), Dst: Cisco_f6:1d:ae (00:be:75:f6:1d:ae
> Internet Protocol Version 4, Src: 192.168.0.100, Dst: 10.10.1.100
> Transmission Control Protocol, Src Port: 22195, Dst Port: 80, Seq: 0, Len: 0
```

Key Points:

1. The source sends a TCP SYN packet.
2. A TCP RST is sent towards the source.
3. The source retransmits the TCP SYN packets.
4. The MAC addresses are correct (on ingress packets the source MAC address belongs to the downstream router, the destination MAC address belongs to the firewall INSIDE interface).

This image shows the capture of CAPO in Wireshark:



```
No.  Time                        Source         Destination   Protocol  Length  Info
  1 2019-10-11 07:20:36.654507 192.168.0.100 10.10.1.100   TCP       70 22195 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=138
  2 2019-10-11 07:20:36.904478 192.168.0.100 10.10.1.100   TCP       70 22196 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=138
  3 2019-10-11 07:20:36.904997 10.10.1.100   192.168.0.100 TCP       58 80 → 22196 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
  4 2019-10-11 07:20:37.414269 192.168.0.100 10.10.1.100   TCP       70 [TCP Port numbers reused] 22196 → 80 [SYN] Se
  5 2019-10-11 07:20:37.414758 10.10.1.100   192.168.0.100 TCP       58 80 → 22196 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
  6 2019-10-11 07:20:37.914305 192.168.0.100 10.10.1.100   TCP       66 [TCP Port numbers reused] 22196 → 80 [SYN] Se
  7 2019-10-11 07:20:37.914762 10.10.1.100   192.168.0.100 TCP       58 80 → 22196 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
  8 2019-10-11 07:20:39.654629 192.168.0.100 10.10.1.100   TCP       70 [TCP Retransmission] 22195 → 80 [SYN] Seq=0 W
  9 2019-10-11 07:20:39.655102 10.10.1.100   192.168.0.100 TCP       58 80 → 22195 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
 10 2019-10-11 07:20:40.154700 192.168.0.100 10.10.1.100   TCP       66 [TCP Port numbers reused] 22195 → 80 [SYN] Se
 11 2019-10-11 07:20:40.155173 10.10.1.100   192.168.0.100 TCP       58 80 → 22195 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

> Frame 1: 70 bytes on wire (560 bits), 70 bytes captured (560 bits)
> Ethernet II, Src: Cisco_f6:1d:8e (00:be:75:f6:1d:8e), Dst: Cisco_fc:fc:d8 (4c:4e:35:fc:fc:d8)
> 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 202
> Internet Protocol Version 4, Src: 192.168.0.100, Dst: 10.10.1.100
> Transmission Control Protocol, Src Port: 22195, Dst Port: 80, Seq: 0, Len: 0
```

Key Points:

1. The source sends a TCP SYN packet.
2. A TCP RST arrives on the OUTSIDE interface.
3. The source retransmits the TCP SYN packets.
4. The MAC addresses are correct (on egress packets the firewall OUTSIDE is the source MAC, upstream router is the destination MAC).

Based on the 2 captures it can be concluded that:

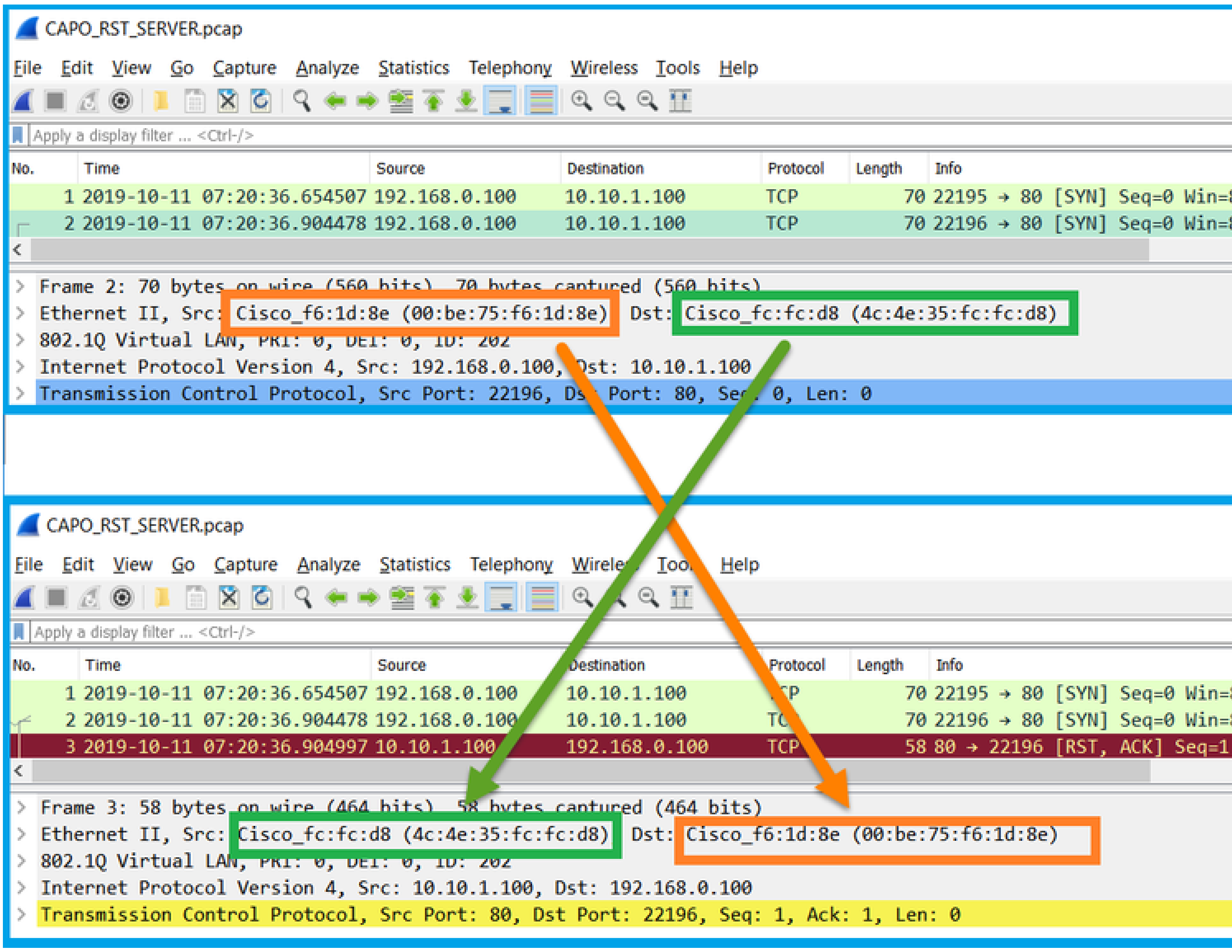
- The TCP 3-way handshake between the client and the server does not get completed
- There is a TCP RST which arrives on the firewall egress interface
- The firewall 'talks' to the proper upstream and downstream devices (based on the MAC addresses)

Recommended Actions

The actions listed in this section have as a goal to further narrow down the issue.

Action 1. Check the source MAC address that sends the TCP RST.

Verify that the destination MAC seen in the TCP SYN packet is the same as the source MAC has seen in the TCP RST packet.


CAPO_RST_SERVER.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.   Time                        Source          Destination    Protocol  Length  Info
    1 2019-10-11 07:20:36.654507 192.168.0.100   10.10.1.100    TCP           70 22195 → 80 [SYN] Seq=0 Win=
    2 2019-10-11 07:20:36.904478 192.168.0.100   10.10.1.100    TCP           70 22196 → 80 [SYN] Seq=0 Win=

> Frame 2: 70 bytes on wire (560 bits), 70 bytes captured (560 bits)
> Ethernet II, Src: Cisco_f6:1d:8e (00:be:75:f6:1d:8e), Dst: Cisco_fc:fc:d8 (4c:4e:35:fc:fc:d8)
> 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 202
> Internet Protocol Version 4, Src: 192.168.0.100, Dst: 10.10.1.100
> Transmission Control Protocol, Src Port: 22196, Dst Port: 80, Seq: 0, Len: 0
```

CAPO_RST_SERVER.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

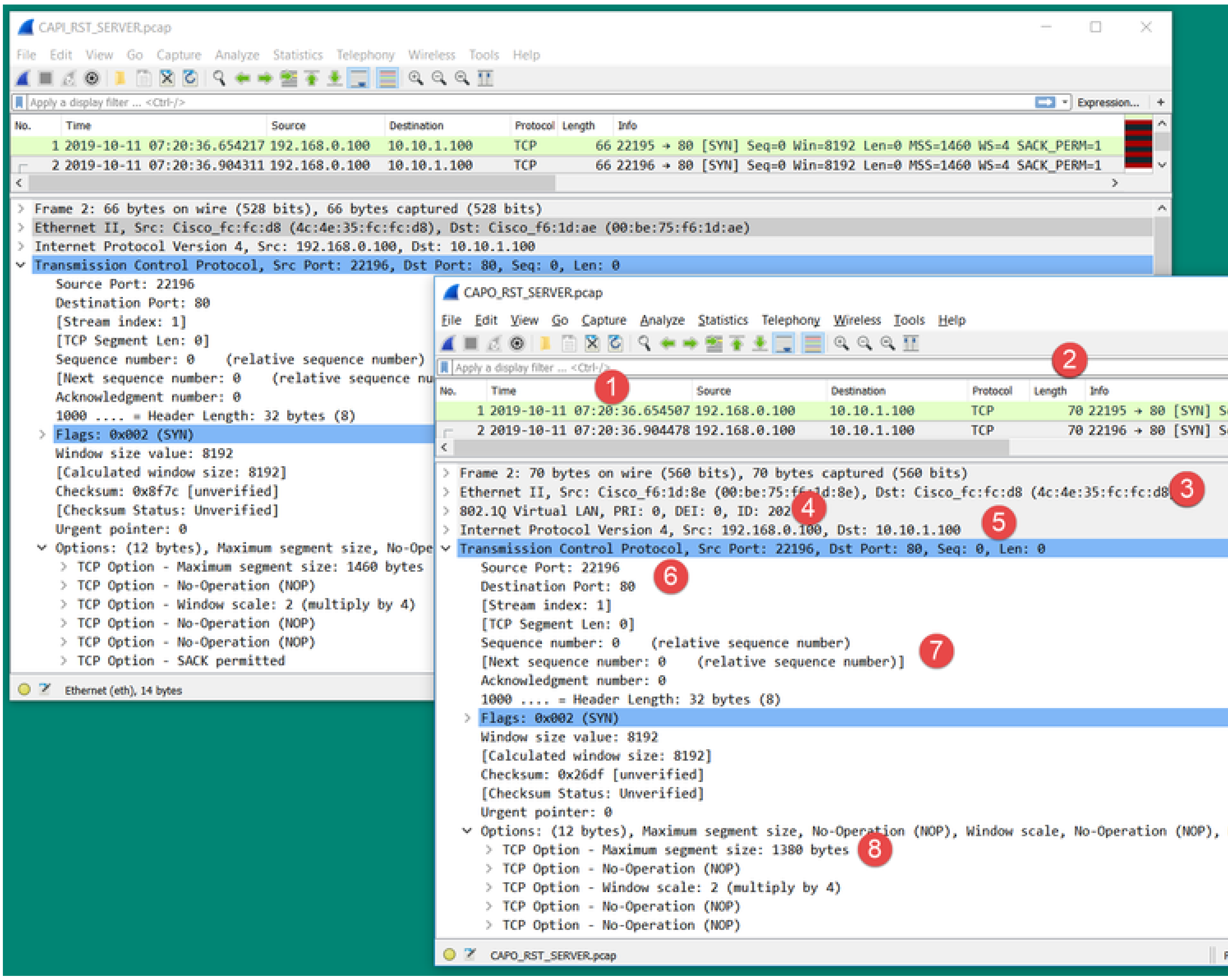
```
No.   Time                        Source          Destination    Protocol  Length  Info
    1 2019-10-11 07:20:36.654507 192.168.0.100   10.10.1.100    TCP           70 22195 → 80 [SYN] Seq=0 Win=
    2 2019-10-11 07:20:36.904478 192.168.0.100   10.10.1.100    TCP           70 22196 → 80 [SYN] Seq=0 Win=
    3 2019-10-11 07:20:36.904997 10.10.1.100     192.168.0.100  TCP           58 80 → 22196 [RST, ACK] Seq=1

> Frame 3: 58 bytes on wire (464 bits), 58 bytes captured (464 bits)
> Ethernet II, Src: Cisco_fc:fc:d8 (4c:4e:35:fc:fc:d8), Dst: Cisco_f6:1d:8e (00:be:75:f6:1d:8e)
> 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 202
> Internet Protocol Version 4, Src: 10.10.1.100, Dst: 192.168.0.100
> Transmission Control Protocol, Src Port: 80, Dst Port: 22196, Seq: 1, Ack: 1, Len: 0
```

This check has as a goal to confirm 2 things:

- Verify that there is no asymmetric flow.
- Verify that the MAC belongs to the expected upstream device.

Action 2. Compare ingress and egress packets.

Visually compare the 2 packets on Wireshark to verify that the firewall does not modify/corrupt the packets. Some expected differences are highlighted.


CAPI_RST_SERVER.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>
Expression...
CAPO_RST_SERVER.pcap
1
2
3
4
5
6
7
8

```
No.  Time                        Source         Destination  Protocol Length Info
  1 2019-10-11 07:20:36.654217 192.168.0.100  10.10.1.100  TCP      66 22195 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
  2 2019-10-11 07:20:36.904311 192.168.0.100  10.10.1.100  TCP      66 22196 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1

> Frame 2: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
> Ethernet II, Src: Cisco_fc:fc:d8 (4c:4e:35:fc:fc:d8), Dst: Cisco_f6:1d:ae (00:be:75:f6:1d:ae)
> Internet Protocol Version 4, Src: 192.168.0.100, Dst: 10.10.1.100
v Transmission Control Protocol, Src Port: 22196, Dst Port: 80, Seq: 0, Len: 0
    Source Port: 22196
    Destination Port: 80
    [Stream index: 1]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    [Next sequence number: 0    (relative sequence nu
    Acknowledgment number: 0
    1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x002 (SYN)
    Window size value: 8192
    [Calculated window size: 8192]
    Checksum: 0x8f7c [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  v Options: (12 bytes), Maximum segment size, No-Ope
    > TCP Option - Maximum segment size: 1460 bytes
    > TCP Option - No-Operation (NOP)
    > TCP Option - Window scale: 2 (multiply by 4)
    > TCP Option - No-Operation (NOP)
    > TCP Option - No-Operation (NOP)
    > TCP Option - SACK permitted
Ethernet (eth), 14 bytes
```

```
No.  Time                        Source         Destination  Protocol Length Info
  1 2019-10-11 07:20:36.654507 192.168.0.100  10.10.1.100  TCP      70 22195 → 80 [SYN] S
  2 2019-10-11 07:20:36.904478 192.168.0.100  10.10.1.100  TCP      70 22196 → 80 [SYN] S

> Frame 2: 70 bytes on wire (560 bits), 70 bytes captured (560 bits)
> Ethernet II, Src: Cisco_f6:1d:8e (00:be:75:f6:1d:8e), Dst: Cisco_fc:fc:d8 (4c:4e:35:fc:fc:d8)
> 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 202
> Internet Protocol Version 4, Src: 192.168.0.100, Dst: 10.10.1.100
v Transmission Control Protocol, Src Port: 22196, Dst Port: 80, Seq: 0, Len: 0
    Source Port: 22196
    Destination Port: 80
    [Stream index: 1]
    [TCP Segment Len: 0]
    Sequence number: 0    (relative sequence number)
    [Next sequence number: 0    (relative sequence number)]
    Acknowledgment number: 0
    1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x002 (SYN)
    Window size value: 8192
    [Calculated window size: 8192]
    Checksum: 0x26df [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  v Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP),
    > TCP Option - Maximum segment size: 1380 bytes
    > TCP Option - No-Operation (NOP)
    > TCP Option - Window scale: 2 (multiply by 4)
    > TCP Option - No-Operation (NOP)
    > TCP Option - No-Operation (NOP)
CAPO_RST_SERVER.pcap
```

Key Points:

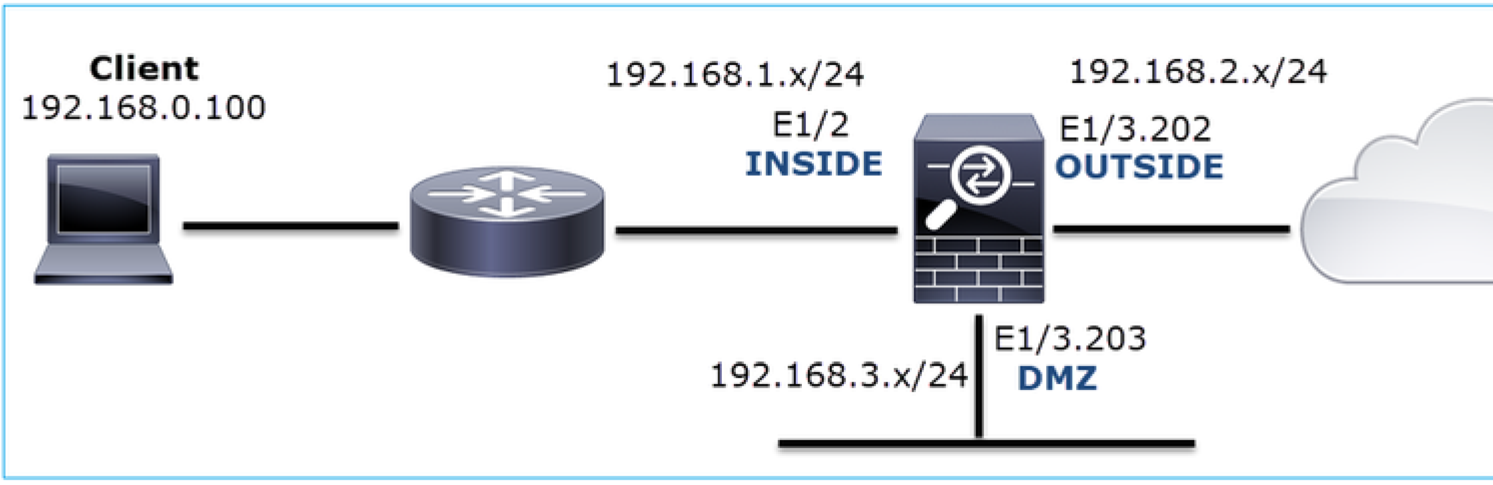
1. Timestamps are different. On the other hand, the difference must be small and reasonable. This depends on the features and policy checks applied to the packet as well as the load on the device.
2. The length of the packets may differ especially if there is a dot1Q header added/removed by the firewall on one side only.
3. The MAC addresses are different.
4. A dot1Q header can be in place if the capture was taken on a subinterface.
5. The IP address(es) are different in case NAT or Port Address Translation (PAT) is applied to the packet.
6. The source or destination ports are different in case NAT or PAT is applied to the packet.
7. If you disable the Wireshark **Relative Sequence Number** option you see that the TCP sequence numbers/acknowledgment numbers are modified by the firewall due to Initial Sequence Number (ISN) randomization.
8. Some TCP options can be overwritten. For example, the firewall by default changes the TCP Maximum Segment Size (MSS) to 1380 in order to avoid packet fragmentation in the transit path.

Action 3. Take a capture at the destination.

If possible, take a capture at the destination itself. If this is not possible take a capture as close to the destination as possible. The goal here is to verify who sends the TCP RST (is the destination server or is some other device in the path?).

Case 3. TCP 3-Way Handshake + RST from One Endpoint

This image shows the topology:



Problem Description: HTTP does not work

Affected Flow:

Src IP: 192.168.0.100

Dst IP: 10.10.1.100

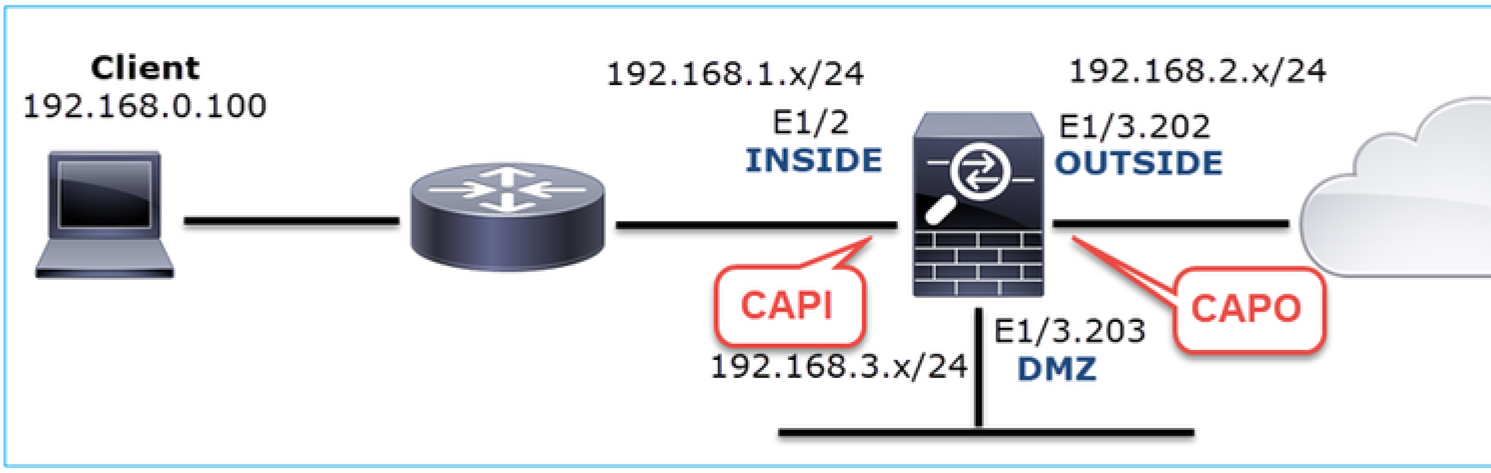
Protocol: TCP 80

Capture Analysis

Enable captures on the FTD LINA engine.

<#root>

```
firepower#
 capture CAPI int INSIDE match ip host 192.168.0.100 host 10.10.1.100
firepower#
 capture CAPO int OUTSIDE match ip host 192.168.0.100 host 10.10.1.100
```

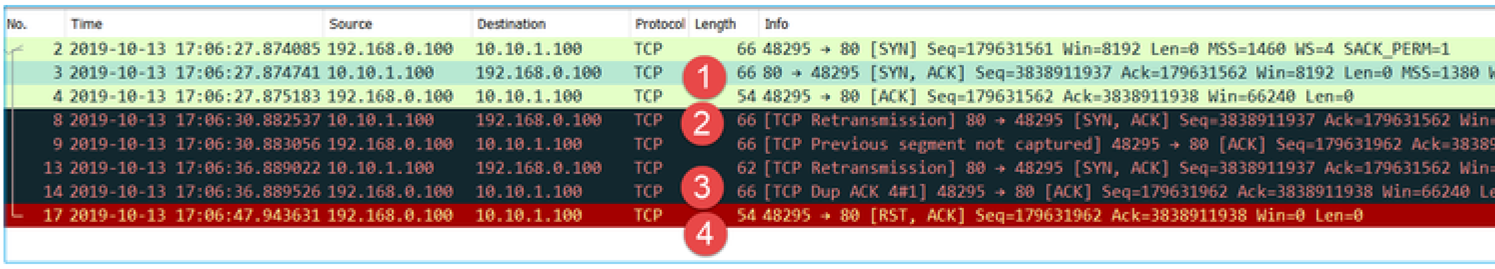


Captures - Non-functional scenario:

There are a couple of different ways this issue can manifest in captures.

3.1 - TCP 3-way Handshake + Delayed RST from the Client

Both the firewall captures CAPI and CAPO contain the same packets, as shown in the image.



```
No.  Time                       Source          Destination     Protocol Length Info
   2 2019-10-13 17:06:27.874085 192.168.0.100   10.10.1.100     TCP      66 48295 → 80 [SYN] Seq=179631561 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
   3 2019-10-13 17:06:27.874741 10.10.1.100     192.168.0.100   TCP      66 80 → 48295 [SYN, ACK] Seq=3838911937 Ack=179631562 Win=8192 Len=0 MSS=1380 W
   4 2019-10-13 17:06:27.875183 192.168.0.100   10.10.1.100     TCP      54 48295 → 80 [ACK] Seq=179631562 Ack=3838911938 Win=66240 Len=0
   8 2019-10-13 17:06:30.882537 10.10.1.100     192.168.0.100   TCP      66 [TCP Retransmission] 80 → 48295 [SYN, ACK] Seq=3838911937 Ack=179631562 Win=
   9 2019-10-13 17:06:30.883056 192.168.0.100   10.10.1.100     TCP      66 [TCP Previous segment not captured] 48295 → 80 [ACK] Seq=179631962 Ack=38389
  13 2019-10-13 17:06:36.889022 10.10.1.100     192.168.0.100   TCP      62 [TCP Retransmission] 80 → 48295 [SYN, ACK] Seq=3838911937 Ack=179631562 Win=
  14 2019-10-13 17:06:36.889526 192.168.0.100   10.10.1.100     TCP      66 [TCP Dup ACK 4#1] 48295 → 80 [ACK] Seq=179631962 Ack=3838911938 Win=66240 Le
  17 2019-10-13 17:06:47.943631 192.168.0.100   10.10.1.100     TCP      54 48295 → 80 [RST, ACK] Seq=179631962 Ack=3838911938 Win=0 Len=0
```

Key Points:

1. The TCP 3-way handshake goes through the firewall.
2. The server retransmits the SYN/ACK.
3. The client retransmits the ACK.
4. After ~20 sec the client gives up and sends a TCP RST.

Recommended Actions

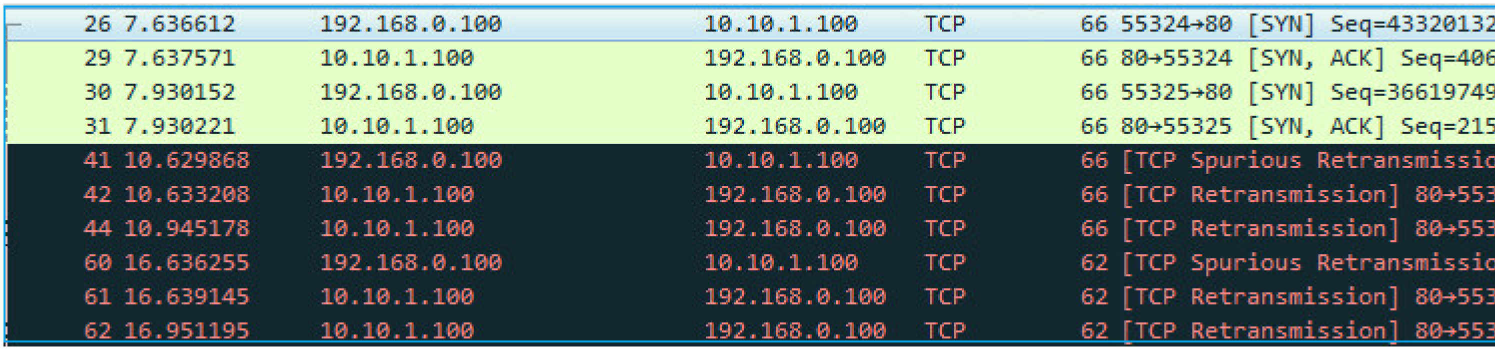
The actions listed in this section have as a goal to further narrow down the issue.

Action 1. Take captures as close to the two endpoints as possible.

The firewall captures indicate that the client ACK was not processed by the server. This is based on these facts:

- The server retransmits the SYN/ACK.
- The client retransmits the ACK.
- The client sends a TCP RST or FIN/ACK before any data.

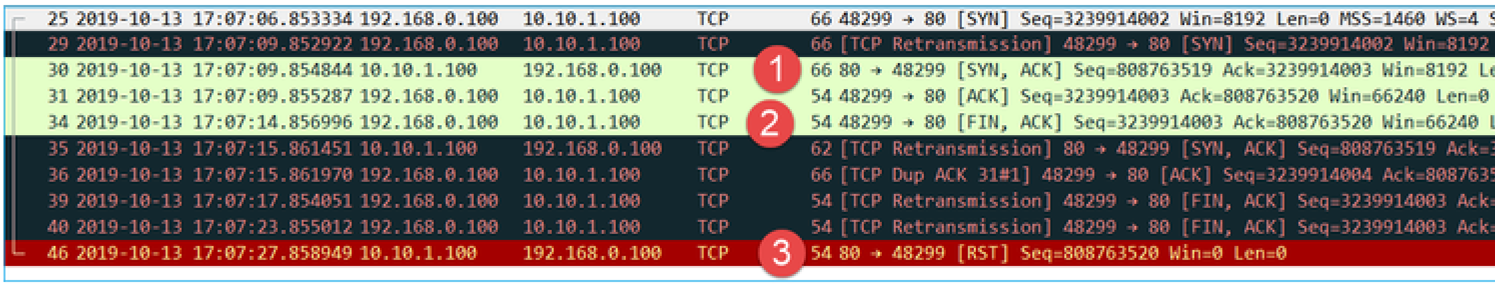
Capture on the server shows the problem. The client ACK from the TCP 3-way handshake never arrived:



```
26 7.636612   192.168.0.100   10.10.1.100     TCP   66 55324→80 [SYN] Seq=43320132
29 7.637571   10.10.1.100     192.168.0.100   TCP   66 80→55324 [SYN, ACK] Seq=406
30 7.930152   192.168.0.100   10.10.1.100     TCP   66 55325→80 [SYN] Seq=36619749
31 7.930221   10.10.1.100     192.168.0.100   TCP   66 80→55325 [SYN, ACK] Seq=215
41 10.629868  192.168.0.100   10.10.1.100     TCP   66 [TCP Spurious Retransmissio
42 10.633208  10.10.1.100     192.168.0.100   TCP   66 [TCP Retransmission] 80→553
44 10.945178  10.10.1.100     192.168.0.100   TCP   66 [TCP Retransmission] 80→553
60 16.636255  192.168.0.100   10.10.1.100     TCP   62 [TCP Spurious Retransmissio
61 16.639145  10.10.1.100     192.168.0.100   TCP   62 [TCP Retransmission] 80→553
62 16.951195  10.10.1.100     192.168.0.100   TCP   62 [TCP Retransmission] 80→553
```

3.2 - TCP 3-way Handshake + Delayed FIN/ACK from Client + Delayed RST from the Server

Both the firewall captures CAPI and CAPO contain the same packets, as shown in the image.



```
25 2019-10-13 17:07:06.853334 192.168.0.100  10.10.1.100    TCP     66 48299 → 80 [SYN] Seq=3239914002 Win=8192 Len=0 MSS=1460 WS=4 S
29 2019-10-13 17:07:09.852922 192.168.0.100  10.10.1.100    TCP     66 [TCP Retransmission] 48299 → 80 [SYN] Seq=3239914002 Win=8192
30 2019-10-13 17:07:09.854844 10.10.1.100    192.168.0.100  TCP  1  66 80 → 48299 [SYN, ACK] Seq=808763519 Ack=3239914003 Win=8192 Le
31 2019-10-13 17:07:09.855287 192.168.0.100  10.10.1.100    TCP     54 48299 → 80 [ACK] Seq=3239914003 Ack=808763520 Win=66240 Len=0
34 2019-10-13 17:07:14.856996 192.168.0.100  10.10.1.100    TCP  2  54 48299 → 80 [FIN, ACK] Seq=3239914003 Ack=808763520 Win=66240 L
35 2019-10-13 17:07:15.861451 10.10.1.100    192.168.0.100  TCP     62 [TCP Retransmission] 80 → 48299 [SYN, ACK] Seq=808763519 Ack=3
36 2019-10-13 17:07:15.861970 192.168.0.100  10.10.1.100    TCP     66 [TCP Dup ACK 31#1] 48299 → 80 [ACK] Seq=3239914004 Ack=8087635
39 2019-10-13 17:07:17.854051 192.168.0.100  10.10.1.100    TCP     54 [TCP Retransmission] 48299 → 80 [FIN, ACK] Seq=3239914003 Ack=
40 2019-10-13 17:07:23.855012 192.168.0.100  10.10.1.100    TCP     54 [TCP Retransmission] 48299 → 80 [FIN, ACK] Seq=3239914003 Ack=
46 2019-10-13 17:07:27.858949 10.10.1.100    192.168.0.100  TCP  3  54 80 → 48299 [RST] Seq=808763520 Win=0 Len=0
```

Key Points:

1. The TCP 3-way handshake goes through the firewall.
2. After ~5 sec the client sends a FIN/ACK.
3. After ~20 sec the server gives up and sends a TCP RST.

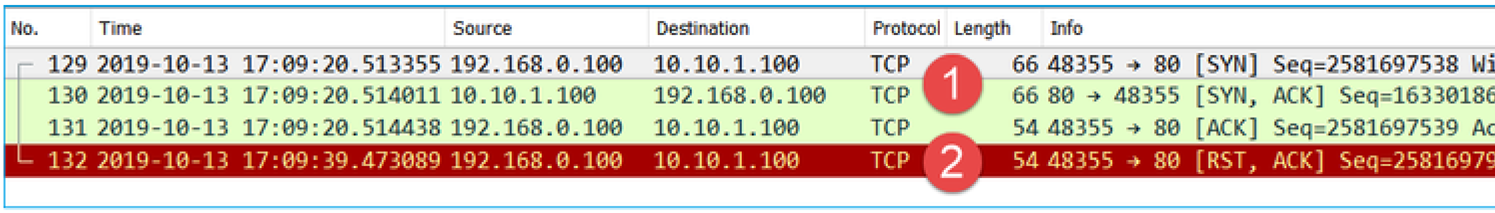
Based on this capture it can be concluded that although there is a TCP 3-way handshake through the firewall it seems that it never actually gets completed on one endpoint (the retransmissions indicate this).

Recommended Actions

Same as in case 3.1

3.3 - TCP 3-way Handshake + Delayed RST from the Client

Both the firewall captures CAPI and CAPO contain the same packets, as shown in the image.



```
No.   Time                        Source          Destination     Protocol  Length  Info
129 2019-10-13 17:09:20.513355 192.168.0.100   10.10.1.100     TCP   1   66 48355 → 80 [SYN] Seq=2581697538 Wi
130 2019-10-13 17:09:20.514011 10.10.1.100     192.168.0.100   TCP       66 80 → 48355 [SYN, ACK] Seq=16330186
131 2019-10-13 17:09:20.514438 192.168.0.100   10.10.1.100     TCP       54 48355 → 80 [ACK] Seq=2581697539 Ac
132 2019-10-13 17:09:39.473089 192.168.0.100   10.10.1.100     TCP   2   54 48355 → 80 [RST, ACK] Seq=25816979
```

Key Points:

1. The TCP 3-way handshake goes through the firewall.
2. After ~20 sec the client gives up and sends a TCP RST.

Based on these captures it can be concluded that:

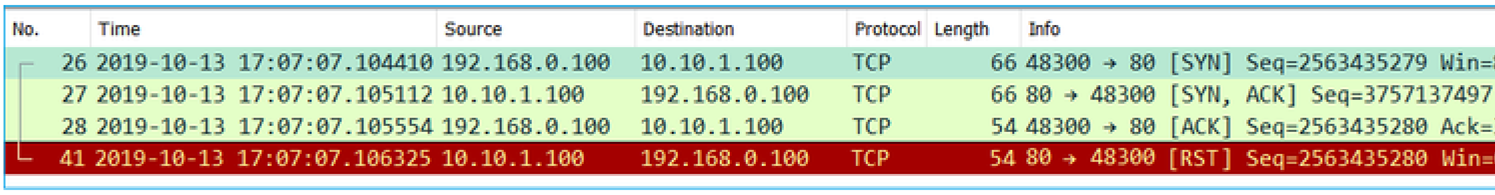
- After 5-20 seconds one endpoint gives up and decides to terminate the connection.

Recommended Actions

Same as in case 3.1

3.4 - TCP 3-way Handshake + Immediate RST from the Server

Both firewall captures CAPI and CAPO contain these packets, as shown in the image.



No.	Time	Source	Destination	Protocol	Length	Info
26	2019-10-13 17:07:07.104410	192.168.0.100	10.10.1.100	TCP	66	48300 → 80 [SYN] Seq=2563435279 Win=
27	2019-10-13 17:07:07.105112	10.10.1.100	192.168.0.100	TCP	66	80 → 48300 [SYN, ACK] Seq=3757137497
28	2019-10-13 17:07:07.105554	192.168.0.100	10.10.1.100	TCP	54	48300 → 80 [ACK] Seq=2563435280 Ack=
41	2019-10-13 17:07:07.106325	10.10.1.100	192.168.0.100	TCP	54	80 → 48300 [RST] Seq=2563435280 Win=

Key Points:

1. The TCP 3-way handshake goes through the firewall.
2. There is a TCP RST from the server a few milliseconds after the ACK packet.

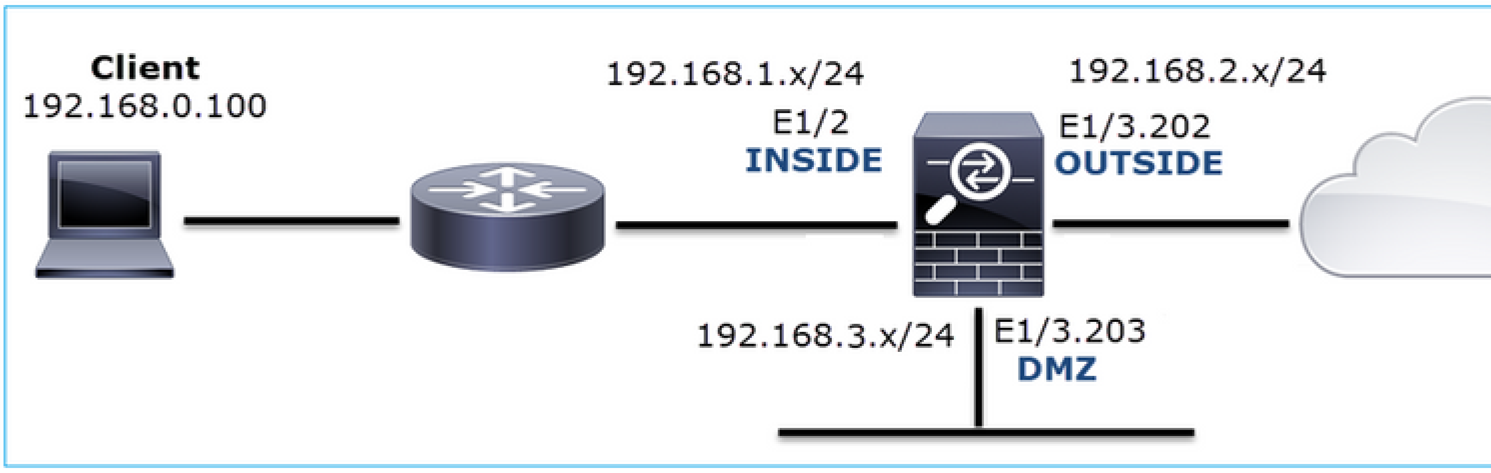
Recommended Actions

Action: Take captures as close to the server as possible.

An immediate TCP RST from the server could indicate a malfunctioning server or a device in the path that sends the TCP RST. Take a capture on the server itself and determine the source of the TCP RST.

Case 4. TCP RST from the Client

This image shows the topology:



Problem Description: HTTP does not work.

Affected Flow:

Src IP: 192.168.0.100

Dst IP: 10.10.1.100

Protocol: TCP 80

Capture Analysis

Enable captures on FTD LINA engine.

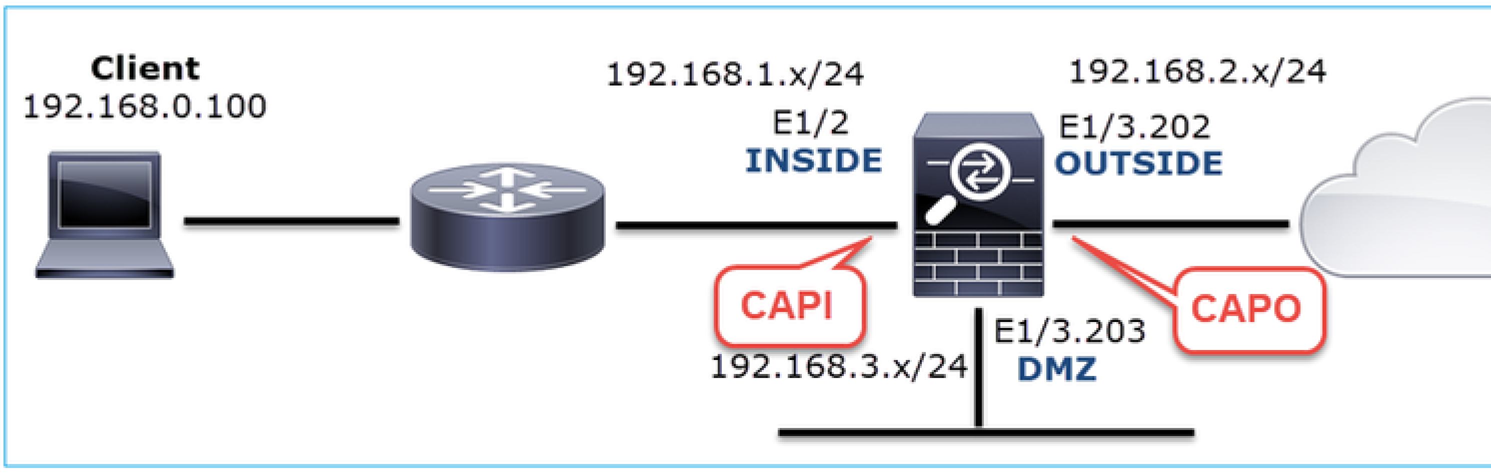
<#root>

```
firepower#
```

`capture CAPI int INSIDE match ip host 192.168.0.100 host 10.10.1.100`

```
firepower#
```

`capture CAPO int OUTSIDE match ip host 192.168.0.100 host 10.10.1.100`



Captures - Non-functional scenario:

These are the CAPI contents.

<#root>

```
firepower#
```

`show capture CAPI`

```
14 packets captured

   1: 12:32:22.860627   192.168.0.100.47078 > 10.10.1.100.80: S 4098574664:4098574664(0) win 8192 <mss 1
   2: 12:32:23.111307   192.168.0.100.47079 > 10.10.1.100.80: S 2486945841:2486945841(0) win 8192 <mss 1
   3: 12:32:23.112390   192.168.0.100.47079 > 10.10.1.100.80: R 3000518858:3000518858(0) win 0
   4: 12:32:25.858109   192.168.0.100.47078 > 10.10.1.100.80: S 4098574664:4098574664(0) win 8192 <mss 1
   5: 12:32:25.868698   192.168.0.100.47078 > 10.10.1.100.80: R 1386249853:1386249853(0) win 0
   6: 12:32:26.108118   192.168.0.100.47079 > 10.10.1.100.80: S 2486945841:2486945841(0) win 8192 <mss 1
   7: 12:32:26.109079   192.168.0.100.47079 > 10.10.1.100.80: R 3000518858:3000518858(0) win 0
   8: 12:32:26.118295   192.168.0.100.47079 > 10.10.1.100.80: R 3000518858:3000518858(0) win 0
   9: 12:32:31.859925   192.168.0.100.47078 > 10.10.1.100.80: S 4098574664:4098574664(0) win 8192 <mss 1
  10: 12:32:31.860902   192.168.0.100.47078 > 10.10.1.100.80: R 1386249853:1386249853(0) win 0
  11: 12:32:31.875229   192.168.0.100.47078 > 10.10.1.100.80: R 1386249853:1386249853(0) win 0
  12: 12:32:32.140632   192.168.0.100.47079 > 10.10.1.100.80: R 3000518858:3000518858(0) win 0
  13: 12:32:32.159995   192.168.0.100.47079 > 10.10.1.100.80: S 2486945841:2486945841(0) win 8192 <mss 1
  14: 12:32:32.160956   192.168.0.100.47079 > 10.10.1.100.80: R 3000518858:3000518858(0) win 0
14 packets shown
```

These are the CAPO contents:

<#root>

```
firepower#
```

`show capture CAPO`

```
11 packets captured

   1: 12:32:22.860780   802.1Q vlan#202 P0 192.168.0.100.47078 > 10.10.1.100.80: S 1386249852:1386249852
   2: 12:32:23.111429   802.1Q vlan#202 P0 192.168.0.100.47079 > 10.10.1.100.80: S 3000518857:3000518857
   3: 12:32:23.112405   802.1Q vlan#202 P0 192.168.0.100.47079 > 10.10.1.100.80: R 3514091874:3514091874
   4: 12:32:25.858125   802.1Q vlan#202 P0 192.168.0.100.47078 > 10.10.1.100.80: S 1386249852:1386249852
   5: 12:32:25.868729   802.1Q vlan#202 P0 192.168.0.100.47078 > 10.10.1.100.80: R 2968892337:2968892337
   6: 12:32:26.108240   802.1Q vlan#202 P0 192.168.0.100.47079 > 10.10.1.100.80: S 3822259745:3822259745
   7: 12:32:26.109094   802.1Q vlan#202 P0 192.168.0.100.47079 > 10.10.1.100.80: R 40865466:40865466(0)
   8: 12:32:31.860062   802.1Q vlan#202 P0 192.168.0.100.47078 > 10.10.1.100.80: S 4294058752:4294058752
   9: 12:32:31.860917   802.1Q vlan#202 P0 192.168.0.100.47078 > 10.10.1.100.80: R 1581733941:1581733941
  10: 12:32:32.160102   802.1Q vlan#202 P0 192.168.0.100.47079 > 10.10.1.100.80: S 4284301197:4284301197
  11: 12:32:32.160971   802.1Q vlan#202 P0 192.168.0.100.47079 > 10.10.1.100.80: R 502906918:502906918(0
11 packets shown
```

The firewall logs show:

<#root>

```
firepower#
```

`show log | i 47741`

```
Oct 13 2019 13:57:36: %FTD-6-302013: Built inbound TCP connection 4869 for INSIDE:192.168.0.100/47741 (1
Oct 13 2019 13:57:36: %FTD-6-302014: Teardown TCP connection 4869 for INSIDE:192.168.0.100/47741 to OUTS
```

`TCP Reset-O from INSIDE`

```
Oct 13 2019 13:57:39: %FTD-6-302013: Built inbound TCP connection 4870 for INSIDE:192.168.0.100/47741 (1
Oct 13 2019 13:57:39: %FTD-6-302014: Teardown TCP connection 4870 for INSIDE:192.168.0.100/47741 to OUTS
```

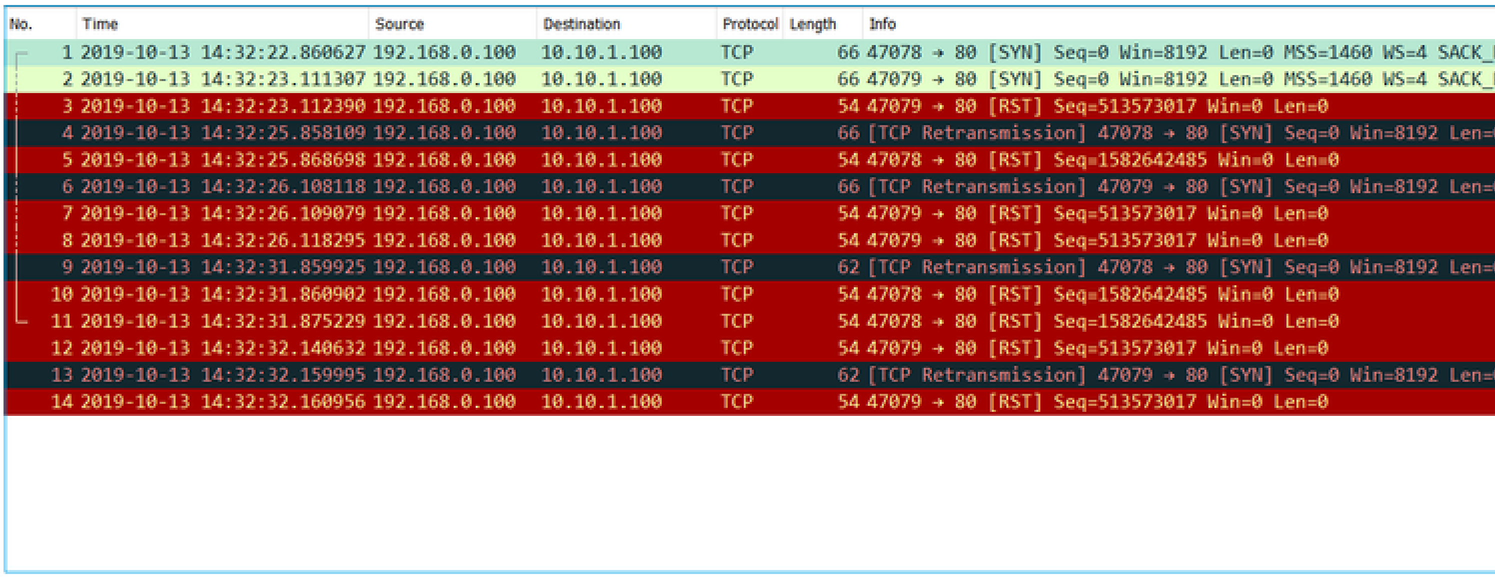
`TCP Reset-O from INSIDE`

```
Oct 13 2019 13:57:45: %FTD-6-302013: Built inbound TCP connection 4871 for INSIDE:192.168.0.100/47741 (1
Oct 13 2019 13:57:45: %FTD-6-302014: Teardown TCP connection 4871 for INSIDE:192.168.0.100/47741 to OUTS
```

These logs indicate that there is a TCP RST which arrives on firewall INSIDE interface

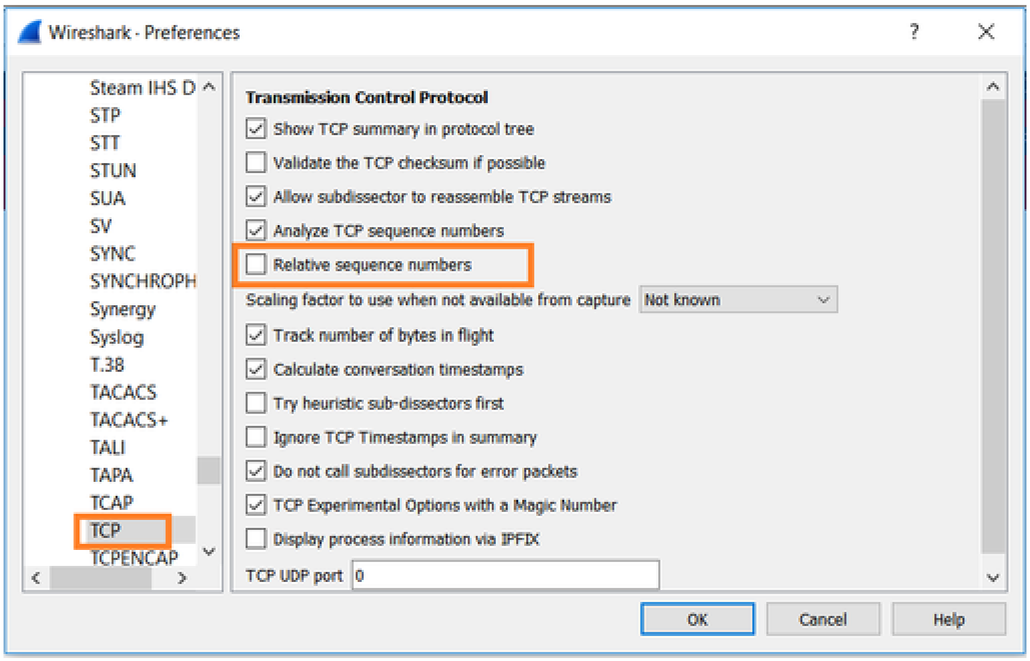
CAPI capture in Wireshark:

Follow the first TCP stream, as shown in the image.



```
No.   Time                        Source         Destination   Protocol Length Info
 1 2019-10-13 14:32:22.860627 192.168.0.100  10.10.1.100   TCP      66 47078 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_
 2 2019-10-13 14:32:23.111307 192.168.0.100  10.10.1.100   TCP      66 47079 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_
 3 2019-10-13 14:32:23.112390 192.168.0.100  10.10.1.100   TCP      54 47079 → 80 [RST] Seq=513573017 Win=0 Len=0
 4 2019-10-13 14:32:25.858109 192.168.0.100  10.10.1.100   TCP      66 [TCP Retransmission] 47078 → 80 [SYN] Seq=0 Win=8192 Len=
 5 2019-10-13 14:32:25.868698 192.168.0.100  10.10.1.100   TCP      54 47078 → 80 [RST] Seq=1582642485 Win=0 Len=0
 6 2019-10-13 14:32:26.108118 192.168.0.100  10.10.1.100   TCP      66 [TCP Retransmission] 47079 → 80 [SYN] Seq=0 Win=8192 Len=
 7 2019-10-13 14:32:26.109079 192.168.0.100  10.10.1.100   TCP      54 47079 → 80 [RST] Seq=513573017 Win=0 Len=0
 8 2019-10-13 14:32:26.118295 192.168.0.100  10.10.1.100   TCP      54 47079 → 80 [RST] Seq=513573017 Win=0 Len=0
 9 2019-10-13 14:32:31.859925 192.168.0.100  10.10.1.100   TCP      62 [TCP Retransmission] 47078 → 80 [SYN] Seq=0 Win=8192 Len=
10 2019-10-13 14:32:31.860902 192.168.0.100  10.10.1.100   TCP      54 47078 → 80 [RST] Seq=1582642485 Win=0 Len=0
11 2019-10-13 14:32:31.875229 192.168.0.100  10.10.1.100   TCP      54 47078 → 80 [RST] Seq=1582642485 Win=0 Len=0
12 2019-10-13 14:32:32.140632 192.168.0.100  10.10.1.100   TCP      54 47079 → 80 [RST] Seq=513573017 Win=0 Len=0
13 2019-10-13 14:32:32.159995 192.168.0.100  10.10.1.100   TCP      62 [TCP Retransmission] 47079 → 80 [SYN] Seq=0 Win=8192 Len=
14 2019-10-13 14:32:32.160956 192.168.0.100  10.10.1.100   TCP      54 47079 → 80 [RST] Seq=513573017 Win=0 Len=0
```

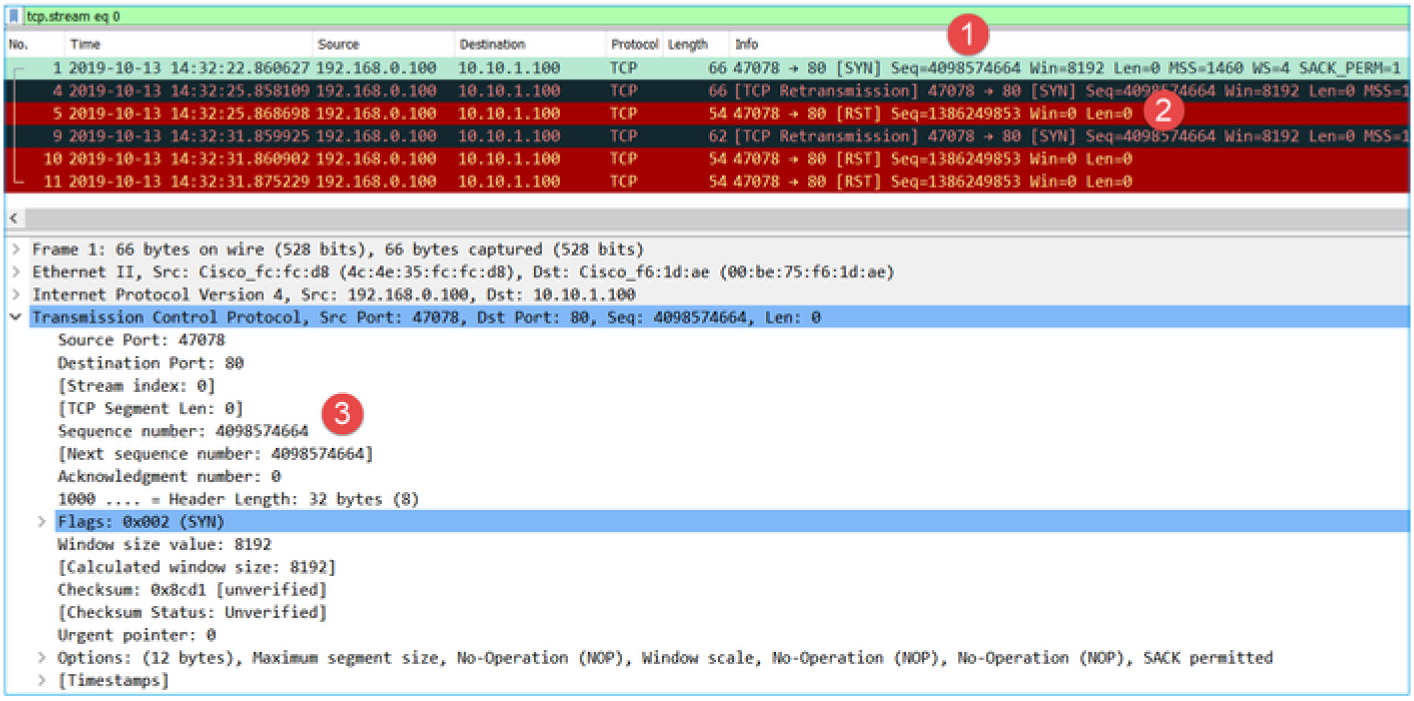
Under **Wireshark**, navigate to **Edit > Preferences > Protocols > TCP** and unselect the **Relative sequence numbers** option as shown in the image.



Transmission Control Protocol

- [✓] Show TCP summary in protocol tree
- [ ] Validate the TCP checksum if possible
- [✓] Allow subdissector to reassemble TCP streams
- [✓] Analyze TCP sequence numbers
- [ ] Relative sequence numbers
- Scaling factor to use when not available from capture: Not known
- [✓] Track number of bytes in flight
- [✓] Calculate conversation timestamps
- [ ] Try heuristic sub-dissectors first
- [ ] Ignore TCP Timestamps in summary
- [✓] Do not call subdissectors for error packets
- [✓] TCP Experimental Options with a Magic Number
- [ ] Display process information via IPFIX
- TCP UDP port: 0

This image shows the contents of the first flow in CAPI capture:

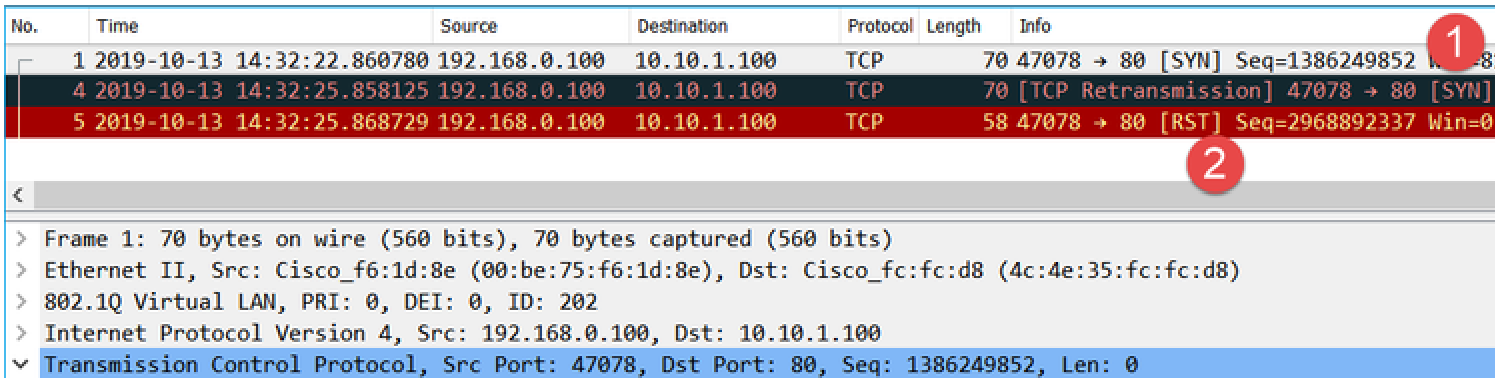


```
tcp.stream eq 0
No.  Time                        Source         Destination  Protocol Length Info
 1 2019-10-13 14:32:22.860627 192.168.0.100  10.10.1.100  TCP      66 47078 → 80 [SYN] Seq=4098574664 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
 4 2019-10-13 14:32:25.858109 192.168.0.100  10.10.1.100  TCP      66 [TCP Retransmission] 47078 → 80 [SYN] Seq=4098574664 Win=8192 Len=0 MSS=1
 5 2019-10-13 14:32:25.868698 192.168.0.100  10.10.1.100  TCP      54 47078 → 80 [RST] Seq=1386249853 Win=0 Len=0
 9 2019-10-13 14:32:31.859925 192.168.0.100  10.10.1.100  TCP      62 [TCP Retransmission] 47078 → 80 [SYN] Seq=4098574664 Win=8192 Len=0 MSS=1
10 2019-10-13 14:32:31.860902 192.168.0.100  10.10.1.100  TCP      54 47078 → 80 [RST] Seq=1386249853 Win=0 Len=0
11 2019-10-13 14:32:31.875229 192.168.0.100  10.10.1.100  TCP      54 47078 → 80 [RST] Seq=1386249853 Win=0 Len=0

> Frame 1: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
> Ethernet II, Src: Cisco_fc:fc:d8 (4c:4e:35:fc:fc:d8), Dst: Cisco_f6:1d:ae (00:be:75:f6:1d:ae)
> Internet Protocol Version 4, Src: 192.168.0.100, Dst: 10.10.1.100
v Transmission Control Protocol, Src Port: 47078, Dst Port: 80, Seq: 4098574664, Len: 0
    Source Port: 47078
    Destination Port: 80
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 4098574664
    [Next sequence number: 4098574664]
    Acknowledgment number: 0
    1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x002 (SYN)
    Window size value: 8192
    [Calculated window size: 8192]
    Checksum: 0x8cd1 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  > Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP), SACK permitted
  > [Timestamps]
```

Key Points:

1. The client sends a TCP SYN packet.
2. The client sends a TCP RST packet.
3. The TCP SYN packet has a Sequence Number value equal to 4098574664.

The same flow in CAPO capture contains:



```
No.  Time                        Source         Destination  Protocol Length Info
 1 2019-10-13 14:32:22.860780 192.168.0.100  10.10.1.100  TCP      70 47078 → 80 [SYN] Seq=1386249852 W
 4 2019-10-13 14:32:25.858125 192.168.0.100  10.10.1.100  TCP      70 [TCP Retransmission] 47078 → 80 [SYN]
 5 2019-10-13 14:32:25.868729 192.168.0.100  10.10.1.100  TCP      58 47078 → 80 [RST] Seq=2968892337 Win=0

> Frame 1: 70 bytes on wire (560 bits), 70 bytes captured (560 bits)
> Ethernet II, Src: Cisco_f6:1d:8e (00:be:75:f6:1d:8e), Dst: Cisco_fc:fc:d8 (4c:4e:35:fc:fc:d8)
> 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 202
> Internet Protocol Version 4, Src: 192.168.0.100, Dst: 10.10.1.100
v Transmission Control Protocol, Src Port: 47078, Dst Port: 80, Seq: 1386249852, Len: 0
```

Key Points:

1. The client sends a TCP SYN packet. The firewall randomizes the ISN.
2. The client sends a TCP RST packet.

Based on the two captures it can be concluded that:

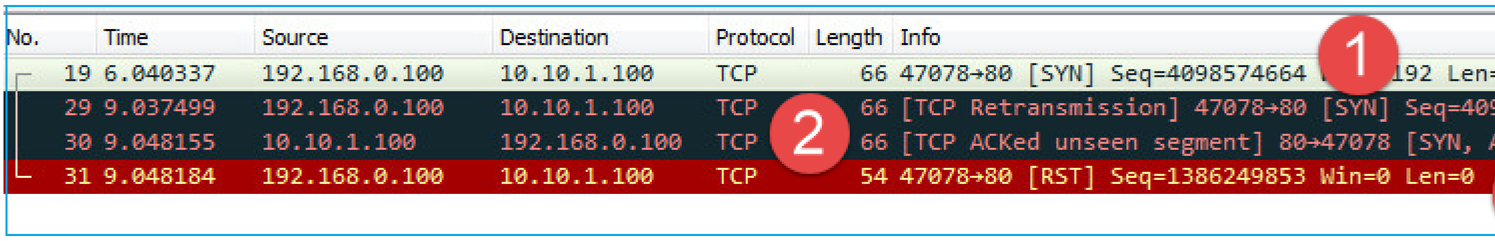
- There is no TCP 3-way handshake between the client and the server.
- There is a TCP RST which comes from the client. The TCP RST sequence number value in CAPI capture is 1386249853.

Recommended Actions

The actions listed in this section have as a goal to further narrow down the issue.

Action 1. Take a capture on the client.

Based on the captures collected on the firewall there is a strong indication of an asymmetric flow. This is based on the fact that the client sends a TCP RST with a value of 1386249853 (the randomized ISN):

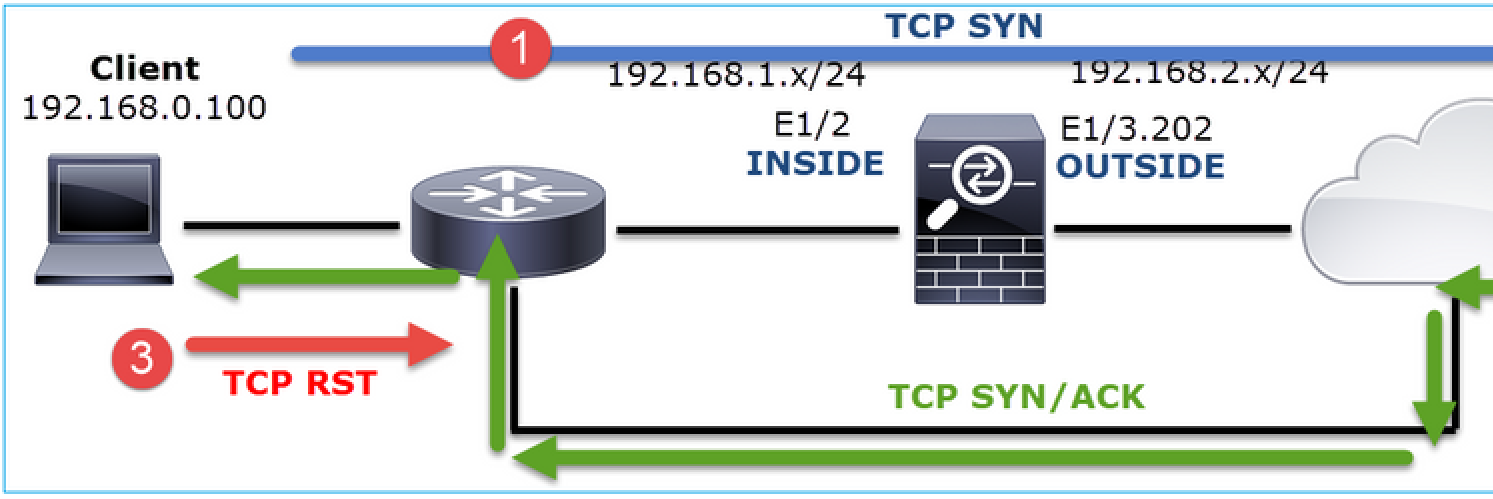


```
No.   Time      Source         Destination    Protocol Length Info
  19 6.040337  192.168.0.100  10.10.1.100    TCP          66 47078→80 [SYN] Seq=4098574664      192 Len=
  29 9.037499  192.168.0.100  10.10.1.100    TCP          66 [TCP Retransmission] 47078→80 [SYN] Seq=409
  30 9.048155  10.10.1.100    192.168.0.100  TCP          66 [TCP ACKed unseen segment] 80→47078 [SYN, A
  31 9.048184  192.168.0.100  10.10.1.100    TCP          54 47078→80 [RST] Seq=1386249853 Win=0 Len=0
```

Key Points:

1. The client sends a TCP SYN packet. The sequence number is 4098574664 and is the same as the one seen on firewall INSIDE interface (CAPI)
2. There is a TCP SYN/ACK with ACK number 1386249853 (which is expected due to ISN randomization). This packet was not seen in the firewall captures
3. The client sends a TCP RST since it expected a SYN/ACK with ACK number value of 4098574665, but it received value of 1386249853

This can be visualized as:

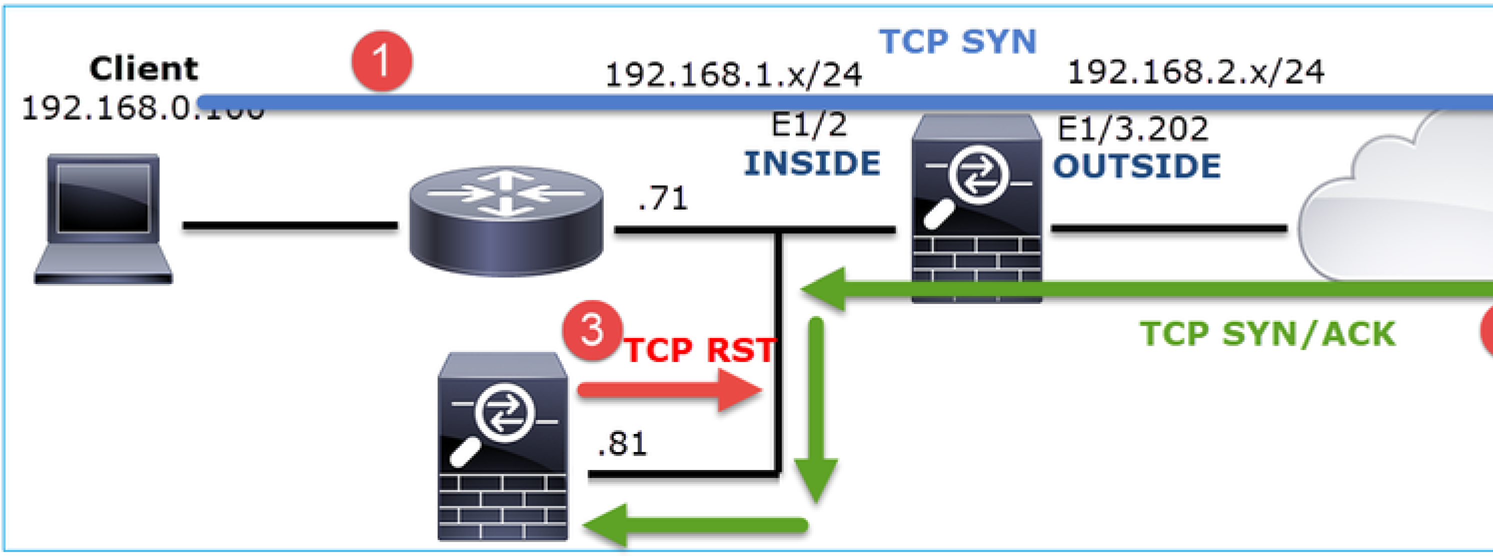


Action 2. Check the routing between the Client and the Firewall.

Confirm that:

- The MAC addresses seen in the captures are the expected ones.
- Ensure that the routing between the firewall and the client is symmetric.

There are scenarios where the RST comes from a device that sits between the firewall and the client while there is an asymmetric routing in the internal network. A typical case is shown in the image:



In this case, the capture has this content. Notice the difference between the source MAC address of the TCP SYN packet vs the source MAC address of the TCP RST and the destination MAC address of the TCP SYN/ACK packet:

<#root>

```
firepower#
```

show capture CAPI detail

```
   1: 13:57:36.730217
```

4c4e.35fc.fcd8

```
 00be.75f6.1dae 0x0800 Length: 66
      192.168.0.100.47740 > 10.10.1.100.80: S [tcp sum ok] 3045001876:3045001876(0) win 8192 <mss 1460,r
   2: 13:57:36.981104 4c4e.35fc.fcd8 00be.75f6.1dae 0x0800 Length: 66
      192.168.0.100.47741 > 10.10.1.100.80: S [tcp sum ok] 3809380540:3809380540(0) win 8192 <mss 1460,r
   3: 13:57:36.981776 00be.75f6.1dae
```

a023.9f92.2a4d

```
 0x0800 Length: 66
      10.10.1.100.80 > 192.168.0.100.47741: S [tcp sum ok] 1304153587:1304153587(0) ack 3809380541 win 8
   4: 13:57:36.982126
```

a023.9f92.2a4d

```
 00be.75f6.1dae 0x0800 Length: 54
      192.168.0.100.47741 > 10.10.1.100.80:
```

R

```
 [tcp sum ok] 3809380541:3809380541(0) ack 1304153588 win 8192 (ttl 255, id 48501)
...
```

Case 5. Slow TCP Transfer (Scenario 1)

Problem Description:

SFTP transfer between hosts 10.11.4.171 and 10.77.19.11 is slow. Although the minimum bandwidth (BW) between the 2 hosts is 100 Mbps the transfer speed does not go beyond 5 Mbps.

At the same time, the transfer speed between hosts 10.11.2.124 and 172.25.18.134 is quite higher.

Background Theory:

The maximum transfer speed for a single TCP flow is determined by the Bandwidth Delay Product (BDP). The formula used is shown in the image:

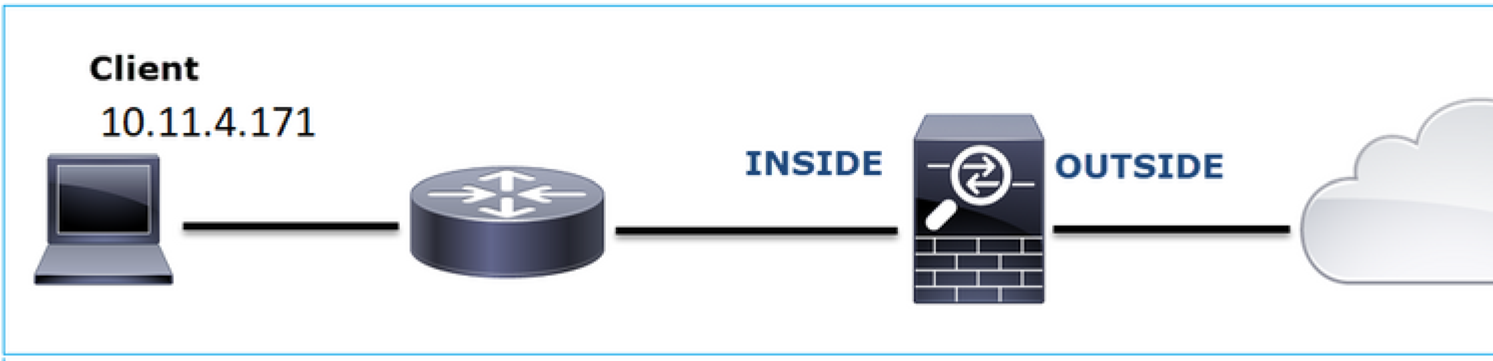
$$\text{Max Single TCP Flow Throughput [bps]} = \frac{\text{TCP Window (Bytes)}}{\text{RTT (Seconds)}} \times 8 \text{ [bits/Byte]}$$

For more details about the BDP check the resources here:

- [Why Your Application only Uses 10Mbps Even the Link is 1Gbps?]
- [BRKSEC-3021 - Advanced - Maximizing Firewall Performance]

Scenario 1. Slow Transfer

This image shows the topology:



Affected Flow:

Src IP: 10.11.4.171

Dst IP: 10.77.19.11

Protocol: SFTP (FTP over SSH)

Capture Analysis

Enable captures on FTD LINA engine:

<#root>

```
firepower#
```

`capture CAPI int INSIDE buffer 33554432 match ip host 10.11.4.171 host 10.77.19.11`

```
firepower#
```

`capture CAPO int OUTSIDE buffer 33554432 match ip host 10.11.4.171 host 10.77.19.11`

> **Warning**: LINA captures on FP1xxx and FP21xx captures affect the transfer rate of traffic that goes through the FTD. Do not enable LINA captures on FP1xxx and FP21xxx platforms when you troubleshoot performance (slow transfer through the FTD) issues. Instead use SPAN or a HW Tap device in addition to captures on the source and destination hosts. The issue is documented in Cisco bug ID CSCvo30697 .

<#root>

```
firepower#
```

`capture CAPI type raw-data trace interface inside match icmp any any`

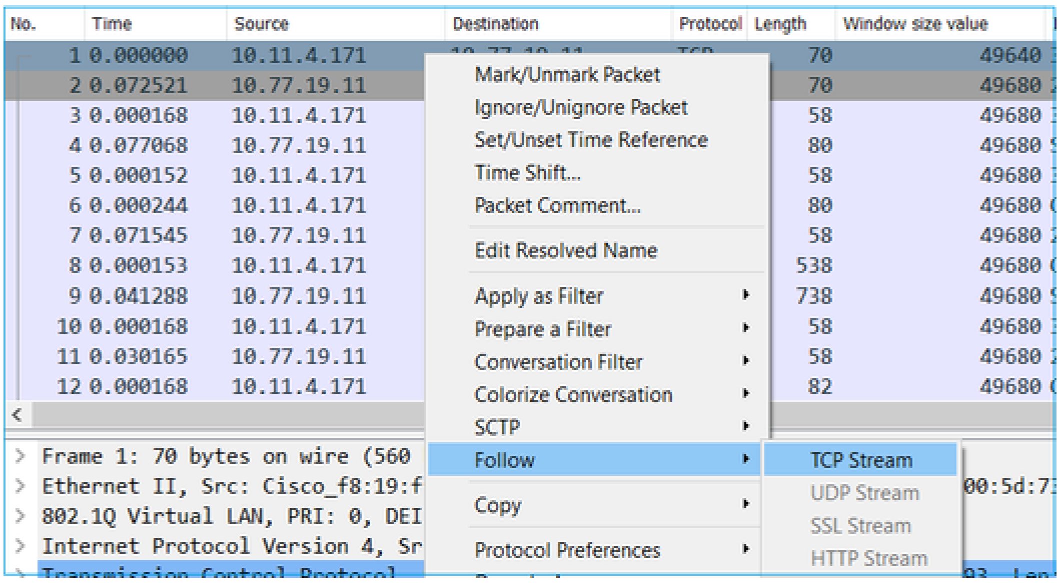
```
WARNING: Running packet capture can have an adverse impact on performance.
```

Recommended Actions

The actions listed in this section have as a goal to further narrow down the issue.

Round Trip Time (RTT) Calculation

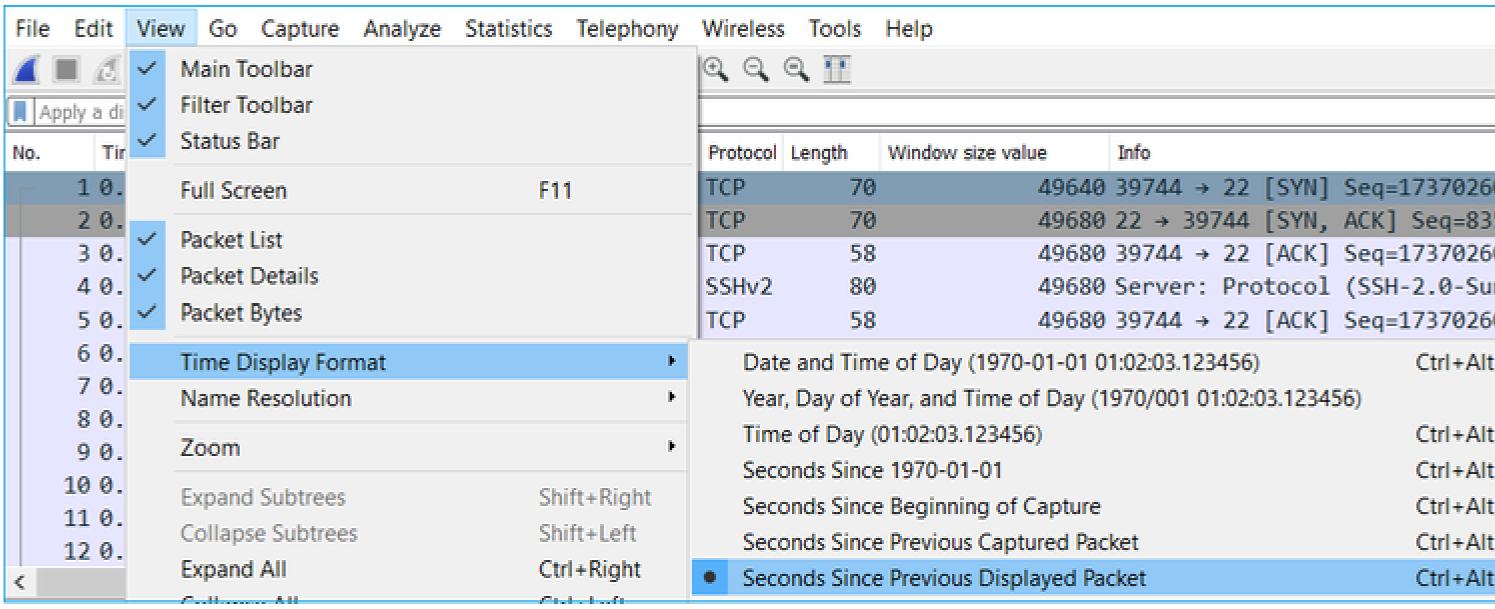
First, identify the transfer flow and follow it:



No.	Time	Source	Destination	Protocol	Length	Window size value
1	0.000000	10.11.4.171			70	49640
2	0.072521	10.77.19.11			70	49680
3	0.000168	10.11.4.171			58	49680
4	0.077068	10.77.19.11			80	49680
5	0.000152	10.11.4.171			58	49680
6	0.000244	10.11.4.171			80	49680
7	0.071545	10.77.19.11			58	49680
8	0.000153	10.11.4.171			538	49680
9	0.041288	10.77.19.11			738	49680
10	0.000168	10.11.4.171			58	49680
11	0.030165	10.77.19.11			58	49680
12	0.000168	10.11.4.171			82	49680

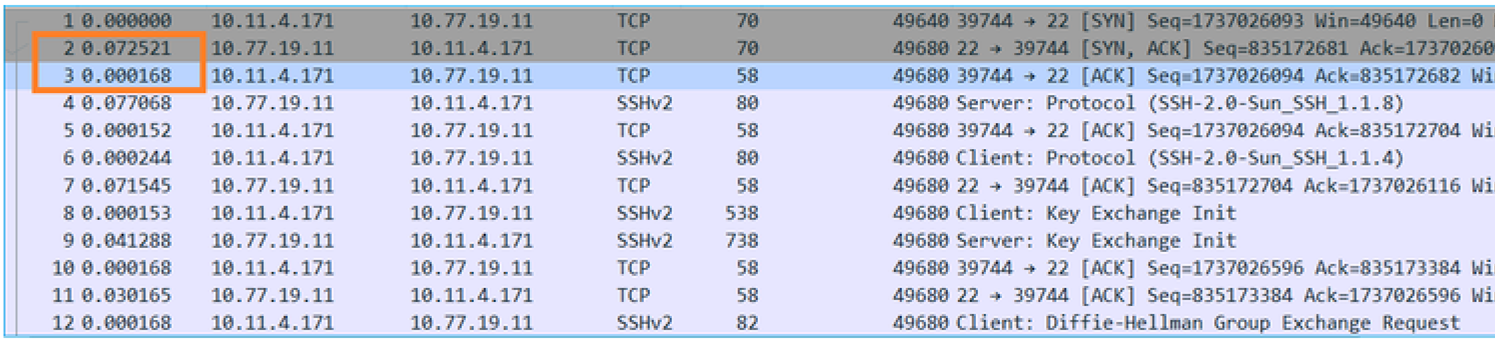
```
> Frame 1: 70 bytes on wire (560
> Ethernet II, Src: Cisco_f8:19:f                                        00:5d:7
> 802.1Q Virtual LAN, PRI: 0, DEI
> Internet Protocol Version 4, Sr
```

Change the Wireshark View to show the **Seconds Since the Previous Displayed Packet**. This eases the calculation of the RTT:


File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
✓ Main Toolbar
✓ Filter Toolbar
✓ Status Bar
Full Screen F11
✓ Packet List
✓ Packet Details
✓ Packet Bytes
Time Display Format
Name Resolution
Zoom
Expand Subtrees Shift+Right
Collapse Subtrees Shift+Left
Expand All Ctrl+Right
Date and Time of Day (1970-01-01 01:02:03.123456) Ctrl+Alt
Year, Day of Year, and Time of Day (1970/001 01:02:03.123456)
Time of Day (01:02:03.123456) Ctrl+Alt
Seconds Since 1970-01-01 Ctrl+Alt
Seconds Since Beginning of Capture Ctrl+Alt
Seconds Since Previous Captured Packet Ctrl+Alt
● Seconds Since Previous Displayed Packet Ctrl+Alt

```
No.  Ti   Protocol Length  Window size value  Info
1 0.      TCP      70      49640 39744 → 22 [SYN] Seq=1737026
2 0.      TCP      70      49680 22 → 39744 [SYN, ACK] Seq=83
3 0.      TCP      58      49680 39744 → 22 [ACK] Seq=1737026
4 0.      SSHv2    80      49680 Server: Protocol (SSH-2.0-Su
5 0.      TCP      58      49680 39744 → 22 [ACK] Seq=1737026
```

The RTT can be calculated by addition of the time values between 2 packet exchanges (one towards the source and one towards the destination). In this case, packet #2 shows the RTT between the firewall and the device who sent the SYN/ACK packet (server). Packet #3 shows the RTT between the firewall and the device who sent the ACK packet (client). The addition of the 2 numbers provides a good estimate about the end-to-end RTT:

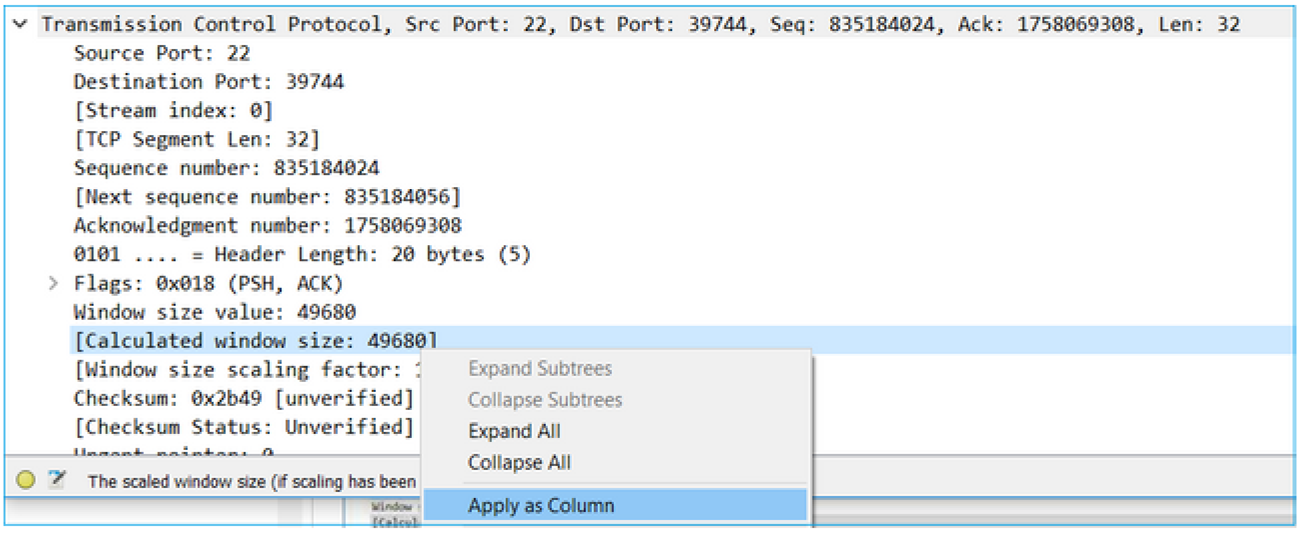


```
 1 0.000000  10.11.4.171   10.77.19.11   TCP    70   49640 39744 → 22 [SYN] Seq=1737026093 Win=49640 Len=0
 2 0.072521  10.77.19.11   10.11.4.171   TCP    70   49680 22 → 39744 [SYN, ACK] Seq=835172681 Ack=17370260
 3 0.000168  10.11.4.171   10.77.19.11   TCP    58   49680 39744 → 22 [ACK] Seq=1737026094 Ack=835172682 Wi
 4 0.077068  10.77.19.11   10.11.4.171   SSHv2  80   49680 Server: Protocol (SSH-2.0-Sun_SSH_1.1.8)
 5 0.000152  10.11.4.171   10.77.19.11   TCP    58   49680 39744 → 22 [ACK] Seq=1737026094 Ack=835172704 Wi
 6 0.000244  10.11.4.171   10.77.19.11   SSHv2  80   49680 Client: Protocol (SSH-2.0-Sun_SSH_1.1.4)
 7 0.071545  10.77.19.11   10.11.4.171   TCP    58   49680 22 → 39744 [ACK] Seq=835172704 Ack=1737026116 Wi
 8 0.000153  10.11.4.171   10.77.19.11   SSHv2  538  49680 Client: Key Exchange Init
 9 0.041288  10.77.19.11   10.11.4.171   SSHv2  738  49680 Server: Key Exchange Init
10 0.000168  10.11.4.171   10.77.19.11   TCP    58   49680 39744 → 22 [ACK] Seq=1737026596 Ack=835173384 Wi
11 0.030165  10.77.19.11   10.11.4.171   TCP    58   49680 22 → 39744 [ACK] Seq=835173384 Ack=1737026596 Wi
12 0.000168  10.11.4.171   10.77.19.11   SSHv2  82   49680 Client: Diffie-Hellman Group Exchange Request
```

RTT â‰ˆ 80 msec

TCP Window Size Calculation

Expand a TCP packet, expand the TCP header, select **Calculated window size** and select **Apply as Column:**

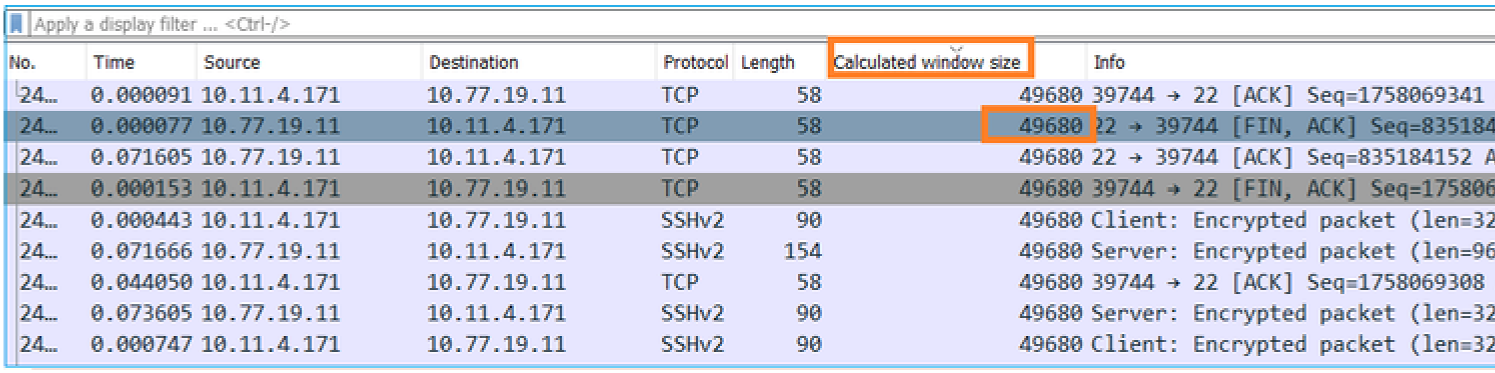


```
Transmission Control Protocol, Src Port: 22, Dst Port: 39744, Seq: 835184024, Ack: 1758069308, Len: 32
    Source Port: 22
    Destination Port: 39744
    [Stream index: 0]
    [TCP Segment Len: 32]
    Sequence number: 835184024
    [Next sequence number: 835184056]
    Acknowledgment number: 1758069308
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x018 (PSH, ACK)
    Window size value: 49680
    [Calculated window size: 49680]
    [Window size scaling factor: 1
    Checksum: 0x2b49 [unverified]
    [Checksum Status: Unverified]
The scaled window size (if scaling has been
```

Check the **Calculated window size value** column to see what the maximum window size value was during the TCP session. You can also select on the column name and sort the values.

If you test a file download (**server** > **client**) you must check the values advertised by the server. The maximum window size value advertised by the server determines the maximum transfer speed achieved.

In this case, the TCP window size is â‰ˆ 50000 Bytes



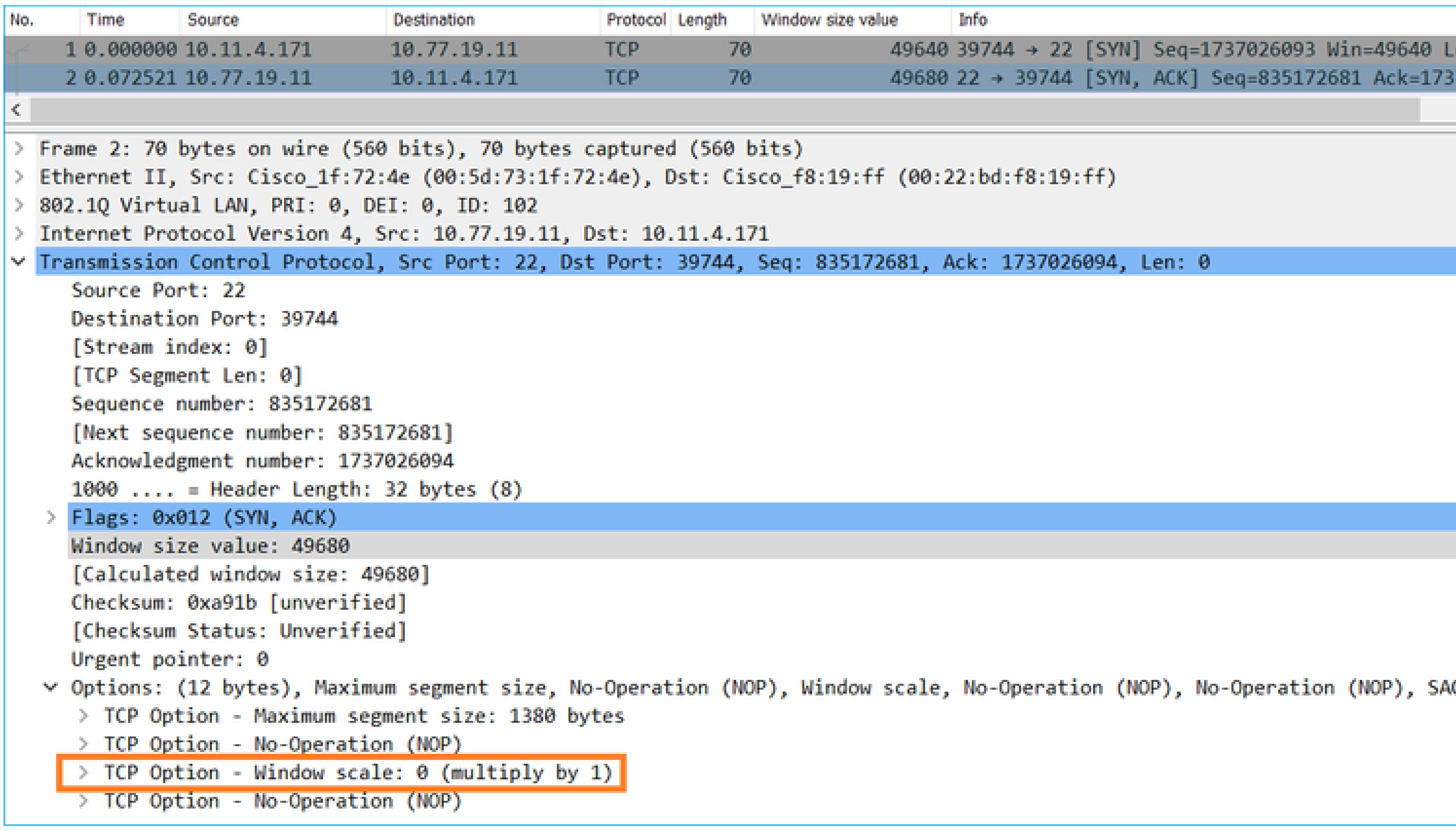
Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Calculated window size	Info
24...	0.000091	10.11.4.171	10.77.19.11	TCP	58	49680	39744 → 22 [ACK] Seq=1758069341
24...	0.000077	10.77.19.11	10.11.4.171	TCP	58	49680	22 → 39744 [FIN, ACK] Seq=835184
24...	0.071605	10.77.19.11	10.11.4.171	TCP	58	49680	22 → 39744 [ACK] Seq=835184152 A
24...	0.000153	10.11.4.171	10.77.19.11	TCP	58	49680	39744 → 22 [FIN, ACK] Seq=175806
24...	0.000443	10.11.4.171	10.77.19.11	SSHv2	90	49680	Client: Encrypted packet (len=32
24...	0.071666	10.77.19.11	10.11.4.171	SSHv2	154	49680	Server: Encrypted packet (len=96
24...	0.044050	10.11.4.171	10.77.19.11	TCP	58	49680	39744 → 22 [ACK] Seq=1758069308
24...	0.073605	10.77.19.11	10.11.4.171	SSHv2	90	49680	Server: Encrypted packet (len=32
24...	0.000747	10.11.4.171	10.77.19.11	SSHv2	90	49680	Client: Encrypted packet (len=32

Based on these values and with the use of the Bandwidth Delay Product formula you get the maximum theoretical bandwidth that can be achieved under these conditions: 50000*8/0.08 = 5 Mbps maximum theoretical bandwidth.

This matches what the client experiences in this case.

Check closely the TCP 3-way handshake. Both sides, and more importantly the server, advertise a window scale value of 0 which means 2^0 = 1 (no windows scaling). This affects negatively the transfer rate:

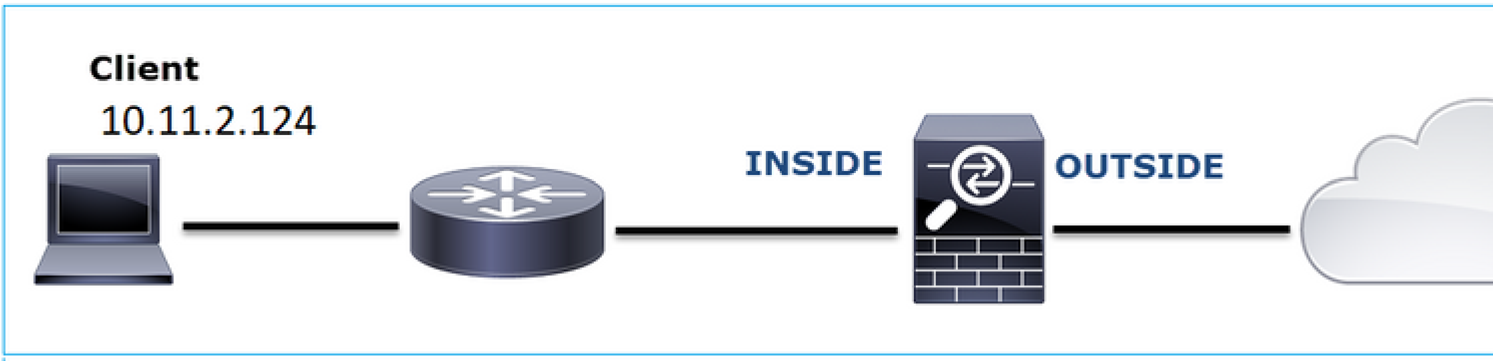

```
No.  Time      Source       Destination  Protocol Length  Window size value  Info
   1 0.000000 10.11.4.171  10.77.19.11  TCP      70       49640 39744 → 22 [SYN] Seq=1737026093 Win=49640 L
   2 0.072521 10.77.19.11  10.11.4.171  TCP      70       49680 22 → 39744 [SYN, ACK] Seq=835172681 Ack=173

> Frame 2: 70 bytes on wire (560 bits), 70 bytes captured (560 bits)
> Ethernet II, Src: Cisco_1f:72:4e (00:5d:73:1f:72:4e), Dst: Cisco_f8:19:ff (00:22:bd:f8:19:ff)
> 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 102
> Internet Protocol Version 4, Src: 10.77.19.11, Dst: 10.11.4.171
v Transmission Control Protocol, Src Port: 22, Dst Port: 39744, Seq: 835172681, Ack: 1737026094, Len: 0
    Source Port: 22
    Destination Port: 39744
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 835172681
    [Next sequence number: 835172681]
    Acknowledgment number: 1737026094
    1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x012 (SYN, ACK)
    Window size value: 49680
    [Calculated window size: 49680]
    Checksum: 0xa91b [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  v Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP), SAC
      > TCP Option - Maximum segment size: 1380 bytes
      > TCP Option - No-Operation (NOP)
      > TCP Option - Window scale: 0 (multiply by 1)
      > TCP Option - No-Operation (NOP)
```

At this point, there is a need to take a capture on the server, confirm that it is the one who advertises window scale = 0 and reconfigure it (check the server documentation for how to do this).

Scenario 2. Fast transfer

Now letâ€™s examine the good scenario (fast transfer through the same network):

Topology:



The flow of interest:

Src IP: 10.11.2.124

Dst IP: 172.25.18.134

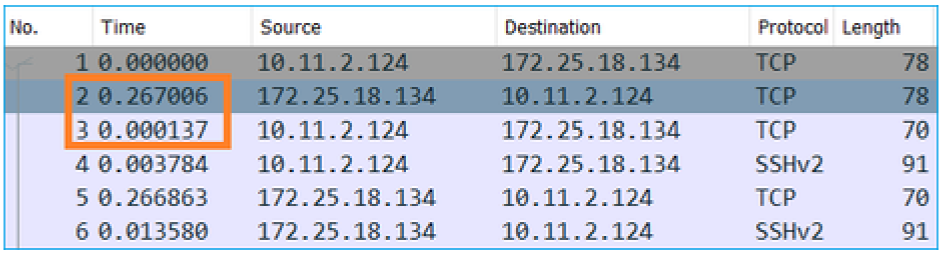
Protocol: SFTP (FTP over SSH)

Enable Captures on FTD LINA engine

<#root>

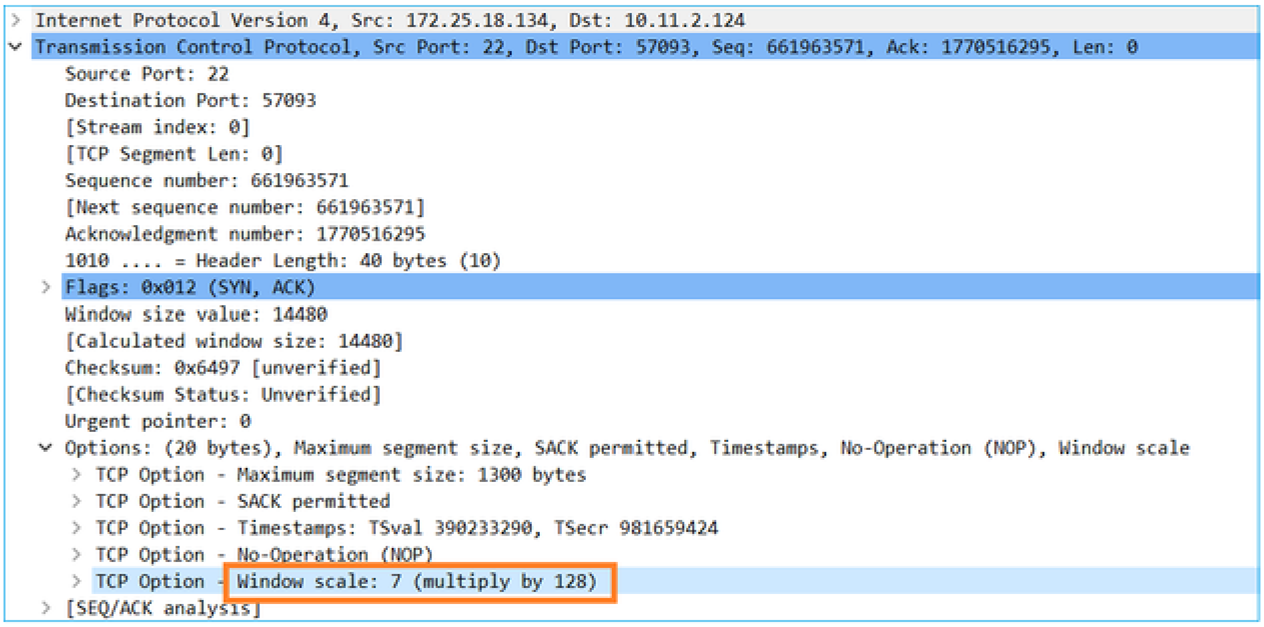
```
firepower#

capture CAPI int INSIDE buffer 33554432 match ip host 10.11.2.124 host 172.25.18.134

firepower#

capture CAPO int OUTSIDE buffer 33554432 match ip host 10.11.2.124 host 172.25.18.134
```

Round Trip Time (RTT) Calculation: In this case, the RTT is â‰ˆ 300 msec.



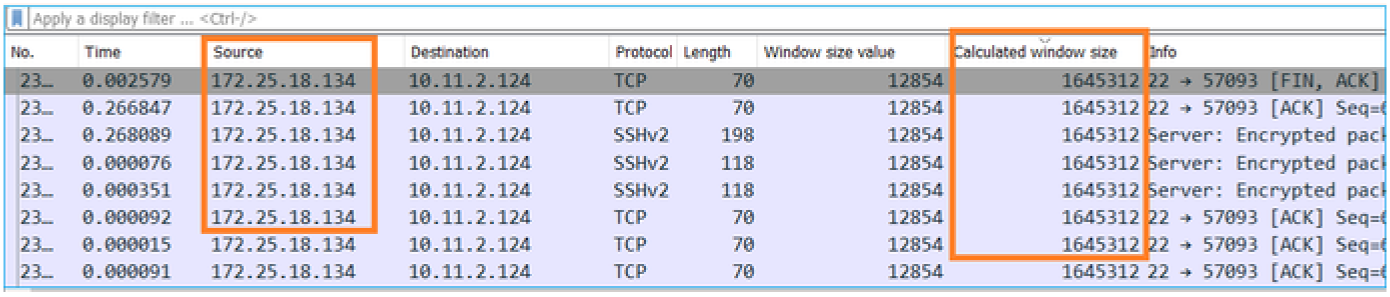
```
No.  Time        Source          Destination     Protocol  Length
  1 0.000000    10.11.2.124     172.25.18.134   TCP           78
  2 0.267006    172.25.18.134   10.11.2.124     TCP           78
  3 0.000137    10.11.2.124     172.25.18.134   TCP           70
  4 0.003784    10.11.2.124     172.25.18.134   SSHv2         91
  5 0.266863    172.25.18.134   10.11.2.124     TCP           70
  6 0.013580    172.25.18.134   10.11.2.124     SSHv2         91
```

TCP Window Size Calculation: The server advertises a TCP window scale factor of 7.



```
> Internet Protocol Version 4, Src: 172.25.18.134, Dst: 10.11.2.124
v Transmission Control Protocol, Src Port: 22, Dst Port: 57093, Seq: 661963571, Ack: 1770516295, Len: 0
    Source Port: 22
    Destination Port: 57093
    [Stream index: 0]
    [TCP Segment Len: 0]
    Sequence number: 661963571
    [Next sequence number: 661963571]
    Acknowledgment number: 1770516295
    1010 .... = Header Length: 40 bytes (10)
  > Flags: 0x012 (SYN, ACK)
    Window size value: 14480
    [Calculated window size: 14480]
    Checksum: 0x6497 [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  v Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale
      > TCP Option - Maximum segment size: 1300 bytes
      > TCP Option - SACK permitted
      > TCP Option - Timestamps: TSval 390233290, TSecr 981659424
      > TCP Option - No-Operation (NOP)
      > TCP Option - Window scale: 7 (multiply by 128)
  > [SEQ/ACK analysis]
```

The serverâ€™s TCP window size is â‰ˆ 1600000 Bytes:



```
Apply a display filter ... <Ctrl-/>
No.  Time       Source          Destination    Protocol Length  Window size value  Calculated window size  Info
23…  0.002579   172.25.18.134   10.11.2.124    TCP         70              12854                 1645312  22 → 57093 [FIN, ACK]
23…  0.266847   172.25.18.134   10.11.2.124    TCP         70              12854                 1645312  22 → 57093 [ACK] Seq=
23…  0.268089   172.25.18.134   10.11.2.124    SSHv2      198              12854                 1645312  Server: Encrypted pac
23…  0.000076   172.25.18.134   10.11.2.124    SSHv2      118              12854                 1645312  Server: Encrypted pac
23…  0.000351   172.25.18.134   10.11.2.124    SSHv2      118              12854                 1645312  Server: Encrypted pac
23…  0.000092   172.25.18.134   10.11.2.124    TCP         70              12854                 1645312  22 → 57093 [ACK] Seq=
23…  0.000015   172.25.18.134   10.11.2.124    TCP         70              12854                 1645312  22 → 57093 [ACK] Seq=
23…  0.000091   172.25.18.134   10.11.2.124    TCP         70              12854                 1645312  22 → 57093 [ACK] Seq=
```

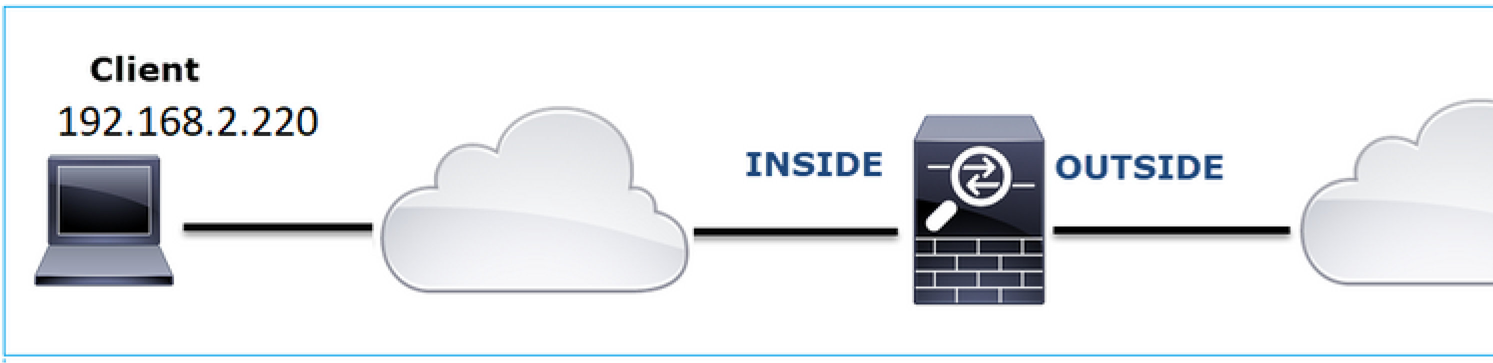
Based on these values the Bandwidth Delay Product formula gives:

1600000*8/0.3 = 43 Mbps maximum theoretical transfer speed

Case 6. Slow TCP Transfer (Scenario 2)

Problem Description: FTP file transfer (download) through the firewall is slow.

This image shows the Topology:



Affected Flow:

Src IP: 192.168.2.220

Dst IP: 192.168.1.220

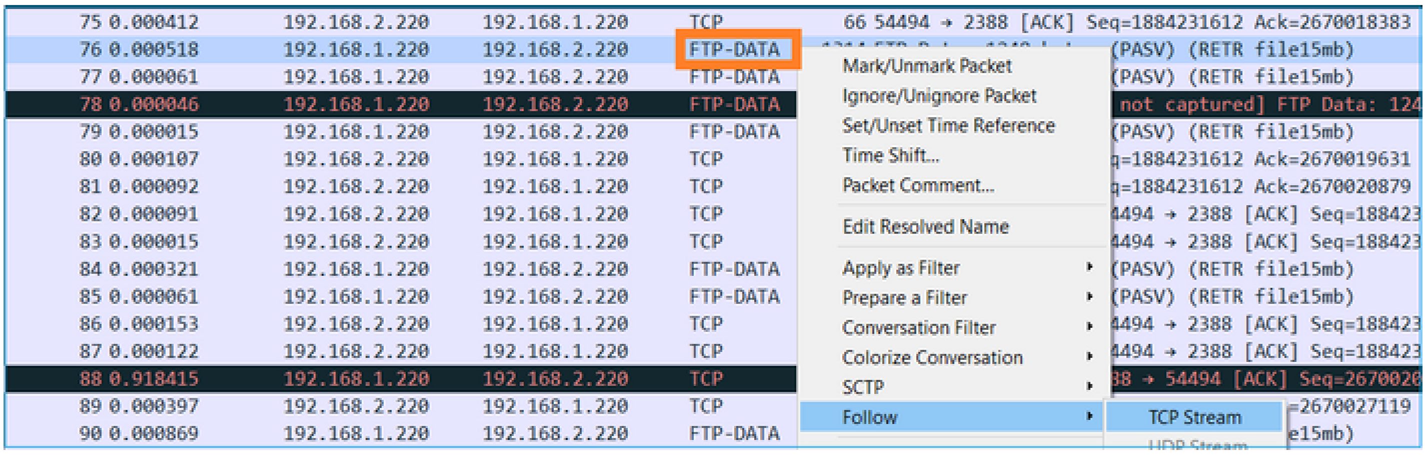
Protocol: FTP

Capture Analysis

Enable captures on the FTD LINA engine.

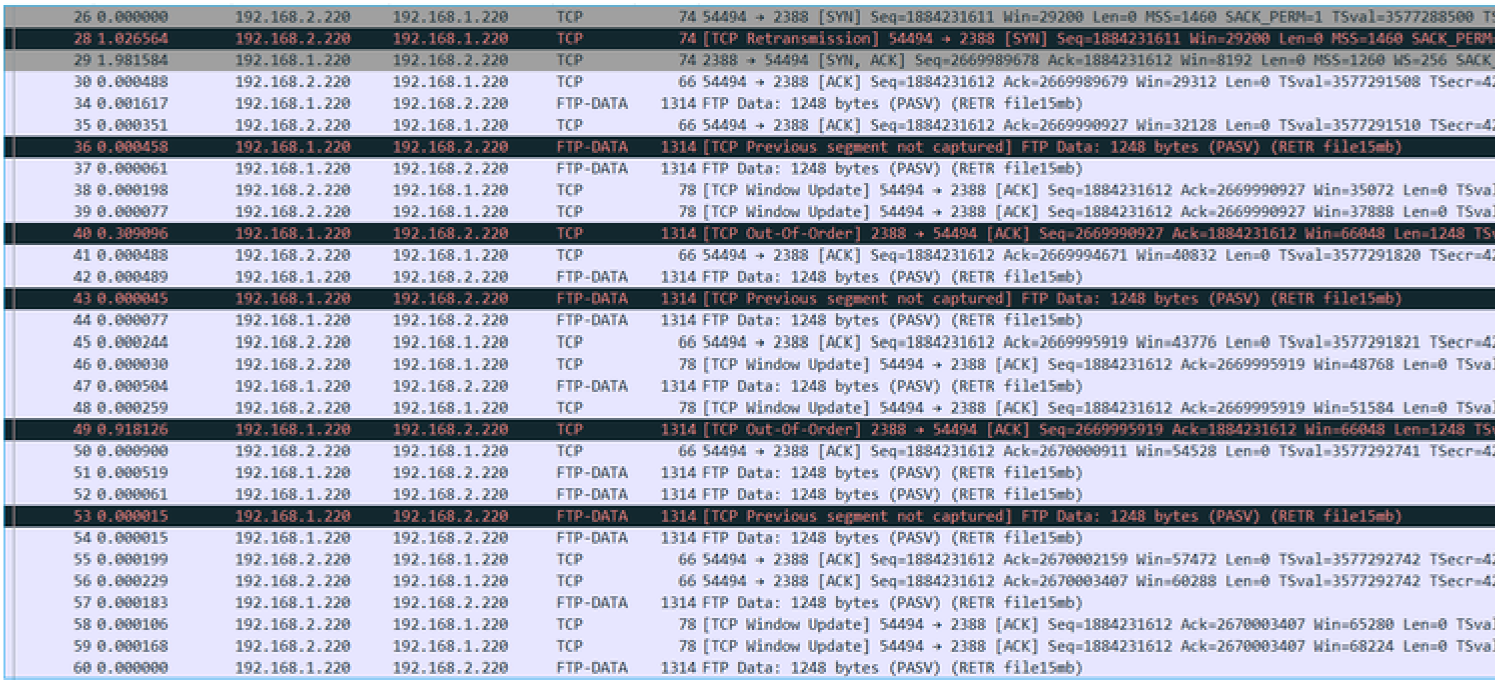
```
<#root>
firepower#
capture CAPI type raw-data buffer 33554432 interface INSIDE match tcp host 192.168.2.220 host 192.168.1.
firepower#
cap CAPO type raw-data buffer 33554432 interface OUTSIDE match tcp host 192.168.2.220 host 192.168.1.220
```

Select an FTP-DATA packet and follow the FTP Data Channel on FTD INSIDE capture (CAPI):



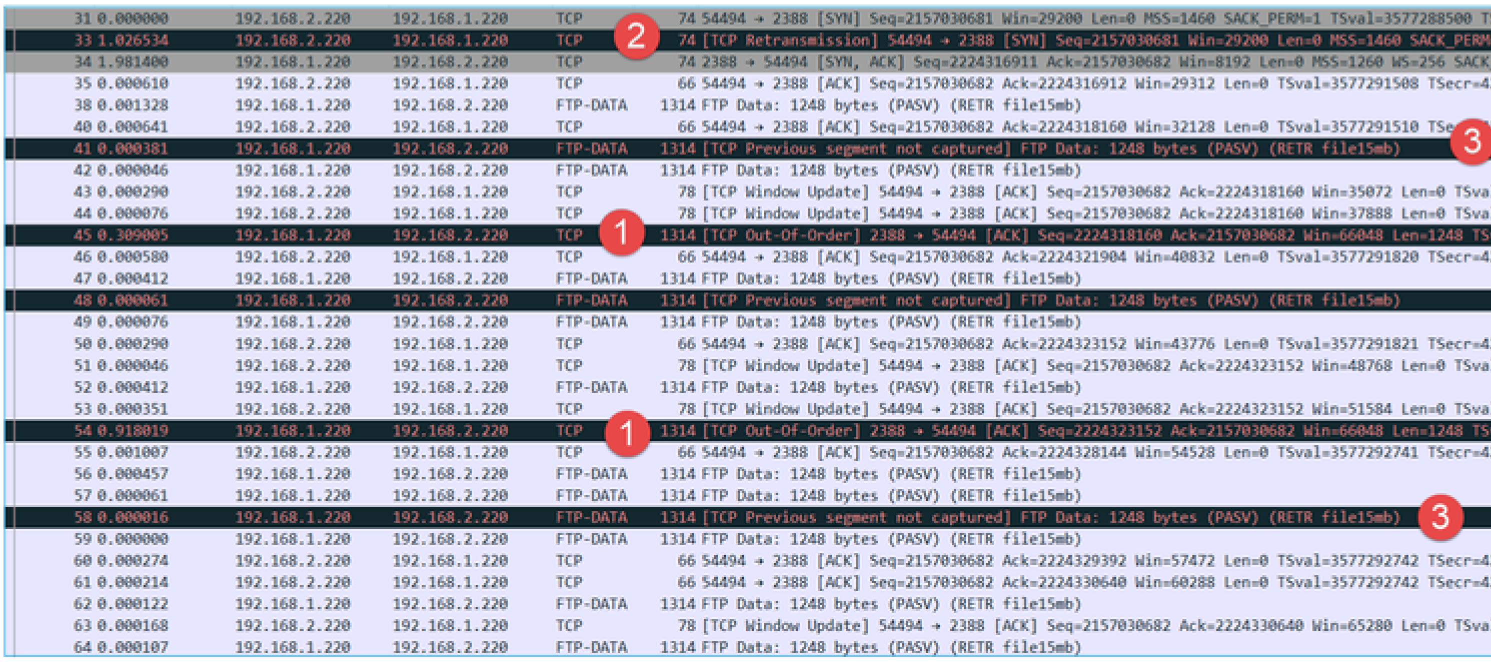
```
75 0.000412   192.168.2.220   192.168.1.220   TCP        66 54494 → 2388 [ACK] Seq=1884231612 Ack=2670018383
76 0.000518   192.168.1.220   192.168.2.220   FTP-DATA   (PASV) (RETR file15mb)
77 0.000061   192.168.1.220   192.168.2.220   FTP-DATA   (PASV) (RETR file15mb)
78 0.000046   192.168.1.220   192.168.2.220   FTP-DATA   not captured] FTP Data: 124
79 0.000015   192.168.1.220   192.168.2.220   FTP-DATA   (PASV) (RETR file15mb)
80 0.000107   192.168.2.220   192.168.1.220   TCP        q=1884231612 Ack=2670019631
81 0.000092   192.168.2.220   192.168.1.220   TCP        q=1884231612 Ack=2670020879
82 0.000091   192.168.2.220   192.168.1.220   TCP        4494 → 2388 [ACK] Seq=188423
83 0.000015   192.168.2.220   192.168.1.220   TCP        4494 → 2388 [ACK] Seq=188423
84 0.000321   192.168.1.220   192.168.2.220   FTP-DATA   (PASV) (RETR file15mb)
85 0.000061   192.168.1.220   192.168.2.220   FTP-DATA   (PASV) (RETR file15mb)
86 0.000153   192.168.2.220   192.168.1.220   TCP        4494 → 2388 [ACK] Seq=188423
87 0.000122   192.168.2.220   192.168.1.220   TCP        4494 → 2388 [ACK] Seq=188423
88 0.918415   192.168.1.220   192.168.2.220   TCP        38 → 54494 [ACK] Seq=2670020
89 0.000397   192.168.2.220   192.168.1.220   TCP        =2670027119
90 0.000869   192.168.1.220   192.168.2.220   FTP-DATA   e15mb)
```

The FTP-DATA stream content:



```
26 0.000000   192.168.2.220   192.168.1.220   TCP          74 54494 → 2388 [SYN] Seq=1884231611 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=3577288500 TS
28 1.026564   192.168.2.220   192.168.1.220   TCP          74 [TCP Retransmission] 54494 → 2388 [SYN] Seq=1884231611 Win=29200 Len=0 MSS=1460 SACK_PERM
29 1.981584   192.168.1.220   192.168.2.220   TCP          74 2388 → 54494 [SYN, ACK] Seq=2669989678 Ack=1884231612 Win=8192 Len=0 MSS=1260 WS=256 SACK_
30 0.000488   192.168.2.220   192.168.1.220   TCP          66 54494 → 2388 [ACK] Seq=1884231612 Ack=2669989679 Win=29312 Len=0 TSval=3577291508 TSecr=4
34 0.001617   192.168.1.220   192.168.2.220   FTP-DATA   1314 FTP Data: 1248 bytes (PASV) (RETR file15mb)
35 0.000351   192.168.2.220   192.168.1.220   TCP          66 54494 → 2388 [ACK] Seq=1884231612 Ack=2669990927 Win=32128 Len=0 TSval=3577291510 TSecr=4
36 0.000458   192.168.1.220   192.168.2.220   FTP-DATA   1314 [TCP Previous segment not captured] FTP Data: 1248 bytes (PASV) (RETR file15mb)
37 0.000061   192.168.1.220   192.168.2.220   FTP-DATA   1314 FTP Data: 1248 bytes (PASV) (RETR file15mb)
38 0.000198   192.168.2.220   192.168.1.220   TCP          78 [TCP Window Update] 54494 → 2388 [ACK] Seq=1884231612 Ack=2669990927 Win=35072 Len=0 TSval
39 0.000077   192.168.2.220   192.168.1.220   TCP          78 [TCP Window Update] 54494 → 2388 [ACK] Seq=1884231612 Ack=2669990927 Win=37888 Len=0 TSval
40 0.309096   192.168.1.220   192.168.2.220   TCP        1314 [TCP Out-Of-Order] 2388 → 54494 [ACK] Seq=2669990927 Ack=1884231612 Win=66048 Len=1248 TS
41 0.000488   192.168.2.220   192.168.1.220   TCP          66 54494 → 2388 [ACK] Seq=1884231612 Ack=2669994671 Win=40832 Len=0 TSval=3577291820 TSecr=4
42 0.000489   192.168.1.220   192.168.2.220   FTP-DATA   1314 FTP Data: 1248 bytes (PASV) (RETR file15mb)
43 0.000045   192.168.1.220   192.168.2.220   FTP-DATA   1314 [TCP Previous segment not captured] FTP Data: 1248 bytes (PASV) (RETR file15mb)
44 0.000077   192.168.1.220   192.168.2.220   FTP-DATA   1314 FTP Data: 1248 bytes (PASV) (RETR file15mb)
45 0.000244   192.168.2.220   192.168.1.220   TCP          66 54494 → 2388 [ACK] Seq=1884231612 Ack=2669995919 Win=43776 Len=0 TSval=3577291821 TSecr=4
46 0.000030   192.168.2.220   192.168.1.220   TCP          78 [TCP Window Update] 54494 → 2388 [ACK] Seq=1884231612 Ack=2669995919 Win=48768 Len=0 TSval
47 0.000504   192.168.1.220   192.168.2.220   FTP-DATA   1314 FTP Data: 1248 bytes (PASV) (RETR file15mb)
48 0.000259   192.168.2.220   192.168.1.220   TCP          78 [TCP Window Update] 54494 → 2388 [ACK] Seq=1884231612 Ack=2669995919 Win=51584 Len=0 TSval
49 0.918126   192.168.1.220   192.168.2.220   TCP        1314 [TCP Out-Of-Order] 2388 → 54494 [ACK] Seq=2669995919 Ack=1884231612 Win=66048 Len=1248 TS
50 0.000900   192.168.2.220   192.168.1.220   TCP          66 54494 → 2388 [ACK] Seq=1884231612 Ack=2670000911 Win=54528 Len=0 TSval=3577292741 TSecr=4
51 0.000519   192.168.1.220   192.168.2.220   FTP-DATA   1314 FTP Data: 1248 bytes (PASV) (RETR file15mb)
52 0.000061   192.168.1.220   192.168.2.220   FTP-DATA   1314 FTP Data: 1248 bytes (PASV) (RETR file15mb)
53 0.000015   192.168.1.220   192.168.2.220   FTP-DATA   1314 [TCP Previous segment not captured] FTP Data: 1248 bytes (PASV) (RETR file15mb)
54 0.000015   192.168.1.220   192.168.2.220   FTP-DATA   1314 FTP Data: 1248 bytes (PASV) (RETR file15mb)
55 0.000199   192.168.2.220   192.168.1.220   TCP          66 54494 → 2388 [ACK] Seq=1884231612 Ack=2670002159 Win=57472 Len=0 TSval=3577292742 TSecr=4
56 0.000229   192.168.2.220   192.168.1.220   TCP          66 54494 → 2388 [ACK] Seq=1884231612 Ack=2670003407 Win=60288 Len=0 TSval=3577292742 TSecr=4
57 0.000183   192.168.1.220   192.168.2.220   FTP-DATA   1314 FTP Data: 1248 bytes (PASV) (RETR file15mb)
58 0.000106   192.168.2.220   192.168.1.220   TCP          78 [TCP Window Update] 54494 → 2388 [ACK] Seq=1884231612 Ack=2670003407 Win=65280 Len=0 TSval
59 0.000168   192.168.2.220   192.168.1.220   TCP          78 [TCP Window Update] 54494 → 2388 [ACK] Seq=1884231612 Ack=2670003407 Win=68224 Len=0 TSval
60 0.000000   192.168.1.220   192.168.2.220   FTP-DATA   1314 FTP Data: 1248 bytes (PASV) (RETR file15mb)
```

The CAPO capture content:



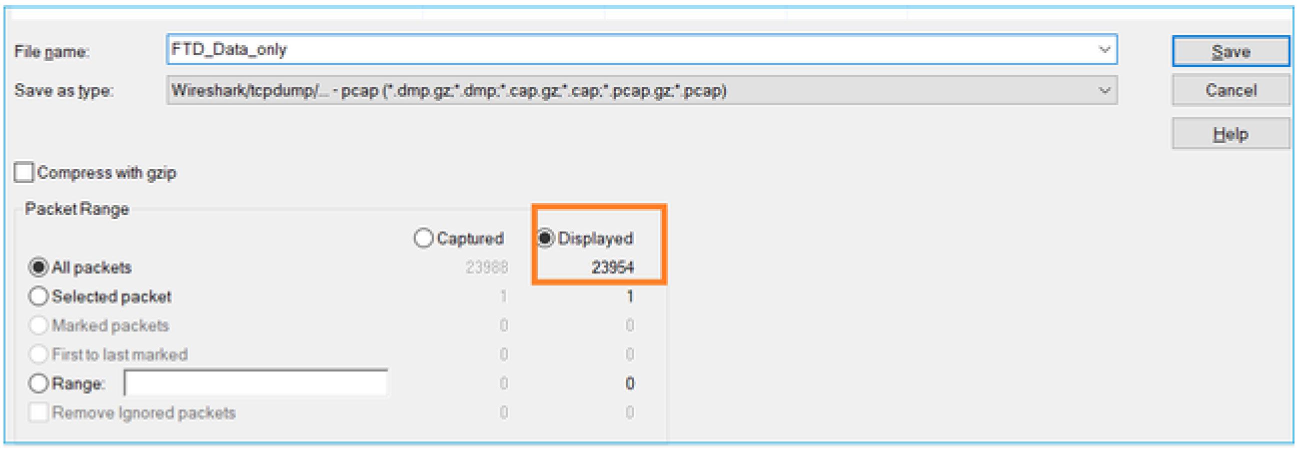
```
31 0.000000   192.168.2.220   192.168.1.220   TCP        74 54494 → 2388 [SYN] Seq=2157030681 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=3577288500 T
33 1.026534   192.168.2.220   192.168.1.220   TCP        74 [TCP Retransmission] 54494 → 2388 [SYN] Seq=2157030681 Win=29200 Len=0 MSS=1460 SACK_PERM
34 1.981400   192.168.1.220   192.168.2.220   TCP        74 2388 → 54494 [SYN, ACK] Seq=2224316911 Ack=2157030682 Win=8192 Len=0 MSS=1260 WS=256 SACK
35 0.000610   192.168.2.220   192.168.1.220   TCP        66 54494 → 2388 [ACK] Seq=2157030682 Ack=2224316912 Win=29312 Len=0 TSval=3577291508 TSecr=4
38 0.001328   192.168.1.220   192.168.2.220   FTP-DATA 1314 FTP Data: 1248 bytes (PASV) (RETR file15mb)
40 0.000641   192.168.2.220   192.168.1.220   TCP        66 54494 → 2388 [ACK] Seq=2157030682 Ack=2224318160 Win=32128 Len=0 TSval=3577291510 TSe
41 0.000381   192.168.1.220   192.168.2.220   FTP-DATA 1314 [TCP Previous segment not captured] FTP Data: 1248 bytes (PASV) (RETR file15mb)
42 0.000046   192.168.1.220   192.168.2.220   FTP-DATA 1314 FTP Data: 1248 bytes (PASV) (RETR file15mb)
43 0.000290   192.168.2.220   192.168.1.220   TCP        78 [TCP Window Update] 54494 → 2388 [ACK] Seq=2157030682 Ack=2224318160 Win=35072 Len=0 TSva
44 0.000076   192.168.2.220   192.168.1.220   TCP        78 [TCP Window Update] 54494 → 2388 [ACK] Seq=2157030682 Ack=2224318160 Win=37888 Len=0 TSva
45 0.309005   192.168.1.220   192.168.2.220   TCP      1314 [TCP Out-Of-Order] 2388 → 54494 [ACK] Seq=2224318160 Ack=2157030682 Win=66048 Len=1248 TS
46 0.000580   192.168.2.220   192.168.1.220   TCP        66 54494 → 2388 [ACK] Seq=2157030682 Ack=2224321904 Win=40832 Len=0 TSval=3577291820 TSecr=4
47 0.000412   192.168.1.220   192.168.2.220   FTP-DATA 1314 FTP Data: 1248 bytes (PASV) (RETR file15mb)
48 0.000061   192.168.1.220   192.168.2.220   FTP-DATA 1314 [TCP Previous segment not captured] FTP Data: 1248 bytes (PASV) (RETR file15mb)
49 0.000076   192.168.1.220   192.168.2.220   FTP-DATA 1314 FTP Data: 1248 bytes (PASV) (RETR file15mb)
50 0.000290   192.168.2.220   192.168.1.220   TCP        66 54494 → 2388 [ACK] Seq=2157030682 Ack=2224323152 Win=43776 Len=0 TSval=3577291821 TSecr=4
51 0.000046   192.168.2.220   192.168.1.220   TCP        78 [TCP Window Update] 54494 → 2388 [ACK] Seq=2157030682 Ack=2224323152 Win=48768 Len=0 TSva
52 0.000412   192.168.1.220   192.168.2.220   FTP-DATA 1314 FTP Data: 1248 bytes (PASV) (RETR file15mb)
53 0.000351   192.168.2.220   192.168.1.220   TCP        78 [TCP Window Update] 54494 → 2388 [ACK] Seq=2157030682 Ack=2224323152 Win=51584 Len=0 TSva
54 0.918019   192.168.1.220   192.168.2.220   TCP      1314 [TCP Out-Of-Order] 2388 → 54494 [ACK] Seq=2224323152 Ack=2157030682 Win=66048 Len=1248 TS
55 0.001007   192.168.2.220   192.168.1.220   TCP        66 54494 → 2388 [ACK] Seq=2157030682 Ack=2224328144 Win=54528 Len=0 TSval=3577292741 TSecr=4
56 0.000457   192.168.1.220   192.168.2.220   FTP-DATA 1314 FTP Data: 1248 bytes (PASV) (RETR file15mb)
57 0.000061   192.168.1.220   192.168.2.220   FTP-DATA 1314 FTP Data: 1248 bytes (PASV) (RETR file15mb)
58 0.000016   192.168.1.220   192.168.2.220   FTP-DATA 1314 [TCP Previous segment not captured] FTP Data: 1248 bytes (PASV) (RETR file15mb)
59 0.000000   192.168.1.220   192.168.2.220   FTP-DATA 1314 FTP Data: 1248 bytes (PASV) (RETR file15mb)
60 0.000274   192.168.2.220   192.168.1.220   TCP        66 54494 → 2388 [ACK] Seq=2157030682 Ack=2224329392 Win=57472 Len=0 TSval=3577292742 TSecr=4
61 0.000214   192.168.2.220   192.168.1.220   TCP        66 54494 → 2388 [ACK] Seq=2157030682 Ack=2224330640 Win=60288 Len=0 TSval=3577292742 TSecr=4
62 0.000122   192.168.1.220   192.168.2.220   FTP-DATA 1314 FTP Data: 1248 bytes (PASV) (RETR file15mb)
63 0.000168   192.168.2.220   192.168.1.220   TCP        78 [TCP Window Update] 54494 → 2388 [ACK] Seq=2157030682 Ack=2224330640 Win=65280 Len=0 TSva
64 0.000107   192.168.1.220   192.168.2.220   FTP-DATA 1314 FTP Data: 1248 bytes (PASV) (RETR file15mb)
```

Key Points:

1. There are TCP Out-Of-Order (OOO) packets.
2. There is a TCP Retransmission.
3. There is an indication of a packet loss (dropped packets).

Tip: Save the captures as you navigate to **File > Export Specified Packets**. Then save only the **Displayed** packet range

---



File name: FTD_Data_only

Save as type: Wireshark/tcpdump/... - pcap (*.dmp.gz;*.dmp;*.cap.gz;*.cap;*.pcap.gz;*.pcap)

Compress with gzip

Packet Range

	Captured	[✓ Displayed]
[✓ All packets]	23988	23954
Selected packet	1	1
Marked packets	0	0
First to last marked	0	0
Range:	0	0
Remove Ignored packets	0	0

Recommended Actions

The actions listed in this section have as a goal to further narrow down the issue.

Action 1. Identify the packet loss location.

In cases like this, you must take simultaneous captures and use the divide and conquer methodology to identify the network segment(s) that cause packet loss. From the firewall point of view there are 3 main scenarios:

1. The packet loss is caused by the firewall itself.
2. The packet loss is caused downstream to the firewall device (direction from server to client).
3. The packet loss is caused upstream to the firewall device (direction from the client to server).

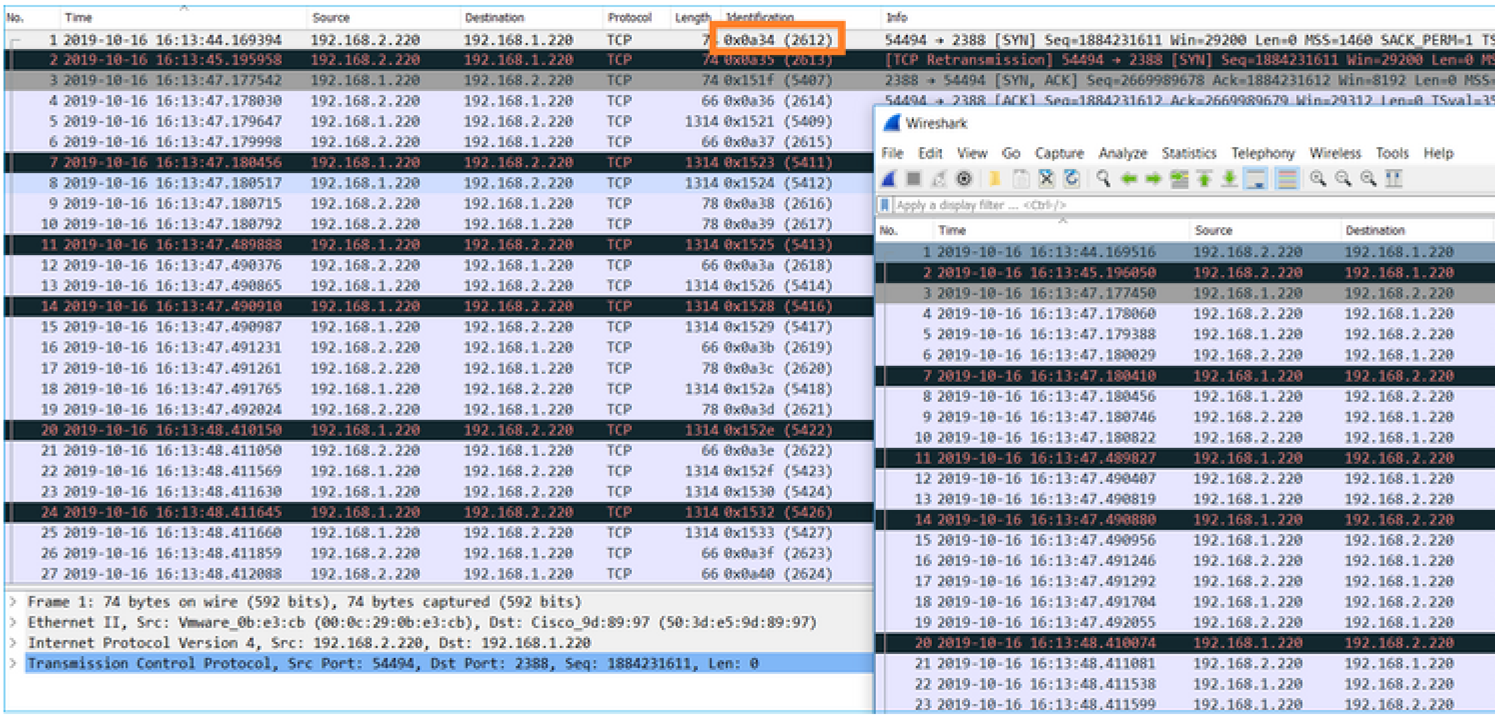
Packet loss caused by the Firewall: In order to identify if the packet loss is caused by the firewall there is a need to compare the ingress capture to the egress capture. There are quite many ways to compare 2 different captures. This section demonstrates one way to do this task.

Procedure to Compare 2 Captures in order to Identify the Packet Loss

Step 1. Ensure that the 2 captures contain packets from the same time window. This means there must be no packets in one capture that were captured before or after the other capture. There are a few ways to do this:

- Check the first and last packet IP identification (ID) values.
- Check the first and last packet timestamp values.

In this example you can see that the first packets of each capture have the same IP ID values:



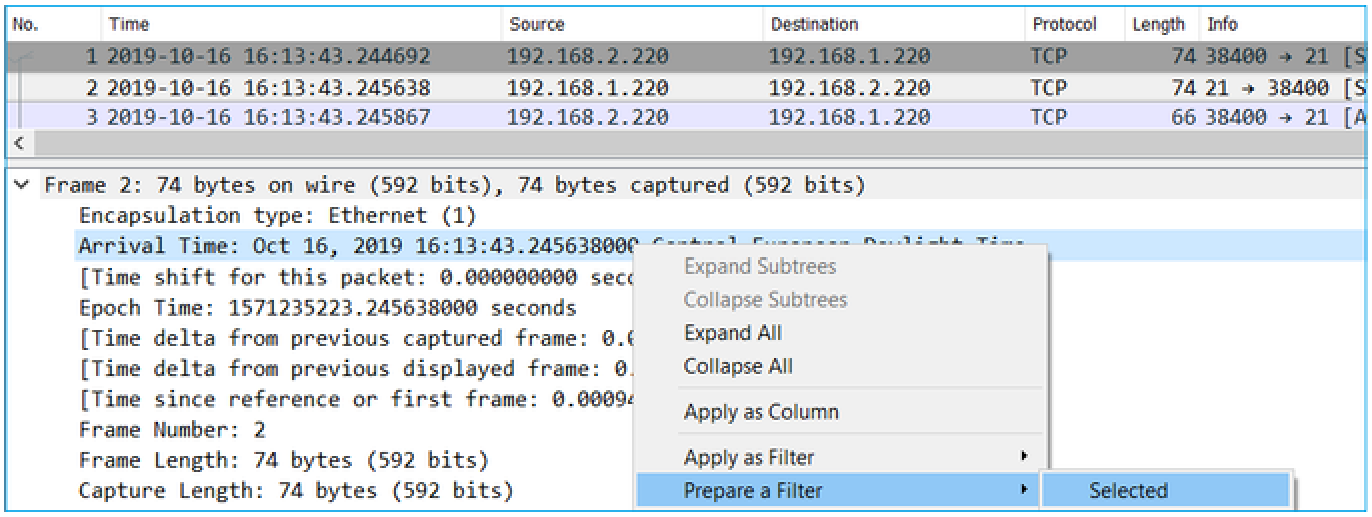
```
No.  Time                        Source         Destination    Protocol Length Identification  Info
   1 2019-10-16 16:13:44.169394  192.168.2.220  192.168.1.220  TCP      7      0x0a34 (2612)   54494 → 2388 [SYN] Seq=1884231611 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TS
   2 2019-10-16 16:13:45.195958  192.168.2.220  192.168.1.220  TCP      74     0x0a35 (2613)   [TCP Retransmission] 54494 → 2388 [SYN] Seq=1884231611 Win=29200 Len=0 MS
   3 2019-10-16 16:13:47.177542  192.168.1.220  192.168.2.220  TCP      74     0x151f (5407)   2388 → 54494 [SYN, ACK] Seq=2669989678 Ack=1884231612 Win=8192 Len=0 MSS=
   4 2019-10-16 16:13:47.178030  192.168.2.220  192.168.1.220  TCP      66     0x0a36 (2614)   54494 → 2388 [ACK] Seq=1884231612 Ack=2669989679 Win=29312 Len=0 TSval=3
   5 2019-10-16 16:13:47.179647  192.168.1.220  192.168.2.220  TCP      1314   0x1521 (5409)
   6 2019-10-16 16:13:47.179998  192.168.2.220  192.168.1.220  TCP      66     0x0a37 (2615)
   7 2019-10-16 16:13:47.180456  192.168.1.220  192.168.2.220  TCP      1314   0x1523 (5411)
   8 2019-10-16 16:13:47.180517  192.168.1.220  192.168.2.220  TCP      1314   0x1524 (5412)
   9 2019-10-16 16:13:47.180715  192.168.2.220  192.168.1.220  TCP      78     0x0a38 (2616)
  10 2019-10-16 16:13:47.180792  192.168.2.220  192.168.1.220  TCP      78     0x0a39 (2617)
  11 2019-10-16 16:13:47.489888  192.168.1.220  192.168.2.220  TCP      1314   0x1525 (5413)
  12 2019-10-16 16:13:47.490376  192.168.2.220  192.168.1.220  TCP      66     0x0a3a (2618)
  13 2019-10-16 16:13:47.490865  192.168.1.220  192.168.2.220  TCP      1314   0x1526 (5414)
  14 2019-10-16 16:13:47.490910  192.168.1.220  192.168.2.220  TCP      1314   0x1528 (5416)
  15 2019-10-16 16:13:47.490987  192.168.1.220  192.168.2.220  TCP      1314   0x1529 (5417)
  16 2019-10-16 16:13:47.491231  192.168.2.220  192.168.1.220  TCP      66     0x0a3b (2619)
  17 2019-10-16 16:13:47.491261  192.168.2.220  192.168.1.220  TCP      78     0x0a3c (2620)
  18 2019-10-16 16:13:47.491765  192.168.1.220  192.168.2.220  TCP      1314   0x152a (5418)
  19 2019-10-16 16:13:47.492024  192.168.2.220  192.168.1.220  TCP      78     0x0a3d (2621)
  20 2019-10-16 16:13:48.410150  192.168.1.220  192.168.2.220  TCP      1314   0x152e (5422)
  21 2019-10-16 16:13:48.411050  192.168.2.220  192.168.1.220  TCP      66     0x0a3e (2622)
  22 2019-10-16 16:13:48.411569  192.168.1.220  192.168.2.220  TCP      1314   0x152f (5423)
  23 2019-10-16 16:13:48.411630  192.168.1.220  192.168.2.220  TCP      1314   0x1530 (5424)
  24 2019-10-16 16:13:48.411645  192.168.1.220  192.168.2.220  TCP      1314   0x1532 (5426)
  25 2019-10-16 16:13:48.411660  192.168.1.220  192.168.2.220  TCP      1314   0x1533 (5427)
  26 2019-10-16 16:13:48.411859  192.168.2.220  192.168.1.220  TCP      66     0x0a3f (2623)
  27 2019-10-16 16:13:48.412088  192.168.2.220  192.168.1.220  TCP      66     0x0a40 (2624)

> Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
> Ethernet II, Src: Vmware_0b:e3:cb (00:0c:29:0b:e3:cb), Dst: Cisco_9d:89:97 (50:3d:e5:9d:89:97)
> Internet Protocol Version 4, Src: 192.168.2.220, Dst: 192.168.1.220
> Transmission Control Protocol, Src Port: 54494, Dst Port: 2388, Seq: 1884231611, Len: 0
```

Wireshark
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

```
No.  Time                        Source         Destination
   1 2019-10-16 16:13:44.169516  192.168.2.220  192.168.1.220
   2 2019-10-16 16:13:45.196050  192.168.2.220  192.168.1.220
   3 2019-10-16 16:13:47.177450  192.168.1.220  192.168.2.220
   4 2019-10-16 16:13:47.178060  192.168.2.220  192.168.1.220
   5 2019-10-16 16:13:47.179388  192.168.1.220  192.168.2.220
   6 2019-10-16 16:13:47.180029  192.168.2.220  192.168.1.220
   7 2019-10-16 16:13:47.180410  192.168.1.220  192.168.2.220
   8 2019-10-16 16:13:47.180456  192.168.1.220  192.168.2.220
   9 2019-10-16 16:13:47.180746  192.168.2.220  192.168.1.220
  10 2019-10-16 16:13:47.180822  192.168.2.220  192.168.1.220
  11 2019-10-16 16:13:47.489827  192.168.1.220  192.168.2.220
  12 2019-10-16 16:13:47.490407  192.168.2.220  192.168.1.220
  13 2019-10-16 16:13:47.490819  192.168.1.220  192.168.2.220
  14 2019-10-16 16:13:47.490880  192.168.1.220  192.168.2.220
  15 2019-10-16 16:13:47.490956  192.168.1.220  192.168.2.220
  16 2019-10-16 16:13:47.491246  192.168.2.220  192.168.1.220
  17 2019-10-16 16:13:47.491292  192.168.2.220  192.168.1.220
  18 2019-10-16 16:13:47.491704  192.168.1.220  192.168.2.220
  19 2019-10-16 16:13:47.492055  192.168.2.220  192.168.1.220
  20 2019-10-16 16:13:48.410074  192.168.1.220  192.168.2.220
  21 2019-10-16 16:13:48.411081  192.168.2.220  192.168.1.220
  22 2019-10-16 16:13:48.411538  192.168.1.220  192.168.2.220
  23 2019-10-16 16:13:48.411599  192.168.1.220  192.168.2.220
```

In case they are not the same then:

1. Compare the Timestamps from the first packet of each capture.
2. From the capture with the latest Timestamp get a filter from it change the Timestamp filter from == to >= (the first packet) and <= (the last packet), e.g:



```
No.  Time                          Source          Destination     Protocol  Length  Info
   1 2019-10-16 16:13:43.244692    192.168.2.220   192.168.1.220   TCP       74 38400 → 21 [S
   2 2019-10-16 16:13:43.245638    192.168.1.220   192.168.2.220   TCP       74 21 → 38400 [S
   3 2019-10-16 16:13:43.245867    192.168.2.220   192.168.1.220   TCP       66 38400 → 21 [A

Frame 2: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
    Encapsulation type: Ethernet (1)
    Arrival Time: Oct 16, 2019 16:13:43.24563800
    [Time shift for this packet: 0.000000000 seco
    Epoch Time: 1571235223.245638000 seconds
    [Time delta from previous captured frame: 0.0
    [Time delta from previous displayed frame: 0.
    [Time since reference or first frame: 0.0009
    Frame Number: 2
    Frame Length: 74 bytes (592 bits)
    Capture Length: 74 bytes (592 bits)
```

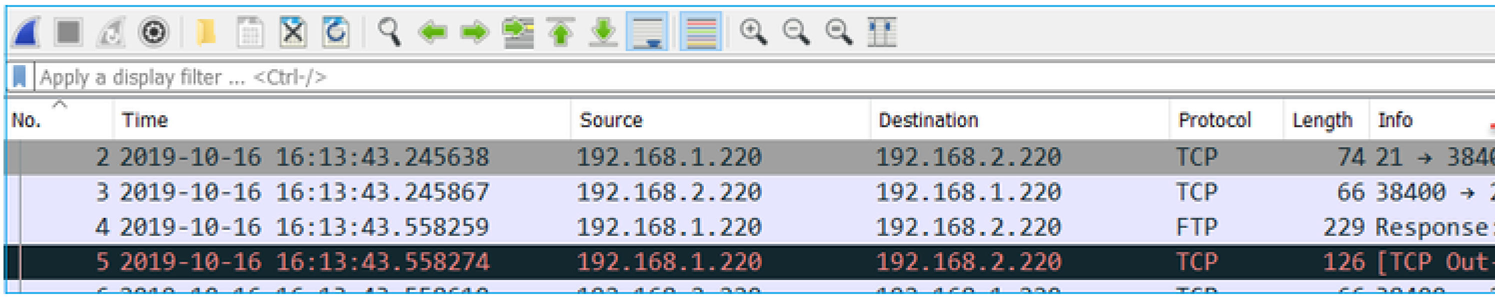
(frame.time >= "Oct 16, 2019 16:13:43.244692000") &&(frame.time <= "Oct 16, 2019 16:20:21.785130000")

3. Export the specified packets to a new capture, select **File > Export Specified Packets** and then save the **Displayed** packets. At this point, both captures must contain packets that cover the same time window. You can now start the comparison of the 2 captures.

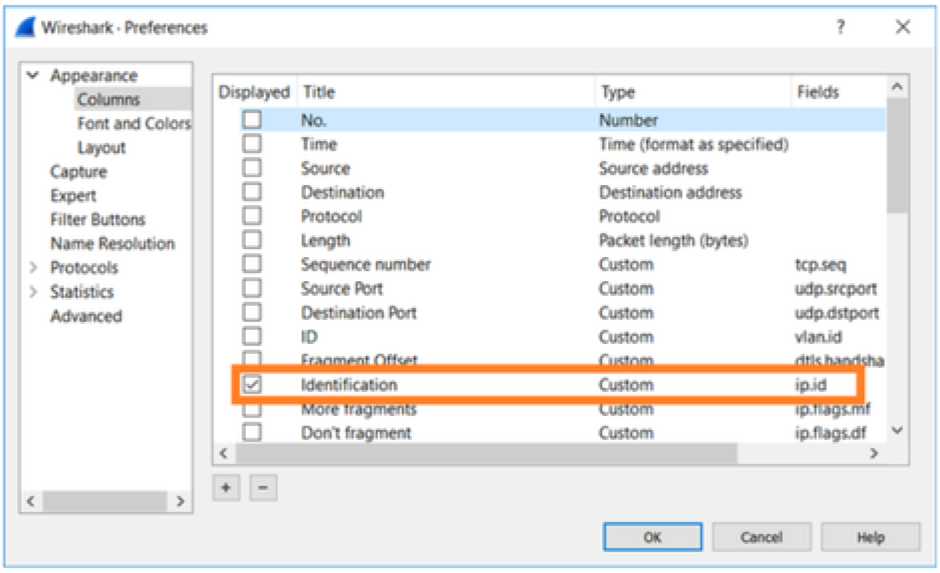
Step 2. Specify which packet field is used for the comparison between the 2 captures. Example of fields that can be used:

- IP Identification
- RTP Sequence Number
- ICMP Sequence Number

Create a text version of each capture which contains the field for each packet that you specified in step 1. In order to do this, leave only the column of interest, for example, if you want to compare packets based on IP Identification then modify the capture as shown in the image.

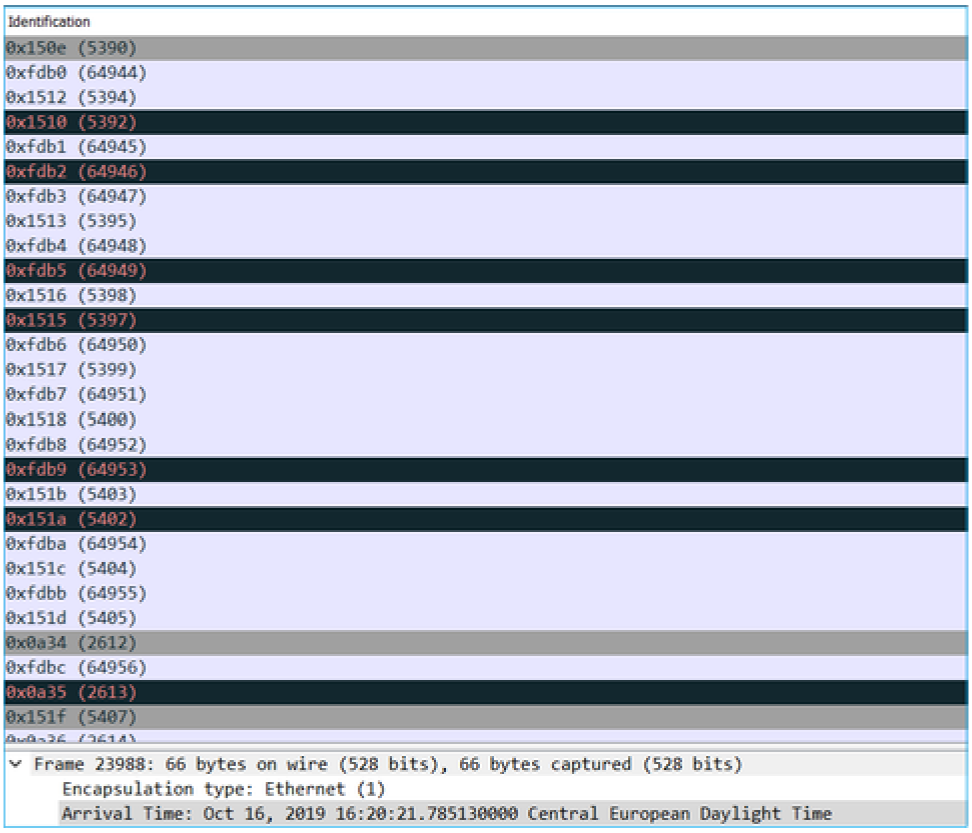


```
No.  Time                          Source          Destination     Protocol  Length  Info
   2 2019-10-16 16:13:43.245638    192.168.1.220   192.168.2.220   TCP       74 21 → 3840
   3 2019-10-16 16:13:43.245867    192.168.2.220   192.168.1.220   TCP       66 38400 → 2
   4 2019-10-16 16:13:43.558259    192.168.1.220   192.168.2.220   FTP       229 Response
   5 2019-10-16 16:13:43.558274    192.168.1.220   192.168.2.220   TCP       126 [TCP Out-
```



Displayed	Title	Type	Fields
☐	No.	Number	
☐	Time	Time (format as specified)	
☐	Source	Source address	
☐	Destination	Destination address	
☐	Protocol	Protocol	
☐	Length	Packet length (bytes)	
☐	Sequence number	Custom	tcp.seq
☐	Source Port	Custom	udp.srcport
☐	Destination Port	Custom	udp.dstport
☐	ID	Custom	vlan.id
☐	Fragment Offset	Custom	dtls.handsha
✓	Identification	Custom	ip.id
☐	More fragments	Custom	ip.flags.mf
☐	Don't fragment	Custom	ip.flags.df

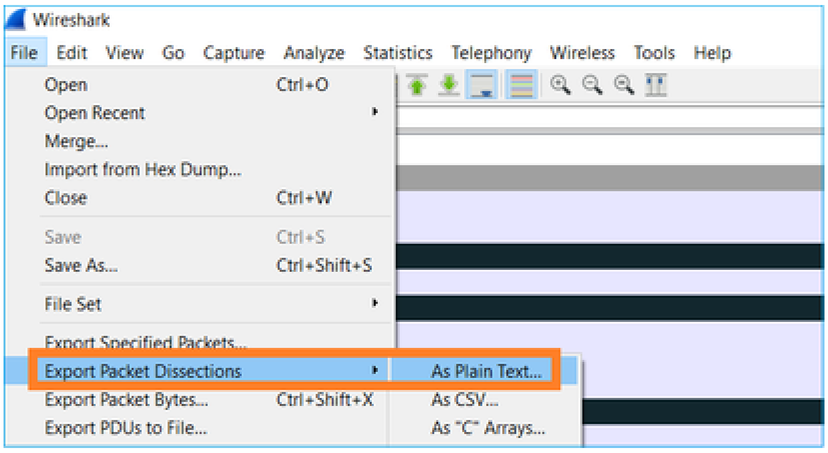
The result:



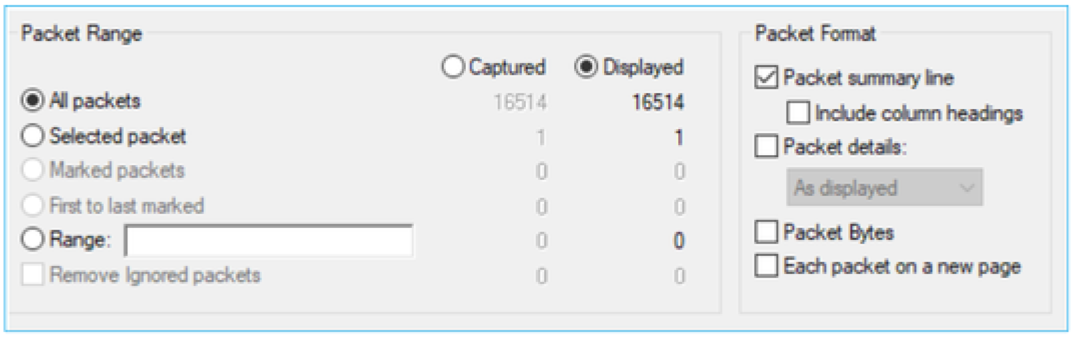
```
Identification
0x150e (5390)
0xfdb0 (64944)
0x1512 (5394)
0x1510 (5392)
0xfdb1 (64945)
0xfdb2 (64946)
0xfdb3 (64947)
0x1513 (5395)
0xfdb4 (64948)
0xfdb5 (64949)
0x1516 (5398)
0x1515 (5397)
0xfdb6 (64950)
0x1517 (5399)
0xfdb7 (64951)
0x1518 (5400)
0xfdb8 (64952)
0xfdb9 (64953)
0x151b (5403)
0x151a (5402)
0xfdba (64954)
0x151c (5404)
0xfdbb (64955)
0x151d (5405)
0x0a34 (2612)
0xfdbc (64956)
0x0a35 (2613)
0x151f (5407)
```

```
v Frame 23988: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)
    Encapsulation type: Ethernet (1)
    Arrival Time: Oct 16, 2019 16:20:21.785130000 Central European Daylight Time
```

Step 3. Create a text version of the capture (**File > Export Packet Dissections > As Plain Text**...), as shown in the image:



Uncheck the **Include column headings** and **Packet details** options to export only the values of the displayed field, as shown in the image:



Packet Range

	Captured	[✓ Displayed]
[✓ All packets]	16514	16514
Selected packet	1	1
Marked packets	0	0
First to last marked	0	0
Range:	0	0
Remove Ignored packets	0	0

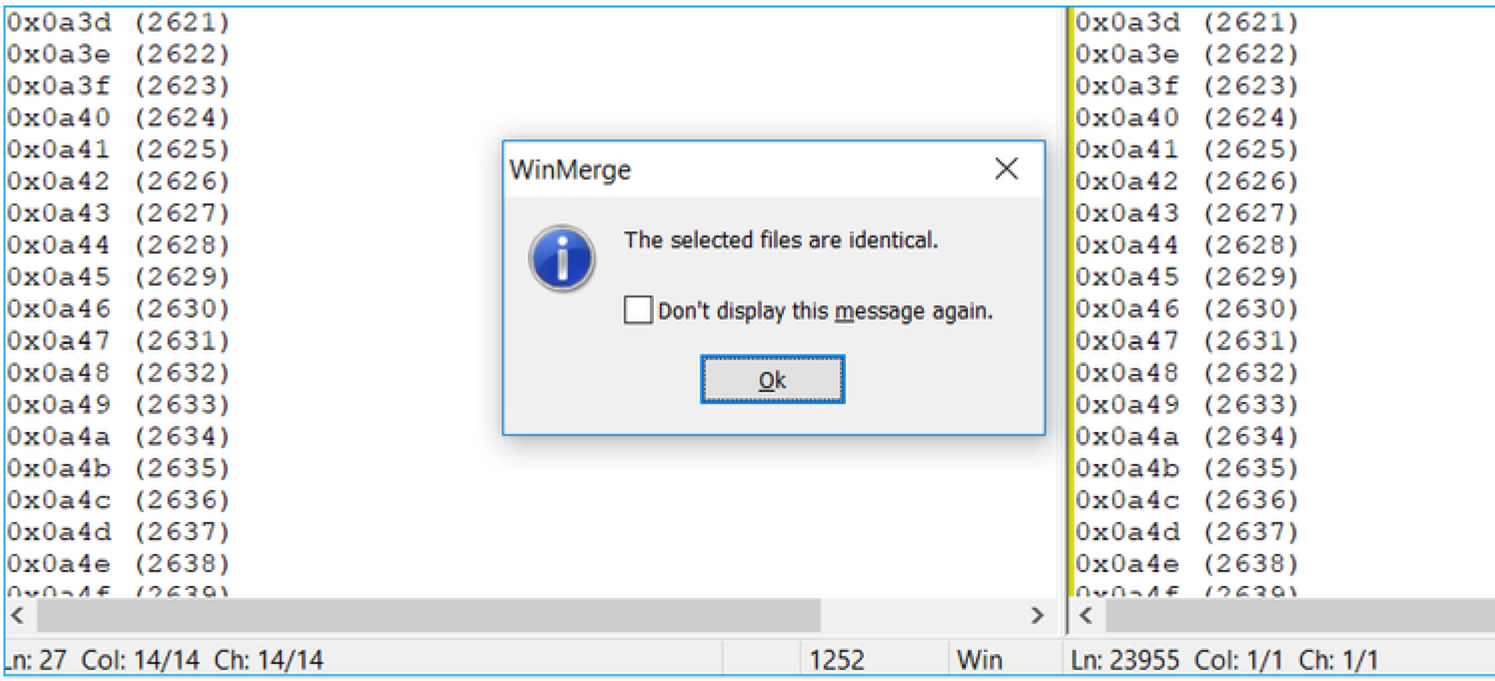
Packet Format

- [✓] Packet summary line
- [ ] Include column headings
- [ ] Packet details: As displayed
- [ ] Packet Bytes
- [ ] Each packet on a new page

Step 4. Sort the packets in the files. You can use the Linux **sort** command to do this:

```
<#root>
#
sort CAPI_IDs > file1.sorted
#
sort CAPO_IDs > file2.sorted
```

Step 5. Use a text comparison tool (for example, WinMerge) or the Linux **diff** command to find the differences between the 2 captures.



```
0x0a3d (2621)                                0x0a3d (2621)
0x0a3e (2622)                                0x0a3e (2622)
0x0a3f (2623)                                0x0a3f (2623)
0x0a40 (2624)                                0x0a40 (2624)
0x0a41 (2625)                                0x0a41 (2625)
0x0a42 (2626)                                0x0a42 (2626)
0x0a43 (2627)                                0x0a43 (2627)
0x0a44 (2628)                                0x0a44 (2628)
0x0a45 (2629)                                0x0a45 (2629)
0x0a46 (2630)                                0x0a46 (2630)
0x0a47 (2631)                                0x0a47 (2631)
0x0a48 (2632)                                0x0a48 (2632)
0x0a49 (2633)                                0x0a49 (2633)
0x0a4a (2634)                                0x0a4a (2634)
0x0a4b (2635)                                0x0a4b (2635)
0x0a4c (2636)                                0x0a4c (2636)
0x0a4d (2637)                                0x0a4d (2637)
0x0a4e (2638)                                0x0a4e (2638)
```

WinMerge

The selected files are identical.

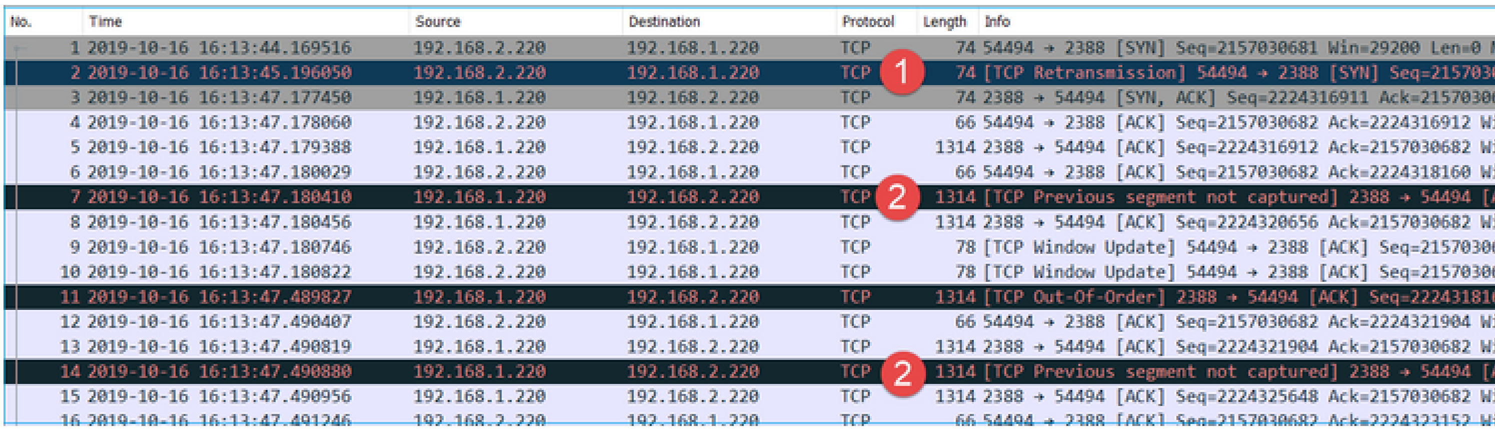
Don't display this message again.

Ok

```
Ln: 27  Col: 14/14  Ch: 14/14      1252      Win      Ln: 23955  Col: 1/1  Ch: 1/1
```

In this case, CAPI and CAPO capture for the FTP Data traffic are identical. This proves that the packet loss was not caused by the firewall.

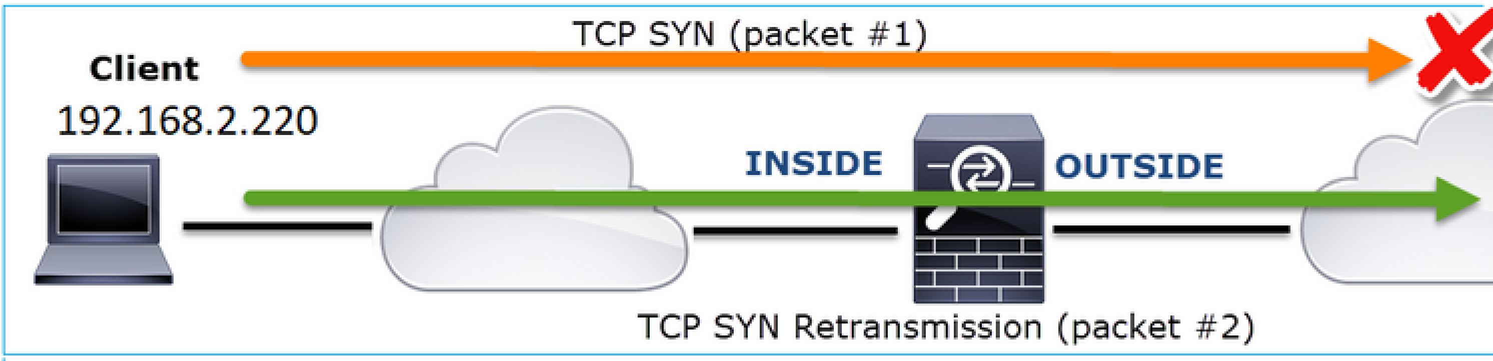
Identify upstream/downstream packet loss.



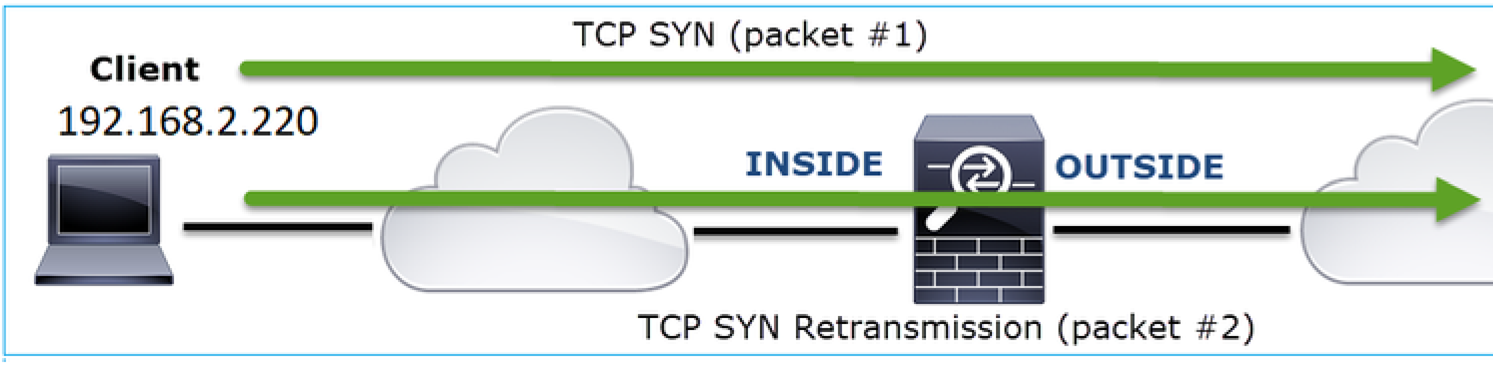
```
No.  Time                        Source         Destination    Protocol Length Info
   1 2019-10-16 16:13:44.169516  192.168.2.220  192.168.1.220  TCP        74 54494 → 2388 [SYN] Seq=2157030681 Win=29200 Len=0 M
   2 2019-10-16 16:13:45.196050  192.168.2.220  192.168.1.220  TCP        74 [TCP Retransmission] 54494 → 2388 [SYN] Seq=215703
   3 2019-10-16 16:13:47.177450  192.168.1.220  192.168.2.220  TCP        74 2388 → 54494 [SYN, ACK] Seq=2224316911 Ack=215703
   4 2019-10-16 16:13:47.178060  192.168.2.220  192.168.1.220  TCP        66 54494 → 2388 [ACK] Seq=2157030682 Ack=2224316912 W
   5 2019-10-16 16:13:47.179388  192.168.1.220  192.168.2.220  TCP      1314 2388 → 54494 [ACK] Seq=2224316912 Ack=2157030682 W
   6 2019-10-16 16:13:47.180029  192.168.2.220  192.168.1.220  TCP        66 54494 → 2388 [ACK] Seq=2157030682 Ack=2224318160 W
   7 2019-10-16 16:13:47.180410  192.168.1.220  192.168.2.220  TCP      1314 [TCP Previous segment not captured] 2388 → 54494 [
   8 2019-10-16 16:13:47.180456  192.168.1.220  192.168.2.220  TCP      1314 2388 → 54494 [ACK] Seq=2224320656 Ack=2157030682 W
   9 2019-10-16 16:13:47.180746  192.168.2.220  192.168.1.220  TCP        78 [TCP Window Update] 54494 → 2388 [ACK] Seq=215703
  10 2019-10-16 16:13:47.180822  192.168.2.220  192.168.1.220  TCP        78 [TCP Window Update] 54494 → 2388 [ACK] Seq=215703
  11 2019-10-16 16:13:47.489827  192.168.1.220  192.168.2.220  TCP      1314 [TCP Out-Of-Order] 2388 → 54494 [ACK] Seq=22243181
  12 2019-10-16 16:13:47.490407  192.168.2.220  192.168.1.220  TCP        66 54494 → 2388 [ACK] Seq=2157030682 Ack=2224321904 W
  13 2019-10-16 16:13:47.490819  192.168.1.220  192.168.2.220  TCP      1314 2388 → 54494 [ACK] Seq=2224321904 Ack=2157030682 W
  14 2019-10-16 16:13:47.490880  192.168.1.220  192.168.2.220  TCP      1314 [TCP Previous segment not captured] 2388 → 54494 [
  15 2019-10-16 16:13:47.490956  192.168.1.220  192.168.2.220  TCP      1314 2388 → 54494 [ACK] Seq=2224325648 Ack=2157030682 W
  16 2019-10-16 16:13:47.491246  192.168.2.220  192.168.1.220  TCP        66 54494 → 2388 [ACK] Seq=2157030682 Ack=2224323152 W
```

Key Points:

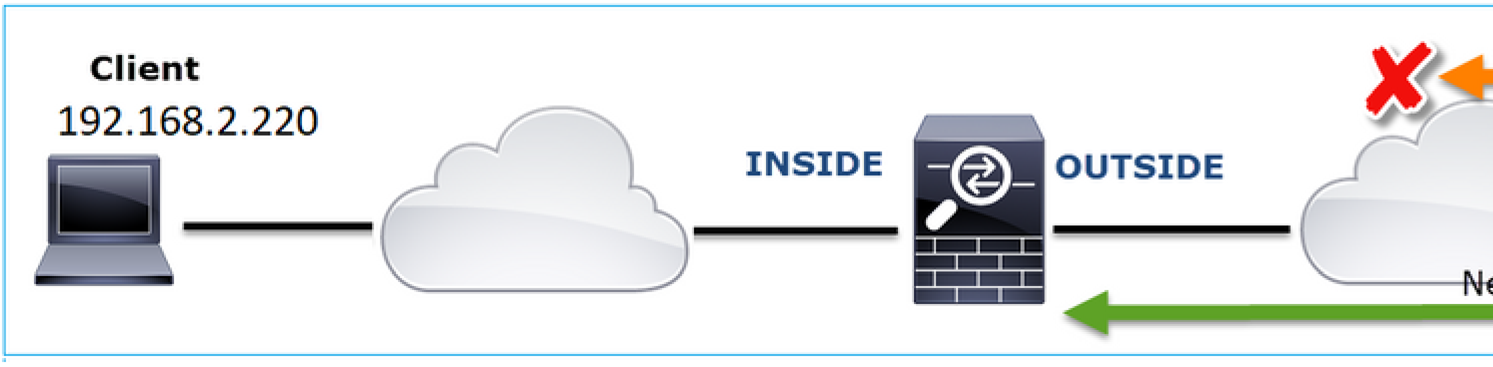
1. This packet is a TCP Retransmission. Specifically, it is a TCP SYN packet sent from the client to the server for FTP Data in Passive Mode. Since the client resends the packet and you can see the initial SYN (packet #1) the packet was lost upstream to the firewall.



In this case, there is the possiblity that the SYN packet made it to the server, but the SYN/ACK packet was lost on the way back:



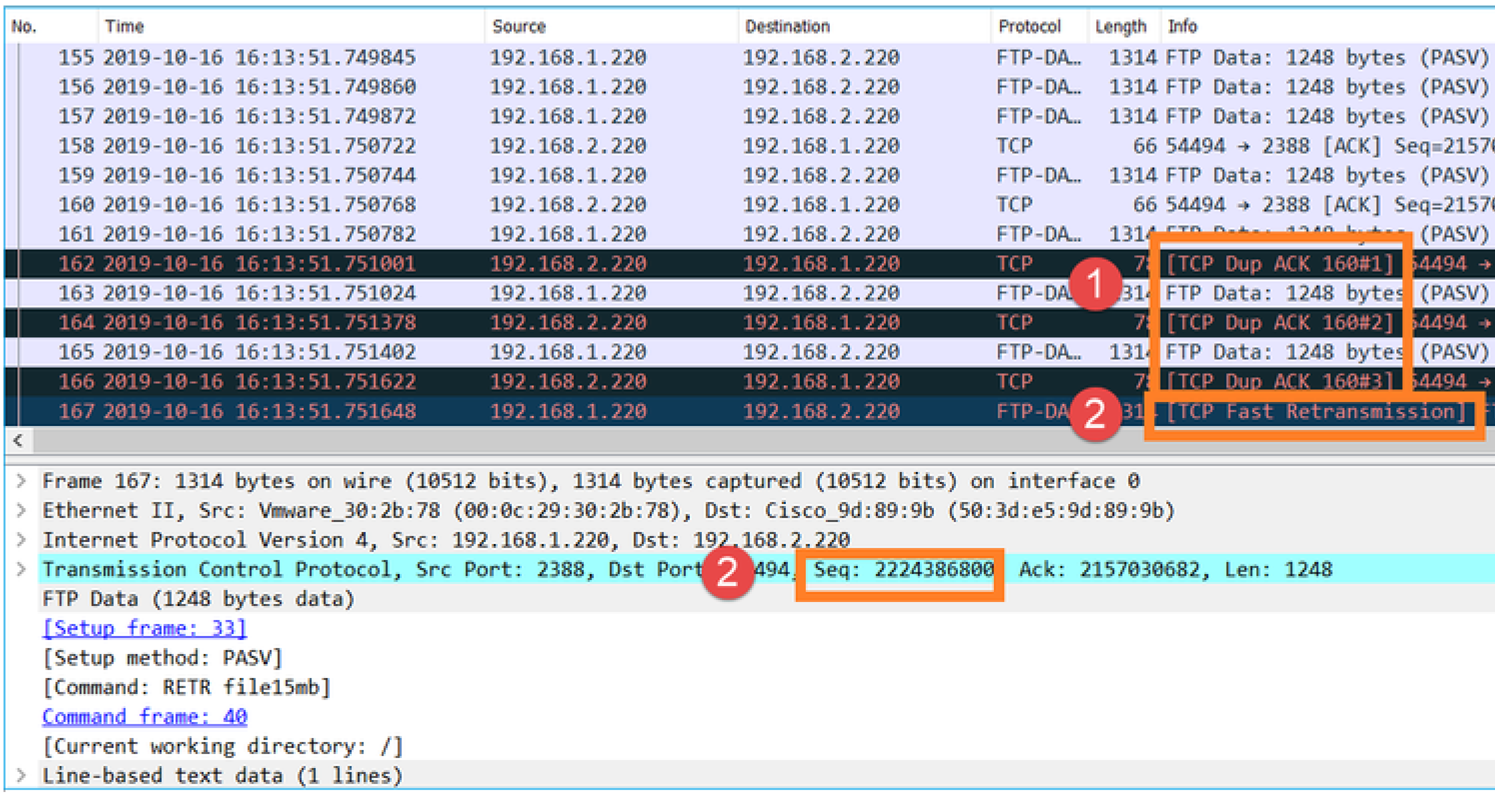
2. There is a packet from the server and Wireshark identified that the previous segment was not seen/captured. Since the non-captured packet was sent from the server to the client and was not seen in the firewall capture that means the packet was lost between the server and the firewall.



This indicates that there is packet loss between the FTP server and the firewall.

Action 2. Take Additional Captures.

Take additional captures along with captures at the endpoints. Try to apply the divide and conquer method to isolate further the problematic segment that causes the packet loss.



```
No.  Time                        Source          Destination     Protocol  Length  Info
155 2019-10-16 16:13:51.749845  192.168.1.220   192.168.2.220   FTP-DA...   1314 FTP Data: 1248 bytes (PASV)
156 2019-10-16 16:13:51.749860  192.168.1.220   192.168.2.220   FTP-DA...   1314 FTP Data: 1248 bytes (PASV)
157 2019-10-16 16:13:51.749872  192.168.1.220   192.168.2.220   FTP-DA...   1314 FTP Data: 1248 bytes (PASV)
158 2019-10-16 16:13:51.750722  192.168.2.220   192.168.1.220   TCP           66 54494 → 2388 [ACK] Seq=2157
159 2019-10-16 16:13:51.750744  192.168.1.220   192.168.2.220   FTP-DA...   1314 FTP Data: 1248 bytes (PASV)
160 2019-10-16 16:13:51.750768  192.168.2.220   192.168.1.220   TCP           66 54494 → 2388 [ACK] Seq=2157
161 2019-10-16 16:13:51.750782  192.168.1.220   192.168.2.220   FTP-DA...   1314 FTP Data: 1248 bytes (PASV)
162 2019-10-16 16:13:51.751001  192.168.2.220   192.168.1.220   TCP           7  [TCP Dup ACK 160#1] 4494 →
163 2019-10-16 16:13:51.751024  192.168.1.220   192.168.2.220   FTP-DA      131  FTP Data: 1248 bytes (PASV)
164 2019-10-16 16:13:51.751378  192.168.2.220   192.168.1.220   TCP           7  [TCP Dup ACK 160#2] 4494 →
165 2019-10-16 16:13:51.751402  192.168.1.220   192.168.2.220   FTP-DA...   131  FTP Data: 1248 bytes (PASV)
166 2019-10-16 16:13:51.751622  192.168.2.220   192.168.1.220   TCP           7  [TCP Dup ACK 160#3] 4494 →
167 2019-10-16 16:13:51.751648  192.168.1.220   192.168.2.220   FTP-DA       31  [TCP Fast Retransmission]
```

```
> Frame 167: 1314 bytes on wire (10512 bits), 1314 bytes captured (10512 bits) on interface 0
> Ethernet II, Src: Vmware_30:2b:78 (00:0c:29:30:2b:78), Dst: Cisco_9d:89:9b (50:3d:e5:9d:89:9b)
> Internet Protocol Version 4, Src: 192.168.1.220, Dst: 192.168.2.220
> Transmission Control Protocol, Src Port: 2388, Dst Port 494, Seq: 2224386800  Ack: 2157030682, Len: 1248
  FTP Data (1248 bytes data)
  [Setup frame: 33]
  [Setup method: PASV]
  [Command: RETR file15mb]
  Command frame: 40
  [Current working directory: /]
> Line-based text data (1 lines)
```

Key Points:

1. The receiver (the FTP client in this case) tracks the incoming TCP sequence numbers. If it detects that a packet was missed (an expected sequence number was skipped) then it generates an ACK packet with the ACK='expected sequence number that was skipped'. In this example the Ack=2224386800.
2. The Dup ACK triggers a TCP Fast Retransmission (retransmission within 20 msec after a Duplicate ACK is received).

What do Duplicate ACKs mean?

- A few duplicate ACKs but no actual retransmissions indicate that more likely there are packets that arrive out of order.
- Duplicate ACKs followed by actual retransmissions indicate that there is some amount of packet loss.

Action 3. Calculate the firewall processing time for transit packets.

Apply the same capture on 2 different interfaces:

<#root>

```
firepower#

capture CAPI buffer 33554432 interface INSIDE match tcp host 192.168.2.220 host 192.168.1.220


firepower#

capture CAPI interface OUTSIDE
```

Export the capture check the time difference between ingress vs egress packets

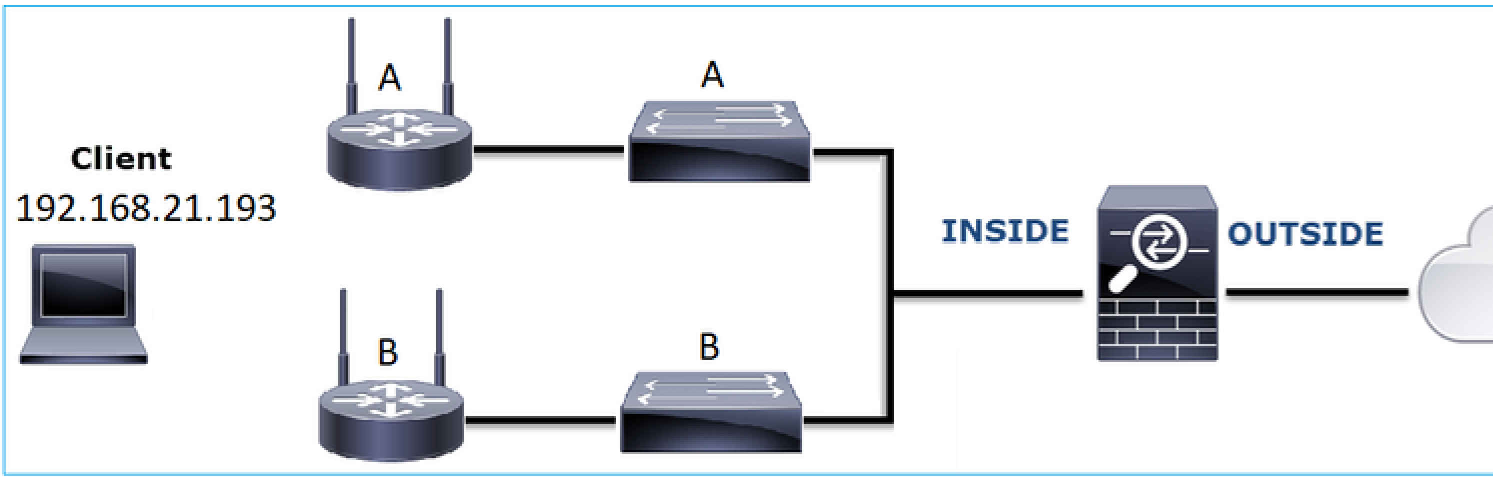
Case 7. TCP Connectivity Problem (Packet Corruption)

Problem Description:

Wireless client (192.168.21.193) tries to connect to a destination server (192.168.14.250 - HTTP) and there are 2 different scenarios:

- When the client connects to Access Point (AP) 'A' then the HTTP connection does not work.
- When the client connects to Access Point (AP) 'B' then the HTTP connection works.

This image shows the topology:



Affected Flow:

Src IP: 192.168.21.193

Dst IP: 192.168.14.250

Protocol: TCP 80

Capture Analysis

Enable captures on FTD LINA engine:

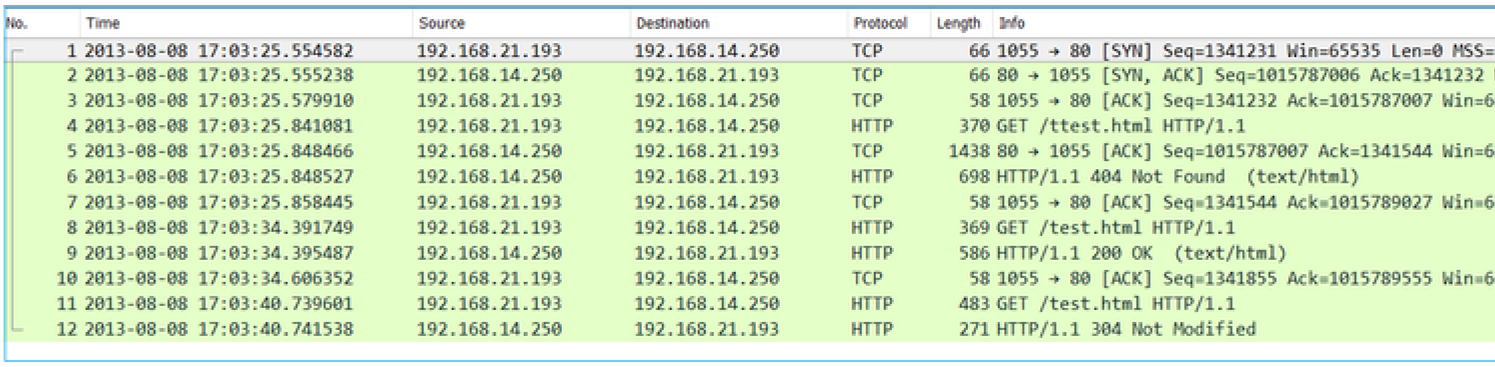
<#root>

```
firepower#

capture CAPI int INSIDE match ip host 192.168.21.193 host 192.168.14.250

firepower#

capture CAPO int OUTSIDE match ip host 192.168.21.193 host 192.168.14.250
```

Captures - Functional Scenario:

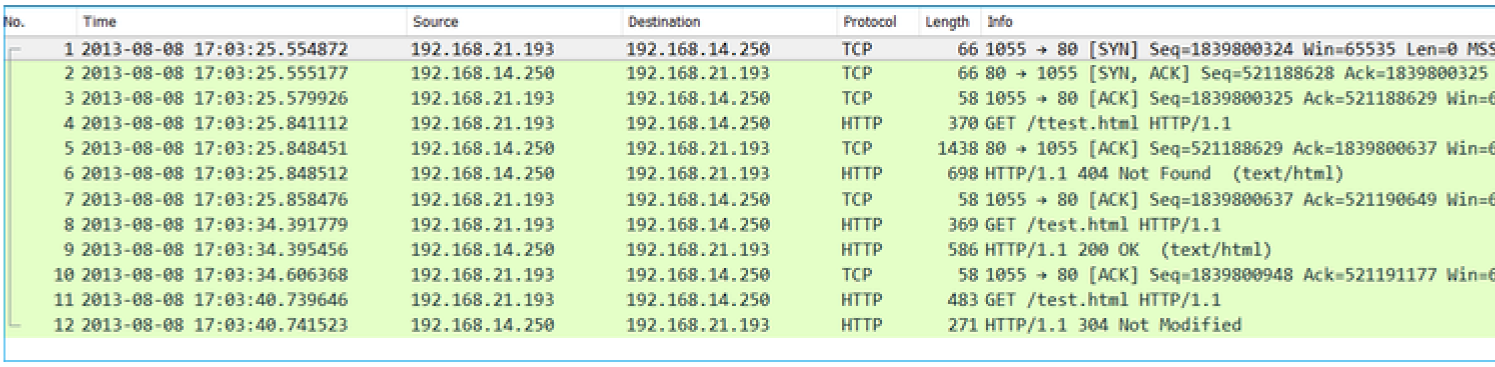
As a baseline, it is always very useful to have captures from a known-good scenario.

This image shows the capture taken on NGFW INSIDE interface



```
No.  Time                        Source           Destination      Protocol Length Info
   1 2013-08-08 17:03:25.554582  192.168.21.193   192.168.14.250   TCP        66 1055 → 80 [SYN] Seq=1341231 Win=65535 Len=0 MSS=
   2 2013-08-08 17:03:25.555238  192.168.14.250   192.168.21.193   TCP        66 80 → 1055 [SYN, ACK] Seq=1015787006 Ack=1341232 
   3 2013-08-08 17:03:25.579910  192.168.21.193   192.168.14.250   TCP        58 1055 → 80 [ACK] Seq=1341232 Ack=1015787007 Win=6
   4 2013-08-08 17:03:25.841081  192.168.21.193   192.168.14.250   HTTP      370 GET /ttest.html HTTP/1.1
   5 2013-08-08 17:03:25.848466  192.168.14.250   192.168.21.193   TCP      1438 80 → 1055 [ACK] Seq=1015787007 Ack=1341544 Win=6
   6 2013-08-08 17:03:25.848527  192.168.14.250   192.168.21.193   HTTP      698 HTTP/1.1 404 Not Found  (text/html)
   7 2013-08-08 17:03:25.858445  192.168.21.193   192.168.14.250   TCP        58 1055 → 80 [ACK] Seq=1341544 Ack=1015789027 Win=6
   8 2013-08-08 17:03:34.391749  192.168.21.193   192.168.14.250   HTTP      369 GET /test.html HTTP/1.1
   9 2013-08-08 17:03:34.395487  192.168.14.250   192.168.21.193   HTTP      586 HTTP/1.1 200 OK  (text/html)
  10 2013-08-08 17:03:34.606352  192.168.21.193   192.168.14.250   TCP        58 1055 → 80 [ACK] Seq=1341855 Ack=1015789555 Win=6
  11 2013-08-08 17:03:40.739601  192.168.21.193   192.168.14.250   HTTP      483 GET /test.html HTTP/1.1
  12 2013-08-08 17:03:40.741538  192.168.14.250   192.168.21.193   HTTP      271 HTTP/1.1 304 Not Modified
```

This image shows the capture taken on NGFW OUTSIDE interface.



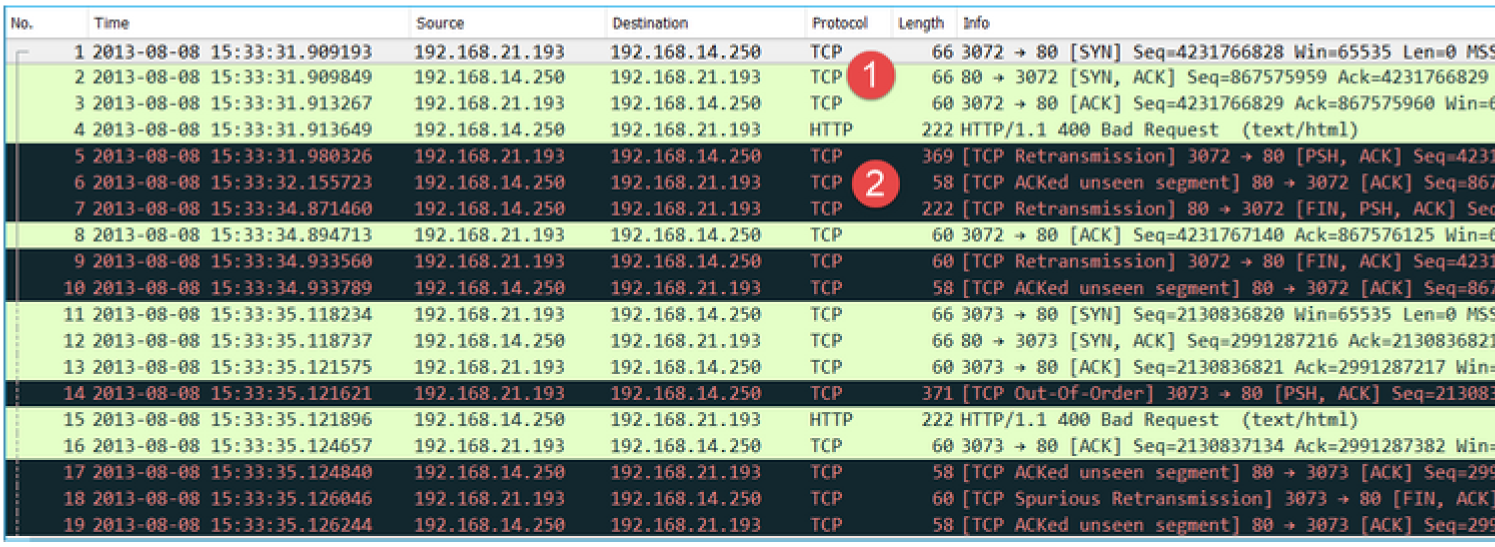
```
No.  Time                        Source           Destination      Protocol Length Info
   1 2013-08-08 17:03:25.554872  192.168.21.193   192.168.14.250   TCP        66 1055 → 80 [SYN] Seq=1839800324 Win=65535 Len=0 MSS
   2 2013-08-08 17:03:25.555177  192.168.14.250   192.168.21.193   TCP        66 80 → 1055 [SYN, ACK] Seq=521188628 Ack=1839800325
   3 2013-08-08 17:03:25.579926  192.168.21.193   192.168.14.250   TCP        58 1055 → 80 [ACK] Seq=1839800325 Ack=521188629 Win=6
   4 2013-08-08 17:03:25.841112  192.168.21.193   192.168.14.250   HTTP      370 GET /ttest.html HTTP/1.1
   5 2013-08-08 17:03:25.848451  192.168.14.250   192.168.21.193   TCP      1438 80 → 1055 [ACK] Seq=521188629 Ack=1839800637 Win=6
   6 2013-08-08 17:03:25.848512  192.168.14.250   192.168.21.193   HTTP      698 HTTP/1.1 404 Not Found  (text/html)
   7 2013-08-08 17:03:25.858476  192.168.21.193   192.168.14.250   TCP        58 1055 → 80 [ACK] Seq=1839800637 Ack=521190649 Win=6
   8 2013-08-08 17:03:34.391779  192.168.21.193   192.168.14.250   HTTP      369 GET /test.html HTTP/1.1
   9 2013-08-08 17:03:34.395456  192.168.14.250   192.168.21.193   HTTP      586 HTTP/1.1 200 OK  (text/html)
  10 2013-08-08 17:03:34.606368  192.168.21.193   192.168.14.250   TCP        58 1055 → 80 [ACK] Seq=1839800948 Ack=521191177 Win=6
  11 2013-08-08 17:03:40.739646  192.168.21.193   192.168.14.250   HTTP      483 GET /test.html HTTP/1.1
  12 2013-08-08 17:03:40.741523  192.168.14.250   192.168.21.193   HTTP      271 HTTP/1.1 304 Not Modified
```

Key Points:

1. The 2 captures are almost identical (consider the ISN randomization).
2. There are no indications of a packet loss.
3. No Out-Of-Order (OOO) packets
4. There are 3 HTTP GET Requests. The first one gets a 404 â€˜Not Foundâ€™, the second one gets a 200 â€˜OKâ€™ and the third one gets a 304 â€˜Not Modifiedâ€™ redirection message.

Captures - Known-faulty Scenario:

The ingress capture (CAPI) contents.

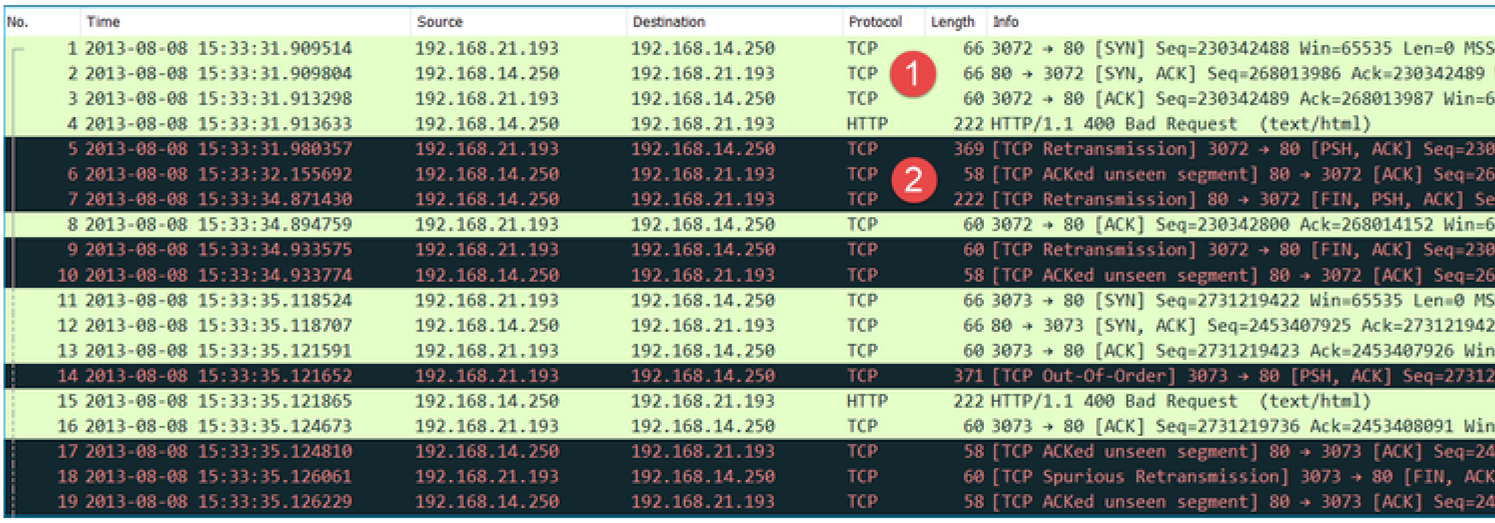


```
No.  Time                        Source          Destination     Protocol Length Info
  1 2013-08-08 15:33:31.909193  192.168.21.193  192.168.14.250  TCP        66 3072 → 80 [SYN] Seq=4231766828 Win=65535 Len=0 MSS
  2 2013-08-08 15:33:31.909849  192.168.14.250  192.168.21.193  TCP        66 80 → 3072 [SYN, ACK] Seq=867575959 Ack=4231766829
  3 2013-08-08 15:33:31.913267  192.168.21.193  192.168.14.250  TCP        60 3072 → 80 [ACK] Seq=4231766829 Ack=867575960 Win=6
  4 2013-08-08 15:33:31.913649  192.168.14.250  192.168.21.193  HTTP      222 HTTP/1.1 400 Bad Request  (text/html)
  5 2013-08-08 15:33:31.980326  192.168.21.193  192.168.14.250  TCP       369 [TCP Retransmission] 3072 → 80 [PSH, ACK] Seq=4231
  6 2013-08-08 15:33:32.155723  192.168.14.250  192.168.21.193  TCP        58 [TCP ACKed unseen segment] 80 → 3072 [ACK] Seq=867
  7 2013-08-08 15:33:34.871460  192.168.14.250  192.168.21.193  TCP       222 [TCP Retransmission] 80 → 3072 [FIN, PSH, ACK] Seq
  8 2013-08-08 15:33:34.894713  192.168.21.193  192.168.14.250  TCP        60 3072 → 80 [ACK] Seq=4231767140 Ack=867576125 Win=6
  9 2013-08-08 15:33:34.933560  192.168.21.193  192.168.14.250  TCP        60 [TCP Retransmission] 3072 → 80 [FIN, ACK] Seq=4231
 10 2013-08-08 15:33:34.933789  192.168.14.250  192.168.21.193  TCP        58 [TCP ACKed unseen segment] 80 → 3072 [ACK] Seq=867
 11 2013-08-08 15:33:35.118234  192.168.21.193  192.168.14.250  TCP        66 3073 → 80 [SYN] Seq=2130836820 Win=65535 Len=0 MSS
 12 2013-08-08 15:33:35.118737  192.168.14.250  192.168.21.193  TCP        66 80 → 3073 [SYN, ACK] Seq=2991287216 Ack=2130836821
 13 2013-08-08 15:33:35.121575  192.168.21.193  192.168.14.250  TCP        60 3073 → 80 [ACK] Seq=2130836821 Ack=2991287217 Win=
 14 2013-08-08 15:33:35.121621  192.168.21.193  192.168.14.250  TCP       371 [TCP Out-Of-Order] 3073 → 80 [PSH, ACK] Seq=213083
 15 2013-08-08 15:33:35.121896  192.168.14.250  192.168.21.193  HTTP      222 HTTP/1.1 400 Bad Request  (text/html)
 16 2013-08-08 15:33:35.124657  192.168.21.193  192.168.14.250  TCP        60 3073 → 80 [ACK] Seq=2130837134 Ack=2991287382 Win=
 17 2013-08-08 15:33:35.124840  192.168.14.250  192.168.21.193  TCP        58 [TCP ACKed unseen segment] 80 → 3073 [ACK] Seq=299
 18 2013-08-08 15:33:35.126046  192.168.21.193  192.168.14.250  TCP        60 [TCP Spurious Retransmission] 3073 → 80 [FIN, ACK
 19 2013-08-08 15:33:35.126244  192.168.14.250  192.168.21.193  TCP        58 [TCP ACKed unseen segment] 80 → 3073 [ACK] Seq=299
```

Key Points:

1. There is a TCP 3-way handshake.
2. There are TCP retransmissions and indications of a packet loss.
3. There is a packet (TCP ACK) that is identified by Wireshark as **Malformed**.

This image shows the egress capture (CAPO) contents.



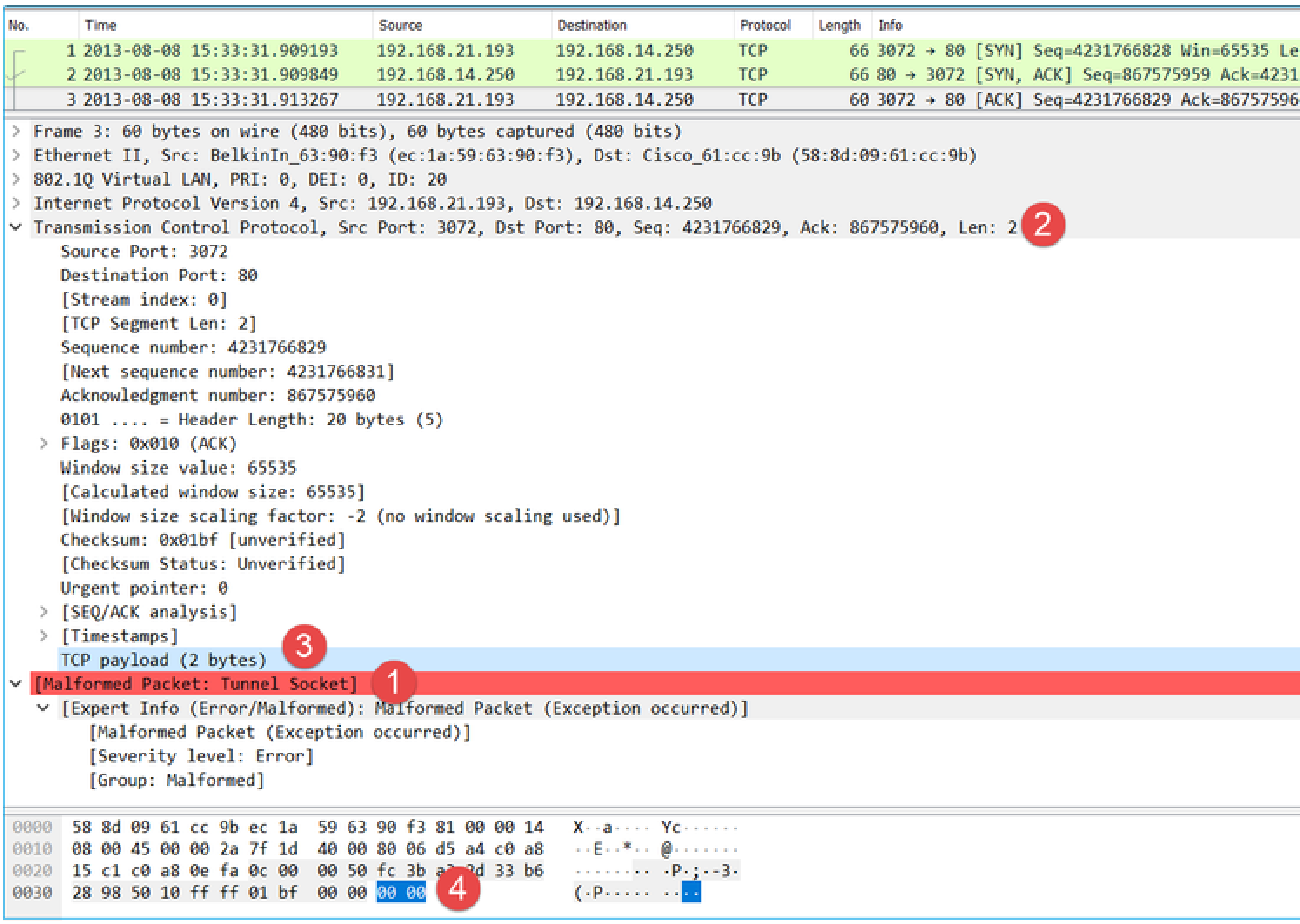
```
No.  Time                        Source          Destination     Protocol Length Info
  1 2013-08-08 15:33:31.909514  192.168.21.193  192.168.14.250  TCP        66 3072 → 80 [SYN] Seq=230342488 Win=65535 Len=0 MSS
  2 2013-08-08 15:33:31.909804  192.168.14.250  192.168.21.193  TCP        66 80 → 3072 [SYN, ACK] Seq=268013986 Ack=230342489
  3 2013-08-08 15:33:31.913298  192.168.21.193  192.168.14.250  TCP        60 3072 → 80 [ACK] Seq=230342489 Ack=268013987 Win=6
  4 2013-08-08 15:33:31.913633  192.168.14.250  192.168.21.193  HTTP      222 HTTP/1.1 400 Bad Request  (text/html)
  5 2013-08-08 15:33:31.980357  192.168.21.193  192.168.14.250  TCP       369 [TCP Retransmission] 3072 → 80 [PSH, ACK] Seq=230
  6 2013-08-08 15:33:32.155692  192.168.14.250  192.168.21.193  TCP        58 [TCP ACKed unseen segment] 80 → 3072 [ACK] Seq=26
  7 2013-08-08 15:33:34.871430  192.168.14.250  192.168.21.193  TCP       222 [TCP Retransmission] 80 → 3072 [FIN, PSH, ACK] Se
  8 2013-08-08 15:33:34.894759  192.168.21.193  192.168.14.250  TCP        60 3072 → 80 [ACK] Seq=230342800 Ack=268014152 Win=6
  9 2013-08-08 15:33:34.933575  192.168.21.193  192.168.14.250  TCP        60 [TCP Retransmission] 3072 → 80 [FIN, ACK] Seq=230
 10 2013-08-08 15:33:34.933774  192.168.14.250  192.168.21.193  TCP        58 [TCP ACKed unseen segment] 80 → 3072 [ACK] Seq=26
 11 2013-08-08 15:33:35.118524  192.168.21.193  192.168.14.250  TCP        66 3073 → 80 [SYN] Seq=2731219422 Win=65535 Len=0 MS
 12 2013-08-08 15:33:35.118707  192.168.14.250  192.168.21.193  TCP        66 80 → 3073 [SYN, ACK] Seq=2453407925 Ack=273121942
 13 2013-08-08 15:33:35.121591  192.168.21.193  192.168.14.250  TCP        60 3073 → 80 [ACK] Seq=2731219423 Ack=2453407926 Win
 14 2013-08-08 15:33:35.121652  192.168.21.193  192.168.14.250  TCP       371 [TCP Out-Of-Order] 3073 → 80 [PSH, ACK] Seq=27312
 15 2013-08-08 15:33:35.121865  192.168.14.250  192.168.21.193  HTTP      222 HTTP/1.1 400 Bad Request  (text/html)
 16 2013-08-08 15:33:35.124673  192.168.21.193  192.168.14.250  TCP        60 3073 → 80 [ACK] Seq=2731219736 Ack=2453408091 Win
 17 2013-08-08 15:33:35.124810  192.168.14.250  192.168.21.193  TCP        58 [TCP ACKed unseen segment] 80 → 3073 [ACK] Seq=24
 18 2013-08-08 15:33:35.126061  192.168.21.193  192.168.14.250  TCP        60 [TCP Spurious Retransmission] 3073 → 80 [FIN, ACK
 19 2013-08-08 15:33:35.126229  192.168.14.250  192.168.21.193  TCP        58 [TCP ACKed unseen segment] 80 → 3073 [ACK] Seq=24
```

Key Points:

The 2 captures are almost identical (consider the ISN randomization):

1. There is a TCP 3-way handshake.
2. There are TCP retransmissions and indications of a packet loss.
3. There is a packet (TCP ACK) that is identified by Wireshark as **Malformed**.

Check the malformed packet:



```
No.  Time                        Source          Destination     Protocol  Length  Info
   1 2013-08-08 15:33:31.909193  192.168.21.193  192.168.14.250  TCP           66 3072 → 80 [SYN] Seq=4231766828 Win=65535 Le
   2 2013-08-08 15:33:31.909849  192.168.14.250  192.168.21.193  TCP           66 80 → 3072 [SYN, ACK] Seq=867575959 Ack=4231
   3 2013-08-08 15:33:31.913267  192.168.21.193  192.168.14.250  TCP           60 3072 → 80 [ACK] Seq=4231766829 Ack=86757596

> Frame 3: 60 bytes on wire (480 bits), 60 bytes captured (480 bits)
> Ethernet II, Src: BelkinIn_63:90:f3 (ec:1a:59:63:90:f3), Dst: Cisco_61:cc:9b (58:8d:09:61:cc:9b)
> 802.1Q Virtual LAN, PRI: 0, DEI: 0, ID: 20
> Internet Protocol Version 4, Src: 192.168.21.193, Dst: 192.168.14.250
v Transmission Control Protocol, Src Port: 3072, Dst Port: 80, Seq: 4231766829, Ack: 867575960, Len: 2
    Source Port: 3072
    Destination Port: 80
    [Stream index: 0]
    [TCP Segment Len: 2]
    Sequence number: 4231766829
    [Next sequence number: 4231766831]
    Acknowledgment number: 867575960
    0101 .... = Header Length: 20 bytes (5)
  > Flags: 0x010 (ACK)
    Window size value: 65535
    [Calculated window size: 65535]
    [Window size scaling factor: -2 (no window scaling used)]
    Checksum: 0x01bf [unverified]
    [Checksum Status: Unverified]
    Urgent pointer: 0
  > [SEQ/ACK analysis]
  > [Timestamps]
    TCP payload (2 bytes)
v [Malformed Packet: Tunnel Socket]
  v [Expert Info (Error/Malformed): Malformed Packet (Exception occurred)]
      [Malformed Packet (Exception occurred)]
      [Severity level: Error]
      [Group: Malformed]

0000  58 8d 09 61 cc 9b ec 1a  59 63 90 f3 81 00 00 14   X··a···· Yc······
0010  08 00 45 00 00 2a 7f 1d  40 00 80 06 d5 a4 c0 a8   ··E··*·· @·······
0020  15 c1 c0 a8 0e fa 0c 00  00 50 fc 3b a2 2d 33 b6   ········ ·P·;·-3·
0030  28 98 50 10 ff ff 01 bf  00 00 00 00               (·P····· ····
```

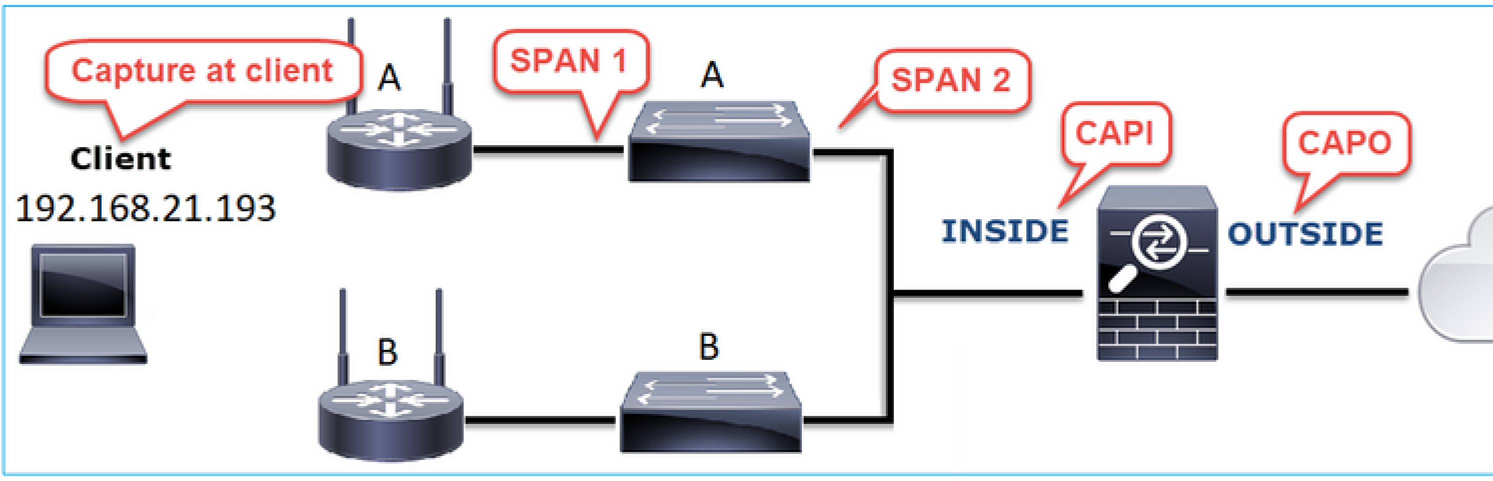
Key Points:

1. The packet is identified as a Malformed by Wireshark.
2. It has a length of 2 Bytes.
3. There is a TCP payload of 2 Bytes.
4. The payload is 4 extra zeroes (00 00).

Recommended Actions

The actions listed in this section have as a goal to further narrow down the issue.

Action 1. Take additional captures. Include captures at the endpoints and if possible, try to apply the divide and conquer method to isolate the source of the packet corruption, for example:

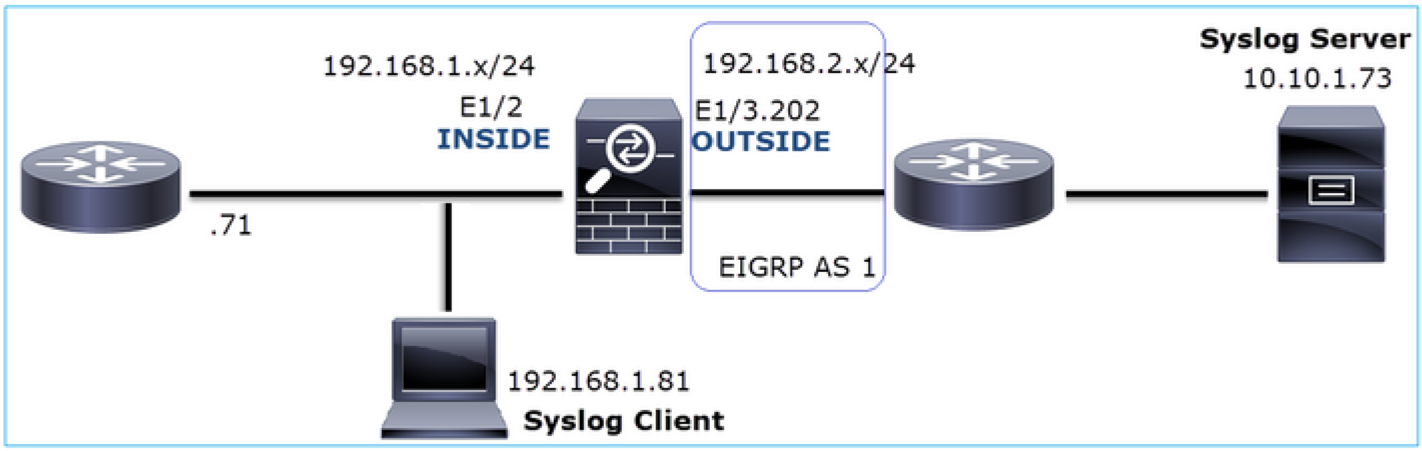


In this case, the 2 extra Bytes were added by the switch 'A' interface driver and the solution was to replace the switch that causes the corruption.

Case 8. UDP Connectivity Problem (Missing Packets)

Problem Description: Syslog (UDP 514) messages are not seen on the destination Syslog server.

This image shows the topology:



Affected Flow:

Src IP: 192.168.1.81

Dst IP: 10.10.1.73

Protocol: UDP 514

Capture Analysis

Enable captures on FTD LINA engine:

<#root>

```
firepower#
```

`capture CAPI int INSIDE trace match udp host 192.168.1.81 host 10.10.1.73 eq 514`

```
firepower#
```

`capture CAPO int OUTSIDE match udp host 192.168.1.81 host 10.10.1.73 eq 514`

FTD captures show no packets:

<#root>

```
firepower#
```

`show capture`

```
capture CAPI type raw-data trace interface INSIDE [Capturing - 0 bytes]
  match udp host 192.168.1.81 host 10.10.1.73 eq syslog
capture CAPO type raw-data interface OUTSIDE [Capturing - 0 bytes]
  match udp host 192.168.1.81 host 10.10.1.73 eq syslog
```

Recommended Actions

The actions listed in this section have as a goal to further narrow down the issue.

Action 1. Check the FTD connection table.

To check a specific connection you can use this syntax:

<#root>

```
firepower#
```

`show conn address 192.168.1.81 port 514`

```
10 in use, 3627189 most used
Inspect Snort:
        preserve-connection: 6 enabled, 0 in effect, 74 most enabled, 0 most in effect

UDP
```

INSIDE

```
 10.10.1.73:514
```

INSIDE

```
 192.168.1.81:514, idle 0:00:00, bytes
```

480379697

```
, flags -
```

o

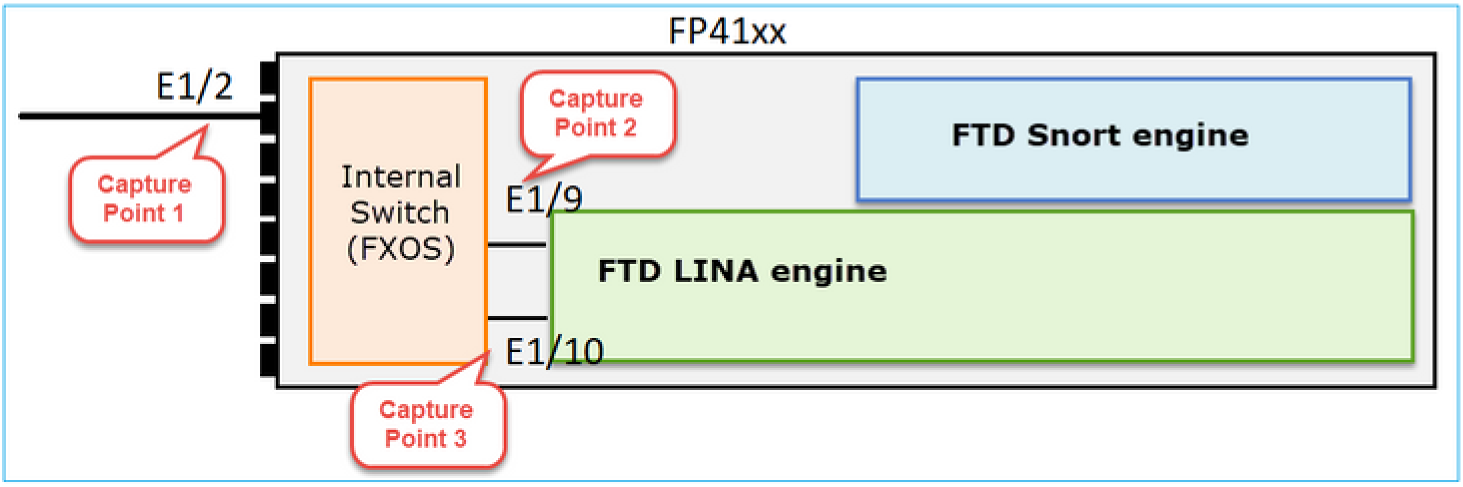
```
N1
```

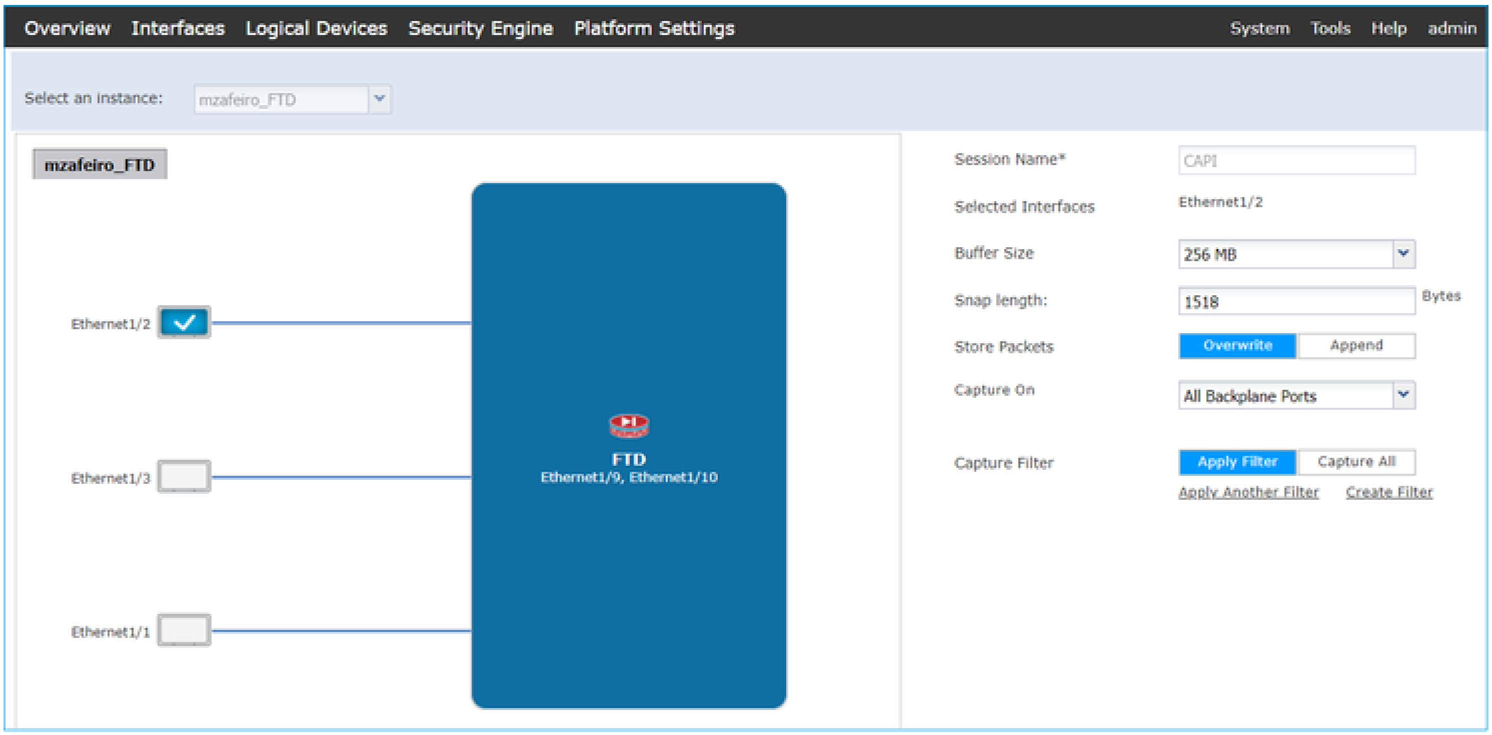
Key Points:

1. The ingress and egress interfaces are the same (U-turn).
2. The number of Bytes has a significantly large value (~5 GBytes).
3. The flag â€˜oâ€™ denotes flow offload (HW accelerated flow). This is the reason why the FTD captures do not show any packets. Flow offload is only supported on 41xx and 93xx platforms. In this case, the device is a 41xx.

Action 2. Take chassis-level captures.

Connect to the Firepower chassis manager and enable capture on the ingress interface (E1/2 in this case) and backplane interfaces (E1/9 and E1/10), as shown in the image:

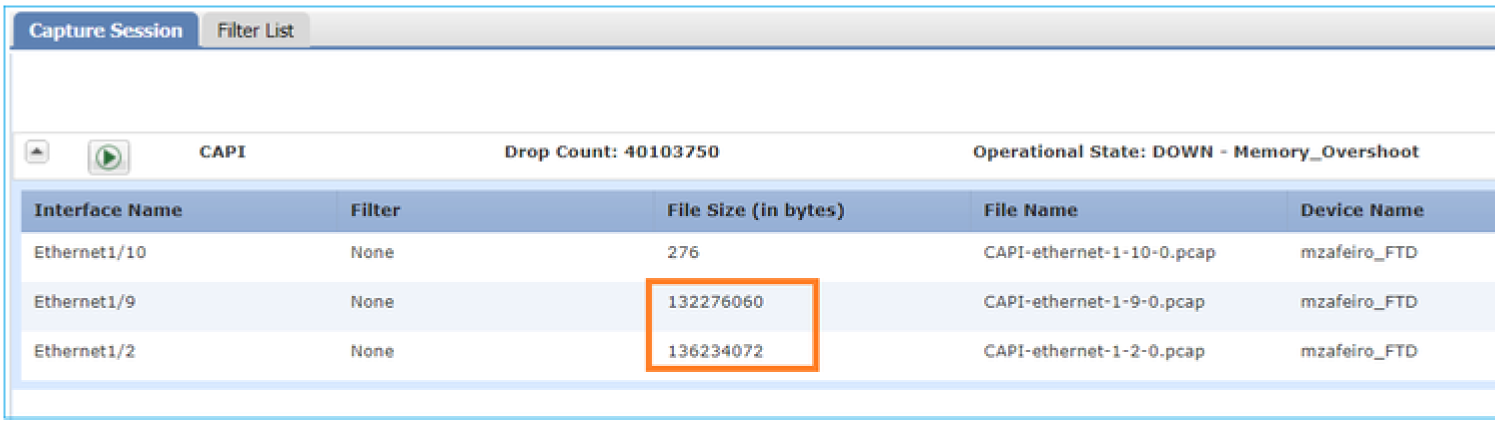




Select an instance: mzafeiro_FTD

Session Name*	CAPI
Selected Interfaces	Ethernet1/2
Buffer Size	256 MB
Snap length:	1518 Bytes
Store Packets	[✓ Overwrite] Append
Capture On	All Backplane Ports
Capture Filter	[✓ Apply Filter] Capture All

After a few seconds:

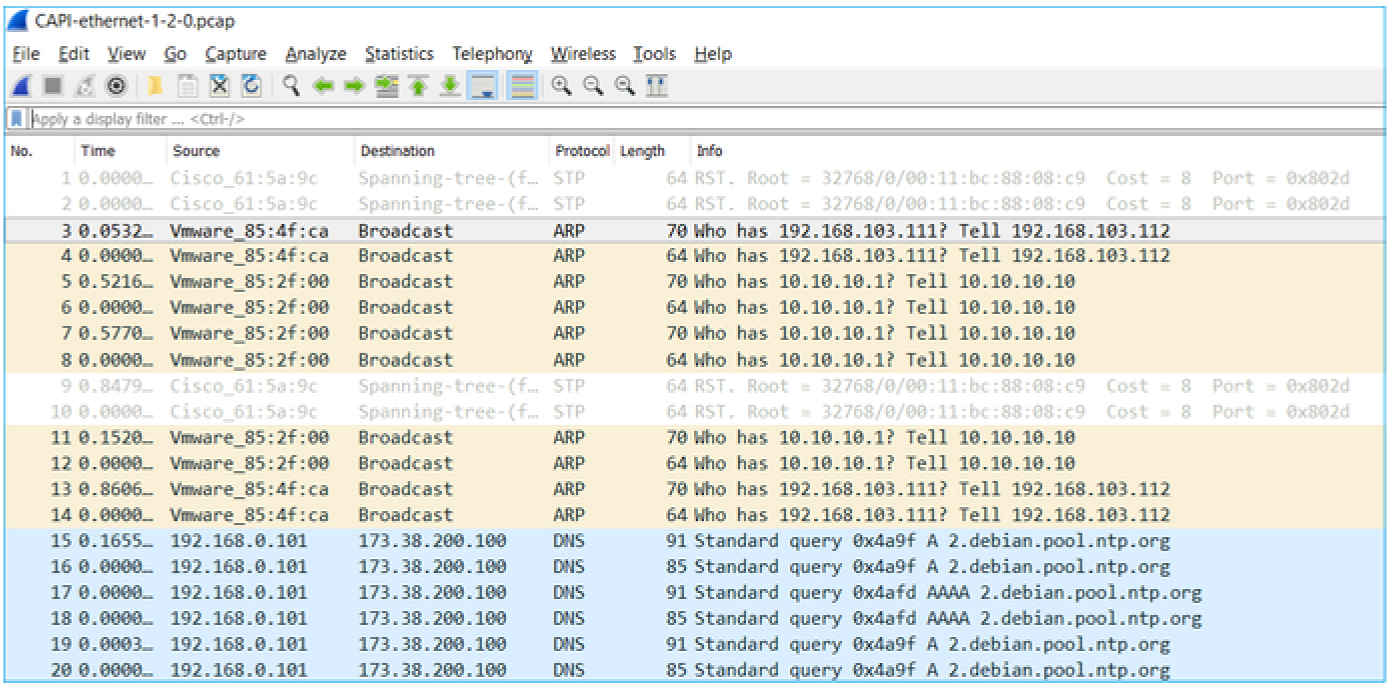


CAPI   Drop Count: 40103750   Operational State: DOWN - Memory_Overshoot

Interface Name	Filter	File Size (in bytes)	File Name	Device Name
Ethernet1/10	None	276	CAPI-ethernet-1-10-0.pcap	mzafeiro_FTD
Ethernet1/9	None	132276060	CAPI-ethernet-1-9-0.pcap	mzafeiro_FTD
Ethernet1/2	None	136234072	CAPI-ethernet-1-2-0.pcap	mzafeiro_FTD

Tip: In Wireshark exclude the VN-tagged packets to eliminate the packet duplication at the physical interface level

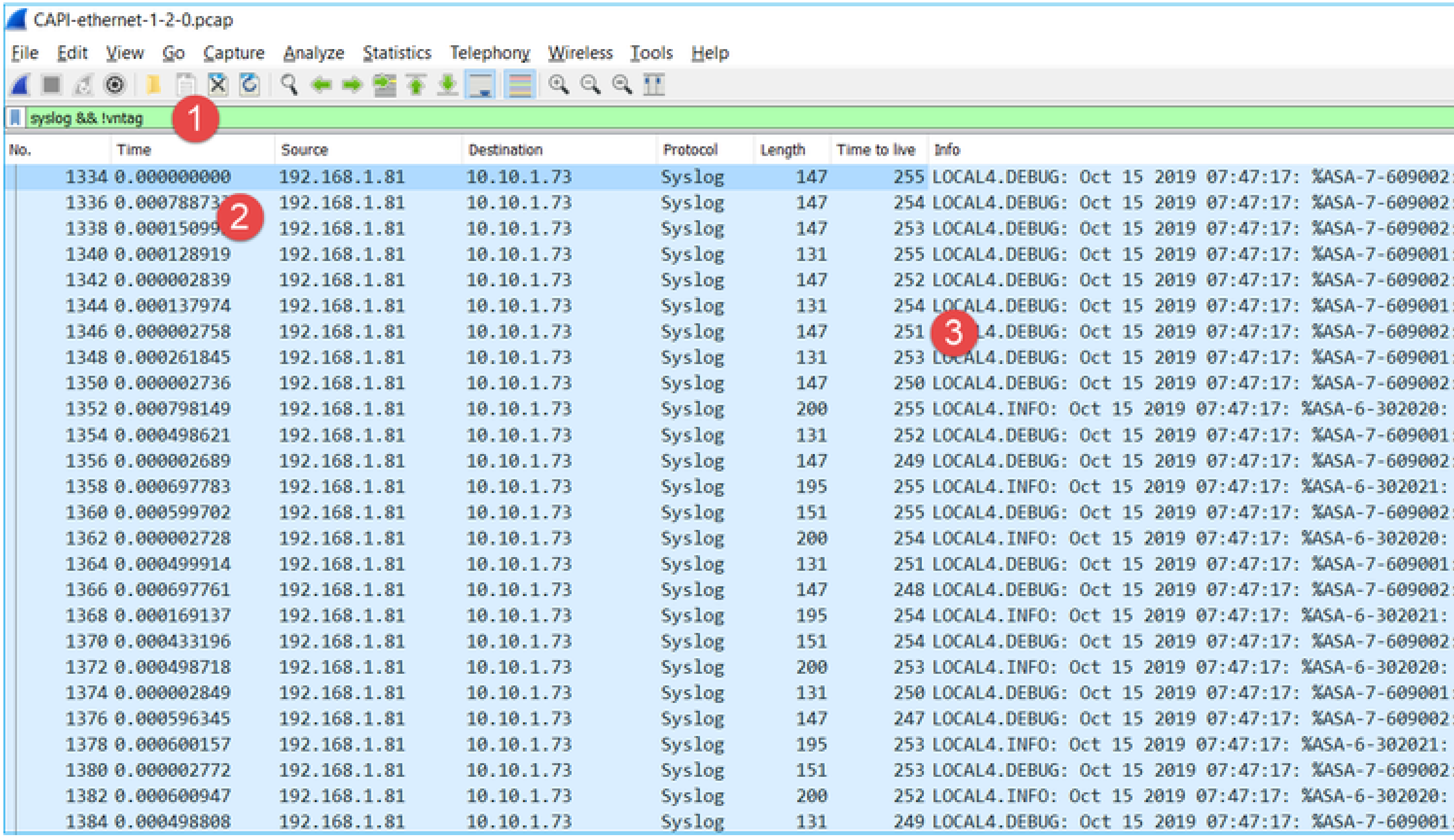
Before:



CAPI-ethernet-1-2-0.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
1	0.0000…	Cisco_61:5a:9c	Spanning-tree-(f…	STP	64	RST. Root = 32768/0/00:11:bc:88:08:c9 Cost = 8 Port = 0x802d
2	0.0000…	Cisco_61:5a:9c	Spanning-tree-(f…	STP	64	RST. Root = 32768/0/00:11:bc:88:08:c9 Cost = 8 Port = 0x802d
3	0.0532…	Vmware_85:4f:ca	Broadcast	ARP	70	Who has 192.168.103.111? Tell 192.168.103.112
4	0.0000…	Vmware_85:4f:ca	Broadcast	ARP	64	Who has 192.168.103.111? Tell 192.168.103.112
5	0.5216…	Vmware_85:2f:00	Broadcast	ARP	70	Who has 10.10.10.1? Tell 10.10.10.10
6	0.0000…	Vmware_85:2f:00	Broadcast	ARP	64	Who has 10.10.10.1? Tell 10.10.10.10
7	0.5770…	Vmware_85:2f:00	Broadcast	ARP	70	Who has 10.10.10.1? Tell 10.10.10.10
8	0.0000…	Vmware_85:2f:00	Broadcast	ARP	64	Who has 10.10.10.1? Tell 10.10.10.10
9	0.8479…	Cisco_61:5a:9c	Spanning-tree-(f…	STP	64	RST. Root = 32768/0/00:11:bc:88:08:c9 Cost = 8 Port = 0x802d
10	0.0000…	Cisco_61:5a:9c	Spanning-tree-(f…	STP	64	RST. Root = 32768/0/00:11:bc:88:08:c9 Cost = 8 Port = 0x802d
11	0.1520…	Vmware_85:2f:00	Broadcast	ARP	70	Who has 10.10.10.1? Tell 10.10.10.10
12	0.0000…	Vmware_85:2f:00	Broadcast	ARP	64	Who has 10.10.10.1? Tell 10.10.10.10
13	0.8606…	Vmware_85:4f:ca	Broadcast	ARP	70	Who has 192.168.103.111? Tell 192.168.103.112
14	0.0000…	Vmware_85:4f:ca	Broadcast	ARP	64	Who has 192.168.103.111? Tell 192.168.103.112
15	0.1655…	192.168.0.101	173.38.200.100	DNS	91	Standard query 0x4a9f A 2.debian.pool.ntp.org
16	0.0000…	192.168.0.101	173.38.200.100	DNS	85	Standard query 0x4a9f A 2.debian.pool.ntp.org
17	0.0000…	192.168.0.101	173.38.200.100	DNS	91	Standard query 0x4afd AAAA 2.debian.pool.ntp.org
18	0.0000…	192.168.0.101	173.38.200.100	DNS	85	Standard query 0x4afd AAAA 2.debian.pool.ntp.org
19	0.0003…	192.168.0.101	173.38.200.100	DNS	91	Standard query 0x4a9f A 2.debian.pool.ntp.org
20	0.0000…	192.168.0.101	173.38.200.100	DNS	85	Standard query 0x4a9f A 2.debian.pool.ntp.org

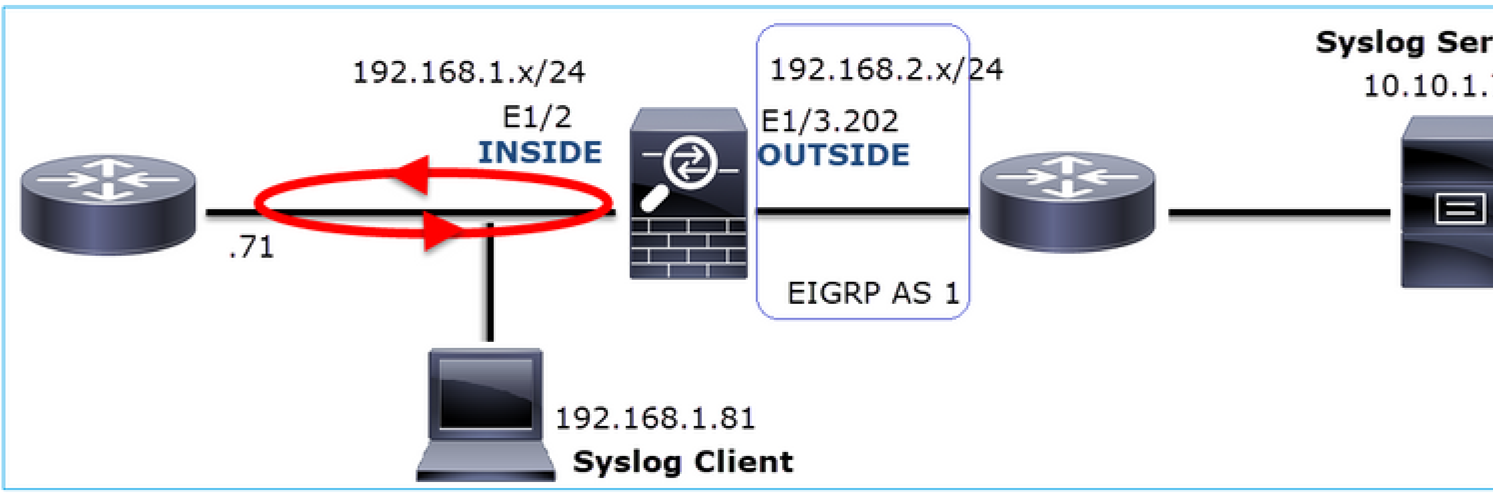
After:


CAPI-ethernet-1-2-0.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
syslog && !vntag
1
2
3

No.	Time	Source	Destination	Protocol	Length	Time to live	Info
1334	0.000000000	192.168.1.81	10.10.1.73	Syslog	147	255	LOCAL4.DEBUG: Oct 15 2019 07:47:17: %ASA-7-609002
1336	0.00078873	192.168.1.81	10.10.1.73	Syslog	147	254	LOCAL4.DEBUG: Oct 15 2019 07:47:17: %ASA-7-609002
1338	0.00015099	192.168.1.81	10.10.1.73	Syslog	147	253	LOCAL4.DEBUG: Oct 15 2019 07:47:17: %ASA-7-609002
1340	0.000128919	192.168.1.81	10.10.1.73	Syslog	131	255	LOCAL4.DEBUG: Oct 15 2019 07:47:17: %ASA-7-609001
1342	0.000002839	192.168.1.81	10.10.1.73	Syslog	147	252	LOCAL4.DEBUG: Oct 15 2019 07:47:17: %ASA-7-609002
1344	0.000137974	192.168.1.81	10.10.1.73	Syslog	131	254	LOCAL4.DEBUG: Oct 15 2019 07:47:17: %ASA-7-609001
1346	0.000002758	192.168.1.81	10.10.1.73	Syslog	147	251	4.DEBUG: Oct 15 2019 07:47:17: %ASA-7-609002
1348	0.000261845	192.168.1.81	10.10.1.73	Syslog	131	253	LOCAL4.DEBUG: Oct 15 2019 07:47:17: %ASA-7-609001
1350	0.000002736	192.168.1.81	10.10.1.73	Syslog	147	250	LOCAL4.DEBUG: Oct 15 2019 07:47:17: %ASA-7-609002
1352	0.000798149	192.168.1.81	10.10.1.73	Syslog	200	255	LOCAL4.INFO: Oct 15 2019 07:47:17: %ASA-6-302020:
1354	0.000498621	192.168.1.81	10.10.1.73	Syslog	131	252	LOCAL4.DEBUG: Oct 15 2019 07:47:17: %ASA-7-609001
1356	0.000002689	192.168.1.81	10.10.1.73	Syslog	147	249	LOCAL4.DEBUG: Oct 15 2019 07:47:17: %ASA-7-609002
1358	0.000697783	192.168.1.81	10.10.1.73	Syslog	195	255	LOCAL4.INFO: Oct 15 2019 07:47:17: %ASA-6-302021:
1360	0.000599702	192.168.1.81	10.10.1.73	Syslog	151	255	LOCAL4.DEBUG: Oct 15 2019 07:47:17: %ASA-7-609002
1362	0.000002728	192.168.1.81	10.10.1.73	Syslog	200	254	LOCAL4.INFO: Oct 15 2019 07:47:17: %ASA-6-302020:
1364	0.000499914	192.168.1.81	10.10.1.73	Syslog	131	251	LOCAL4.DEBUG: Oct 15 2019 07:47:17: %ASA-7-609001
1366	0.000697761	192.168.1.81	10.10.1.73	Syslog	147	248	LOCAL4.DEBUG: Oct 15 2019 07:47:17: %ASA-7-609002
1368	0.000169137	192.168.1.81	10.10.1.73	Syslog	195	254	LOCAL4.INFO: Oct 15 2019 07:47:17: %ASA-6-302021:
1370	0.000433196	192.168.1.81	10.10.1.73	Syslog	151	254	LOCAL4.DEBUG: Oct 15 2019 07:47:17: %ASA-7-609002
1372	0.000498718	192.168.1.81	10.10.1.73	Syslog	200	253	LOCAL4.INFO: Oct 15 2019 07:47:17: %ASA-6-302020:
1374	0.000002849	192.168.1.81	10.10.1.73	Syslog	131	250	LOCAL4.DEBUG: Oct 15 2019 07:47:17: %ASA-7-609001
1376	0.000596345	192.168.1.81	10.10.1.73	Syslog	147	247	LOCAL4.DEBUG: Oct 15 2019 07:47:17: %ASA-7-609002
1378	0.000600157	192.168.1.81	10.10.1.73	Syslog	195	253	LOCAL4.INFO: Oct 15 2019 07:47:17: %ASA-6-302021:
1380	0.000002772	192.168.1.81	10.10.1.73	Syslog	151	253	LOCAL4.DEBUG: Oct 15 2019 07:47:17: %ASA-7-609002
1382	0.000600947	192.168.1.81	10.10.1.73	Syslog	200	252	LOCAL4.INFO: Oct 15 2019 07:47:17: %ASA-6-302020:
1384	0.000498808	192.168.1.81	10.10.1.73	Syslog	131	249	LOCAL4.DEBUG: Oct 15 2019 07:47:17: %ASA-7-609001

Key Points:

1. A display filter is applied to remove packet duplicates and show only syslogs.
2. The diff between the packets is at the microsecond level. This indicates a very high packet rate.
3. The Time to Live (TTL) value decreases continuously. This indicates a packet loop.



Action 3. Use packet-tracer.

Since the packets do not traverse the firewall LINA engine you cannot do a live trace (capture w/trace), but you can trace an emulated packet with packet-tracer:

<#root>

```
firepower#

packet-tracer input INSIDE udp 10.10.1.73 514 192.168.1.81 514
```

```
Phase: 1
Type: CAPTURE
Subtype:
Result: ALLOW
Config:
Additional Information:
MAC Access list

Phase: 2
Type: ACCESS-LIST
Subtype:
Result: ALLOW
Config:
Implicit Rule
Additional Information:
MAC Access list

Phase: 3
Type: FLOW-LOOKUP
Subtype:
Result: ALLOW
Config:
Additional Information:
Found flow with id 25350892, using existing flow

Phase: 4
Type: SNORT
Subtype:
Result: ALLOW
Config:
Additional Information:
Snort Verdict: (fast-forward) fast forward this flow

Phase: 5
Type: ROUTE-LOOKUP
Subtype: Resolve Egress Interface
Result: ALLOW
Config:
Additional Information:
found next-hop 192.168.1.81 using egress ifc  INSIDE

Phase: 6
Type: ADJACENCY-LOOKUP
Subtype: next-hop and adjacency
Result: ALLOW
Config:
Additional Information:
adjacency Active
next-hop mac address a023.9f92.2a4d hits 1 reference 1

Phase: 7
Type: CAPTURE
Subtype:
Result: ALLOW
Config:
Additional Information:
MAC Access list

Result:
```

input-interface: INSIDE

```
input-status: up
```

```
input-line-status: up
```

```
output-interface: INSIDE
```

```
output-status: up
output-line-status: up
Action: allow
```

Action 4. Confirm the FTD routing.

Check the firewall routing table to see if there are any routing issues:

<#root>

```
firepower#
```

```
show route 10.10.1.73
```

```
Routing entry for 10.10.1.0 255.255.255.0
  Known via "eigrp 1", distance 90, metric 3072, type internal
  Redistributing via eigrp 1
  Last update from 192.168.2.72 on
```

```
OUTSIDE, 0:03:37 ago
```

```
  Routing Descriptor Blocks:
  * 192.168.2.72, from 192.168.2.72,
```

```
0:02:37 ago, via OUTSIDE
```

```
      Route metric is 3072, traffic share count is 1
      Total delay is 20 microseconds, minimum bandwidth is 1000000 Kbit
      Reliability 255/255, minimum MTU 1500 bytes
      Loading 29/255, Hops 1
```

Key Points:

1. The route points towards the correct egress interface.
2. The route was learned a few minutes ago (0:02:37).

Action 5. Confirm the connection uptime.

Check the connection uptime to see when this connection was established:

<#root>

```
firepower#
```

```
show conn address 192.168.1.81 port 514 detail
```

```
21 in use, 3627189 most used
Inspect Snort:
        preserve-connection: 19 enabled, 0 in effect, 74 most enabled, 0 most in effect
Flags: A - awaiting responder ACK to SYN, a - awaiting initiator ACK to SYN,
       b - TCP state-bypass or nailed,
       C - CTIQBE media, c - cluster centralized,
       D - DNS, d - dump, E - outside back connection, e - semi-distributed,
       F - initiator FIN, f - responder FIN,
       G - group, g - MGCP, H - H.323, h - H.225.0, I - initiator data,
       i - incomplete, J - GTP, j - GTP data, K - GTP t3-response
       k - Skinny media, L - decap tunnel, M - SMTP data, m - SIP media
       N - inspected by Snort (1 - preserve-connection enabled, 2 - preserve-connection in effect)
       n - GUP, O - responder data, o - offloaded,
       P - inside back connection, p - passenger flow
       q - SQL*Net data, R - initiator acknowledged FIN,
       R - UDP SUNRPC, r - responder acknowledged FIN,
       T - SIP, t - SIP transient, U - up,
       V - VPN orphan, v - M3UA W - WAAS,
       w - secondary domain backup,
       X - inspected by service module,
       x - per session, Y - director stub flow, y - backup stub flow,
       Z - Scansafe redirection, z - forwarding stub flow

UDP INSIDE: 10.10.1.73/514 INSIDE: 192.168.1.81/514,
    flags -oN1, idle 0s,
```

`uptime 3m49s`

```
, timeout 2m0s, bytes 4801148711
```

Key Point:

1. The connection was established ~4 minutes ago (this is before the EIGRP route installation in the routing table)

Action 6. Clear the established connection.

In this case, the packets match an established connection and are routed to a wrong egress interface; this causes a loop. This is because of the firewall order of operations:

1. Established connection lookup (this takes priority over the global routing table lookup).
2. Network Address Translation (NAT) lookup - UN-NAT (destination NAT) phase takes precedence over PBR and route lookup.
3. Policy-Based Routing (PBR)
4. Global routing table lookup

Since the connection never times out (the Syslog client continuously sends packets while the UDP conn idle timeout is 2 minutes) there is a need to manually clear the connection:

<#root>

```
firepower#
```

`clear conn address 10.10.1.73 address 192.168.1.81 protocol udp port 514`

```
1 connection(s) deleted.
```

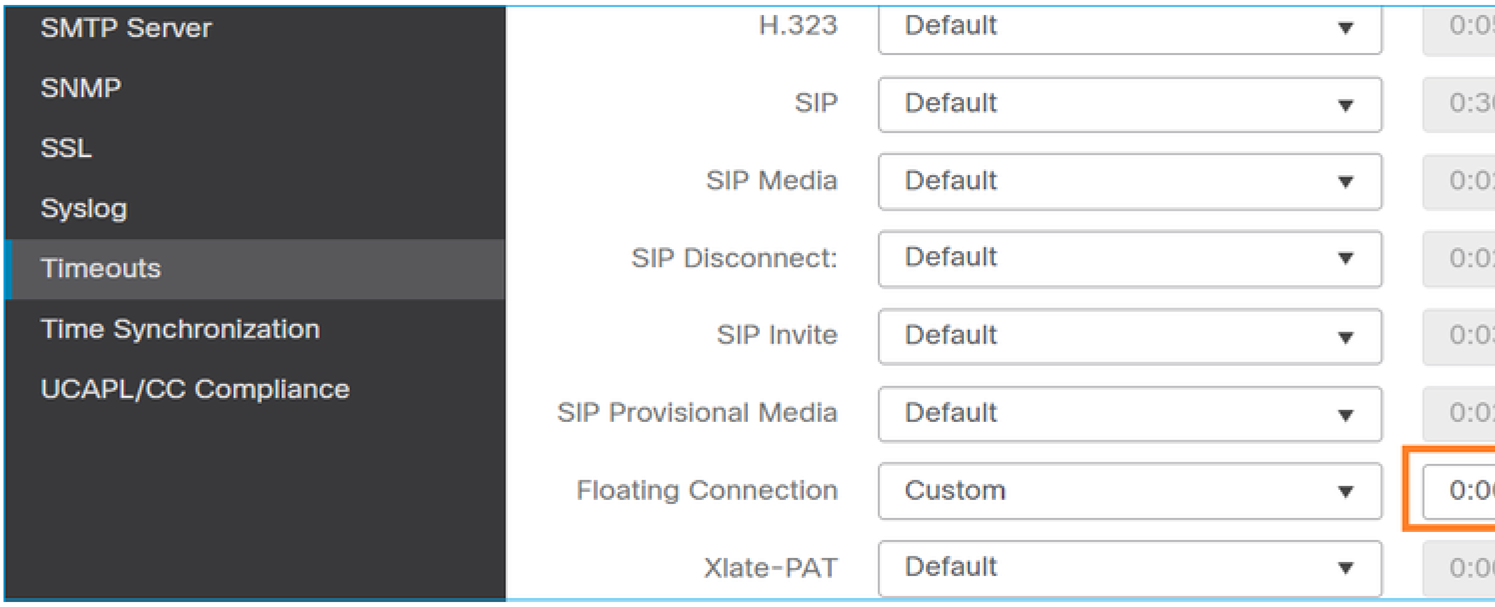
Verify that a new connection is established:

<#root>

```
firepower#

show conn address 192.168.1.81 port 514 detail | b 10.10.1.73.*192.168.1.81

UDP

OUTSIDE

: 10.10.1.73/514

INSIDE

: 192.168.1.81/514,
    flags -oN1, idle 1m15s, uptime 1m15s, timeout 2m0s, bytes 408
```

Action 7. Configure floating conn timeout.

This is the proper solution to address the issue and avoid suboptimal routing, especially for UDP flows. Navigate to **Devices** > **Platform Settings** > **Timeouts** and set the value:



H.323	Default	0:0
SIP	Default	0:3
SIP Media	Default	0:0
SIP Disconnect:	Default	0:0
SIP Invite	Default	0:0
SIP Provisional Media	Default	0:0
Floating Connection	Custom	0:0
Xlate-PAT	Default	0:0

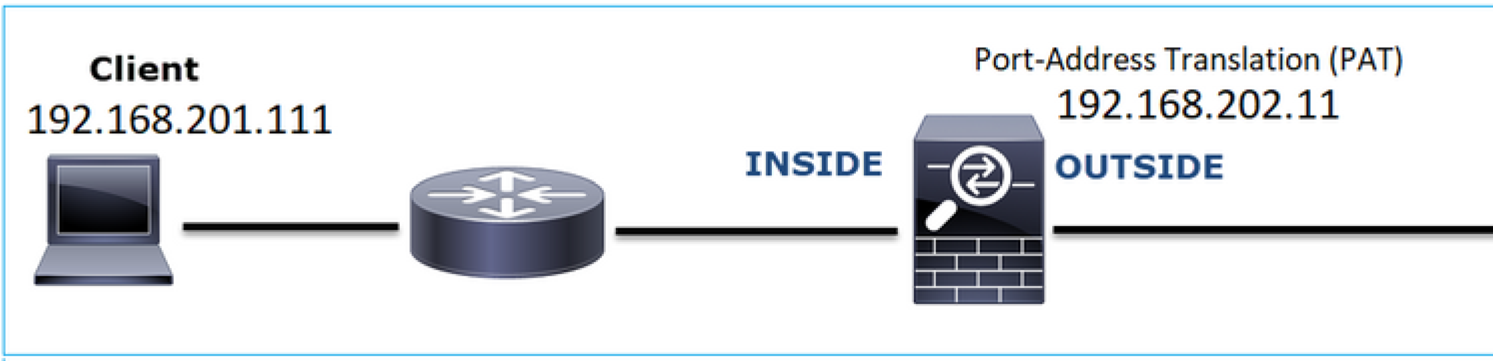
You can find more details about the floating conn timeout in the Command Reference:

https://www.cisco.com/c/en/us/td/docs/security/asa/asa-command-reference/T-Z/cmdref4/t1.html#pgfId-1649892

Case 9. HTTPS Connectivity Problem (Scenario 1)

Problem Description: HTTPS communication between the client 192.168.201.105 and server 192.168.202.101 cannot be established

This image shows the topology:



Affected Flow:

Src IP: 192.168.201.111

Dst IP: 192.168.202.111

Protocol: TCP 443 (HTTPS)

Capture Analysis

Enable captures on FTD LINA engine:

The IP used in the OUTSIDE capture is different due to the Port-Address Translation configuration.

<#root>

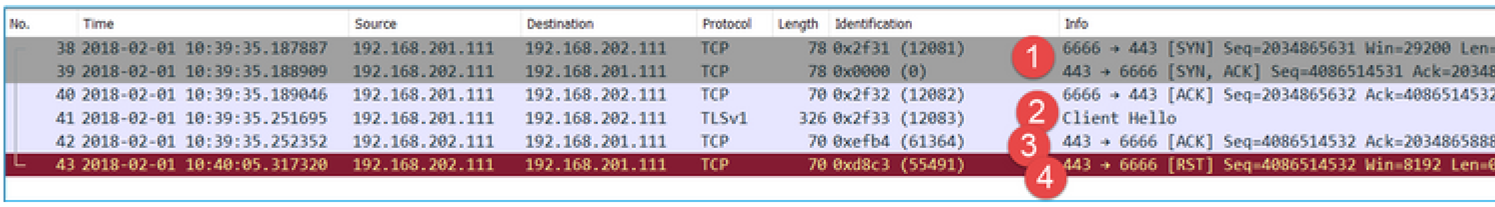
```
firepower#
```

`capture CAPI int INSIDE match ip host 192.168.201.111 host 192.168.202.111`

```
firepower#
```

`capture CAPO int OUTSIDE match ip host 192.168.202.11 host 192.168.202.111`

This image shows the capture taken on NGFW INSIDE interface:

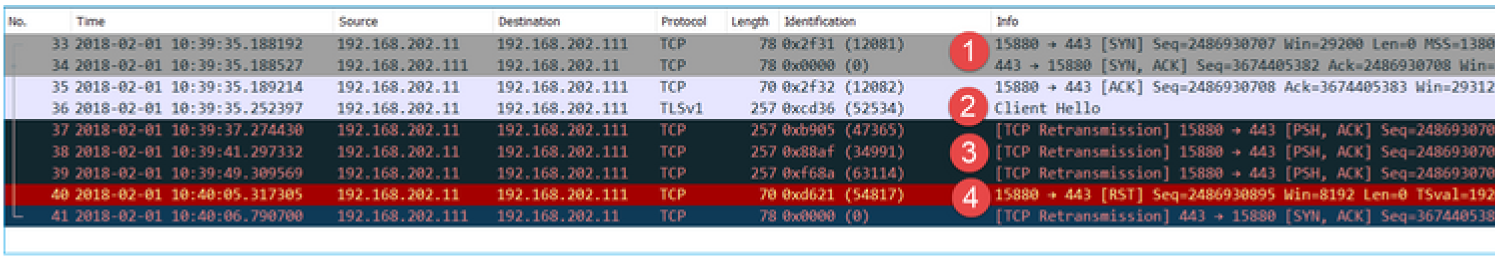


```
No.  Time                        Source           Destination      Protocol  Length  Identification     Info
  38 2018-02-01 10:39:35.187887  192.168.201.111  192.168.202.111  TCP           78  0x2f31 (12081)     6666 → 443 [SYN] Seq=2034865631 Win=29200 Len=
  39 2018-02-01 10:39:35.188909  192.168.202.111  192.168.201.111  TCP           78  0x0000 (0)         443 → 6666 [SYN, ACK] Seq=4086514531 Ack=20348
  40 2018-02-01 10:39:35.189046  192.168.201.111  192.168.202.111  TCP           70  0x2f32 (12082)     6666 → 443 [ACK] Seq=2034865632 Ack=408651453
  41 2018-02-01 10:39:35.251695  192.168.201.111  192.168.202.111  TLSv1        326  0x2f33 (12083)     Client Hello
  42 2018-02-01 10:39:35.252352  192.168.202.111  192.168.201.111  TCP           70  0xefb4 (61364)     443 → 6666 [ACK] Seq=4086514532 Ack=203486588
  43 2018-02-01 10:40:05.317320  192.168.202.111  192.168.201.111  TCP           70  0xd8c3 (55491)     443 → 6666 [RST] Seq=4086514532 Win=8192 Len=
```

Key Points:

1. There is a TCP 3-way handshake.
2. SSL Negotiation starts. The client sends a Client Hello message.
3. There is a TCP ACK sent to the client.
4. There is a TCP RST sent to the client.

This image shows the capture taken on NGFW OUTSIDE interface.



No.	Time	Source	Destination	Protocol	Length	Identification		Info
33	2018-02-01 10:39:35.188192	192.168.202.11	192.168.202.111	TCP	78	0x2f31 (12081)	1	15880 → 443 [SYN] Seq=2486930707 Win=29200 Len=0 MSS=1380
34	2018-02-01 10:39:35.188527	192.168.202.111	192.168.202.11	TCP	78	0x0000 (0)		443 → 15880 [SYN, ACK] Seq=3674405382 Ack=2486930708 Win=
35	2018-02-01 10:39:35.189214	192.168.202.11	192.168.202.111	TCP	70	0x2f32 (12082)		15880 → 443 [ACK] Seq=2486930708 Ack=3674405383 Win=29312
36	2018-02-01 10:39:35.252397	192.168.202.11	192.168.202.111	TLSv1	257	0xcd36 (52534)	2	Client Hello
37	2018-02-01 10:39:37.274430	192.168.202.11	192.168.202.111	TCP	257	0xb905 (47365)		[TCP Retransmission] 15880 → 443 [PSH, ACK] Seq=24869307
38	2018-02-01 10:39:41.297332	192.168.202.11	192.168.202.111	TCP	257	0x88af (34991)	3	[TCP Retransmission] 15880 → 443 [PSH, ACK] Seq=24869307
39	2018-02-01 10:39:49.309569	192.168.202.11	192.168.202.111	TCP	257	0xf68a (63114)		[TCP Retransmission] 15880 → 443 [PSH, ACK] Seq=24869307
40	2018-02-01 10:40:05.317305	192.168.202.11	192.168.202.111	TCP	70	0xd621 (54817)	4	15880 → 443 [RST] Seq=2486930895 Win=8192 Len=0 TSval=192
41	2018-02-01 10:40:06.790700	192.168.202.111	192.168.202.11	TCP	78	0x0000 (0)		[TCP Retransmission] 443 → 15880 [SYN, ACK] Seq=36744053

Key Points:

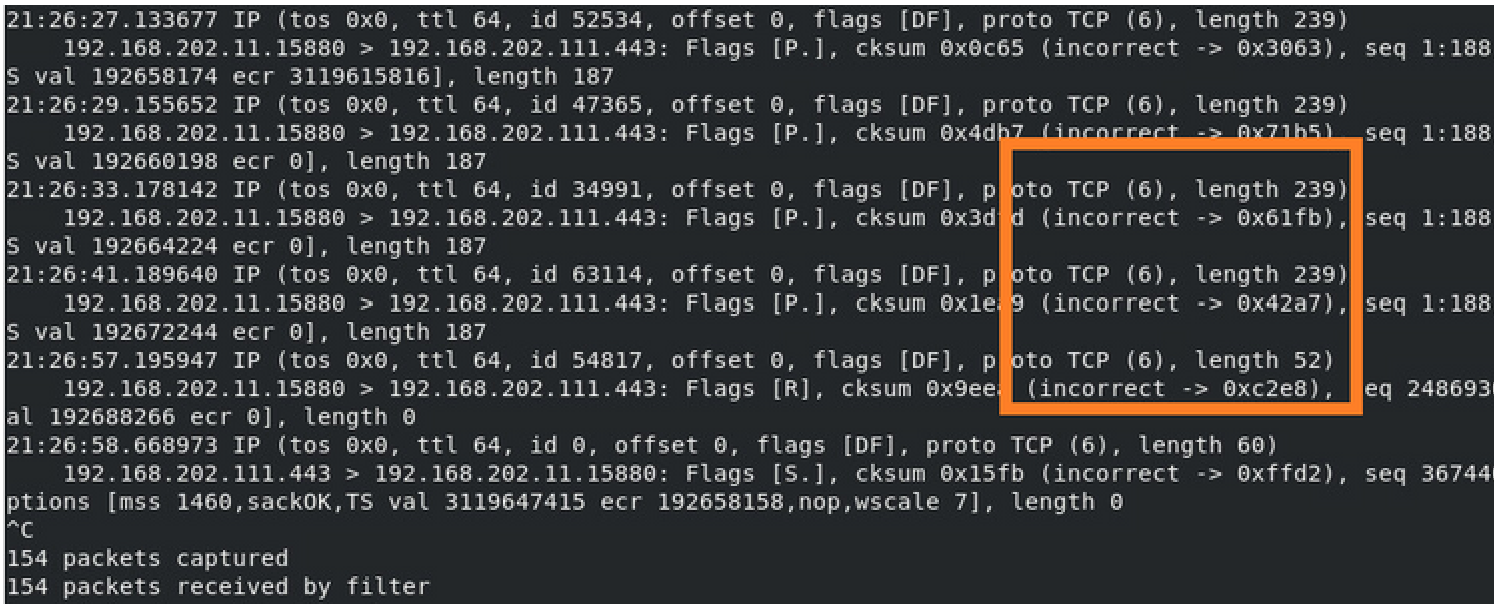
1. There is a TCP 3-way handshake.
2. SSL Negotiation starts. The client sends a Client Hello message.
3. There are TCP Retransmissions sent from the firewall towards the server.
4. There is a TCP RST sent to the server.

Recommended Actions

The actions listed in this section have as a goal to further narrow down the issue.

Action 1. Take additional captures.

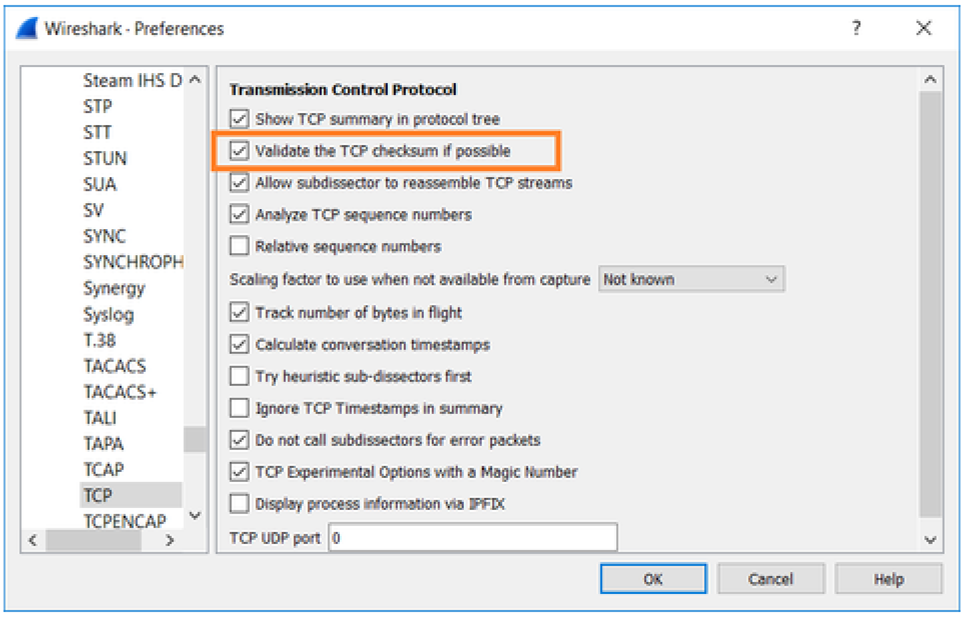
A capture taken on the server reveals that the server received the TLS Client Hellos with corrupted TCP checksum and silently drops them (there is no TCP RST or any other reply packet towards the client):



```
21:26:27.133677 IP (tos 0x0, ttl 64, id 52534, offset 0, flags [DF], proto TCP (6), length 239)
    192.168.202.11.15880 > 192.168.202.111.443: Flags [P.], cksum 0x0c65 (incorrect -> 0x3063), seq 1:188
S val 192658174 ecr 3119615816], length 187
21:26:29.155652 IP (tos 0x0, ttl 64, id 47365, offset 0, flags [DF], proto TCP (6), length 239)
    192.168.202.11.15880 > 192.168.202.111.443: Flags [P.], cksum 0x4db7 (incorrect -> 0x71b5), seq 1:188
S val 192660198 ecr 0], length 187
21:26:33.178142 IP (tos 0x0, ttl 64, id 34991, offset 0, flags [DF], p oto TCP (6), length 239)
    192.168.202.11.15880 > 192.168.202.111.443: Flags [P.], cksum 0x3d d (incorrect -> 0x61fb), seq 1:188
S val 192664224 ecr 0], length 187
21:26:41.189640 IP (tos 0x0, ttl 64, id 63114, offset 0, flags [DF], p oto TCP (6), length 239)
    192.168.202.11.15880 > 192.168.202.111.443: Flags [P.], cksum 0x1e 9 (incorrect -> 0x42a7), seq 1:188
S val 192672244 ecr 0], length 187
21:26:57.195947 IP (tos 0x0, ttl 64, id 54817, offset 0, flags [DF], p oto TCP (6), length 52)
    192.168.202.11.15880 > 192.168.202.111.443: Flags [R], cksum 0x9ee  (incorrect -> 0xc2e8),  eq 248693
al 192688266 ecr 0], length 0
21:26:58.668973 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 60)
    192.168.202.111.443 > 192.168.202.11.15880: Flags [S.], cksum 0x15fb (incorrect -> 0xffd2), seq 36744
ptions [mss 1460,sackOK,TS val 3119647415 ecr 192658158,nop,wscale 7], length 0
^C
154 packets captured
154 packets received by filter
```

When you put everything together:

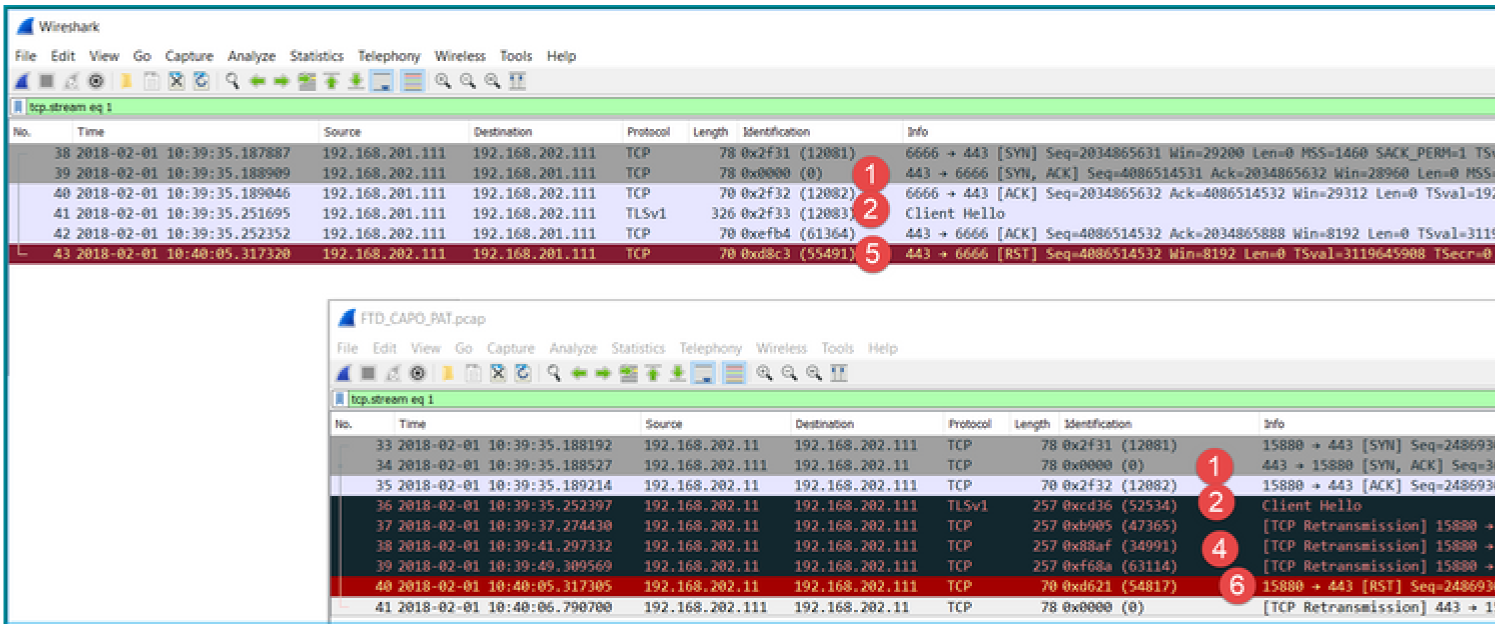
In this case, to understand, there is a need to enable on Wireshark the **Validate the TCP checksum if possible** option. Navigate to **Edit > Preferences > Protocols > TCP**, as shown in the image.



Transmission Control Protocol

- [x] Show TCP summary in protocol tree
- [x] Validate the TCP checksum if possible
- [x] Allow subdissector to reassemble TCP streams
- [x] Analyze TCP sequence numbers
- [ ] Relative sequence numbers
- Scaling factor to use when not available from capture: Not known
- [x] Track number of bytes in flight
- [x] Calculate conversation timestamps
- [ ] Try heuristic sub-dissectors first
- [ ] Ignore TCP Timestamps in summary
- [x] Do not call subdissectors for error packets
- [x] TCP Experimental Options with a Magic Number
- [ ] Display process information via IPFIX
- TCP UDP port: 0

In this case, it is helpful to put the captures side-by-side in order to get the full picture:

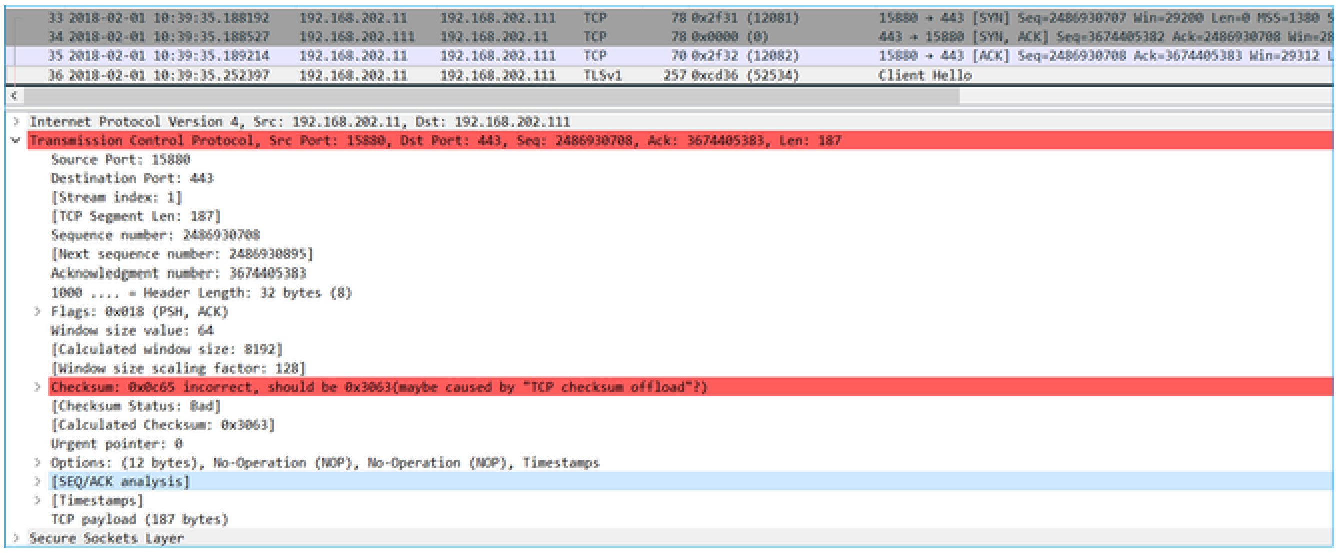

Wireshark
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
FTD_CAPO_PAT.pcap
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help
1
2
5
1
2
4
6

```
tcp.stream eq 1
No.  Time                        Source           Destination      Protocol  Length  Identification    Info
  38 2018-02-01 10:39:35.187887  192.168.201.111  192.168.202.111  TCP           78  0x2f31 (12081)    6666 → 443 [SYN] Seq=2034865631 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TS
  39 2018-02-01 10:39:35.188909  192.168.202.111  192.168.201.111  TCP           78  0x0000 (0)        443 → 6666 [SYN, ACK] Seq=4086514531 Ack=2034865632 Win=28960 Len=0 MSS
  40 2018-02-01 10:39:35.189046  192.168.201.111  192.168.202.111  TCP           70  0x2f32 (12082)    6666 → 443 [ACK] Seq=2034865632 Ack=4086514532 Win=29312 Len=0 TSval=19
  41 2018-02-01 10:39:35.251695  192.168.201.111  192.168.202.111  TLSv1        326  0x2f33 (12083)    Client Hello
  42 2018-02-01 10:39:35.252352  192.168.202.111  192.168.201.111  TCP           70  0xefb4 (61364)    443 → 6666 [ACK] Seq=4086514532 Ack=2034865888 Win=8192 Len=0 TSval=311
  43 2018-02-01 10:40:05.317320  192.168.202.111  192.168.201.111  TCP           70  0xd8c3 (55491)    443 → 6666 [RST] Seq=4086514532 Win=8192 Len=0 TSval=3119645908 TSecr=0
```

```
tcp.stream eq 1
No.  Time                        Source           Destination      Protocol  Length  Identification    Info
  33 2018-02-01 10:39:35.188192  192.168.202.11   192.168.202.111  TCP           78  0x2f31 (12081)    15880 → 443 [SYN] Seq=248693
  34 2018-02-01 10:39:35.188527  192.168.202.111  192.168.202.11   TCP           78  0x0000 (0)        443 → 15880 [SYN, ACK] Seq=3
  35 2018-02-01 10:39:35.189214  192.168.202.11   192.168.202.111  TCP           70  0x2f32 (12082)    15880 → 443 [ACK] Seq=248693
  36 2018-02-01 10:39:35.252397  192.168.202.11   192.168.202.111  TLSv1        257  0xcd36 (52534)    Client Hello
  37 2018-02-01 10:39:37.274430  192.168.202.11   192.168.202.111  TCP          257  0xb905 (47365)    [TCP Retransmission] 15880 →
  38 2018-02-01 10:39:41.297332  192.168.202.11   192.168.202.111  TCP          257  0x88af (34991)    [TCP Retransmission] 15880 →
  39 2018-02-01 10:39:49.309569  192.168.202.11   192.168.202.111  TCP          257  0xf68a (63114)    [TCP Retransmission] 15880 →
  40 2018-02-01 10:40:05.317305  192.168.202.11   192.168.202.111  TCP           70  0xd621 (54817)    15880 → 443 [RST] Seq=248693
  41 2018-02-01 10:40:06.790700  192.168.202.111  192.168.202.11   TCP           78  0x0000 (0)        [TCP Retransmission] 443 → 1
```

Key Points:

1. There is a TCP 3-way handshake. The IP IDs are the same. This means the flow was not proxied by the firewall.
2. A TLS Client Hello comes from the client with IP ID 12083. The packet is proxied by the firewall (the firewall, in this case, was configured with TLS Decryption Policy) and the IP ID is changed to 52534. Additionally, the packet TCP checksum gets corrupted (due to a software defect that later got fixed).
3. The firewall is in TCP Proxy mode and sends an ACK to the client (which spoofs the server).



```
33 2018-02-01 10:39:35.188192   192.168.202.11    192.168.202.111   TCP      78 0x2f31 (12081)    15880 → 443 [SYN] Seq=2486930707 Win=29200 Len=0 MSS=1380 S
34 2018-02-01 10:39:35.188527   192.168.202.111   192.168.202.11    TCP      78 0x0000 (0)        443 → 15880 [SYN, ACK] Seq=3674405382 Ack=2486930708 Win=28
35 2018-02-01 10:39:35.189214   192.168.202.11    192.168.202.111   TCP      70 0x2f32 (12082)    15880 → 443 [ACK] Seq=2486930708 Ack=3674405383 Win=29312 L
36 2018-02-01 10:39:35.252397   192.168.202.11    192.168.202.111   TLSv1   257 0xcd36 (52534)    Client Hello
```

```
> Internet Protocol Version 4, Src: 192.168.202.11, Dst: 192.168.202.111
v Transmission Control Protocol, Src Port: 15880, Dst Port: 443, Seq: 2486930708, Ack: 3674405383, Len: 187
    Source Port: 15880
    Destination Port: 443
    [Stream index: 1]
    [TCP Segment Len: 187]
    Sequence number: 2486930708
    [Next sequence number: 2486930895]
    Acknowledgment number: 3674405383
    1000 .... = Header Length: 32 bytes (8)
  > Flags: 0x018 (PSH, ACK)
    Window size value: 64
    [Calculated window size: 8192]
    [Window size scaling factor: 128]
  > Checksum: 0x0c65 incorrect, should be 0x3063(maybe caused by "TCP checksum offload"?)
    [Checksum Status: Bad]
    [Calculated Checksum: 0x3063]
    Urgent pointer: 0
  > Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  > [SEQ/ACK analysis]
  > [Timestamps]
    TCP payload (187 bytes)
> Secure Sockets Layer
```

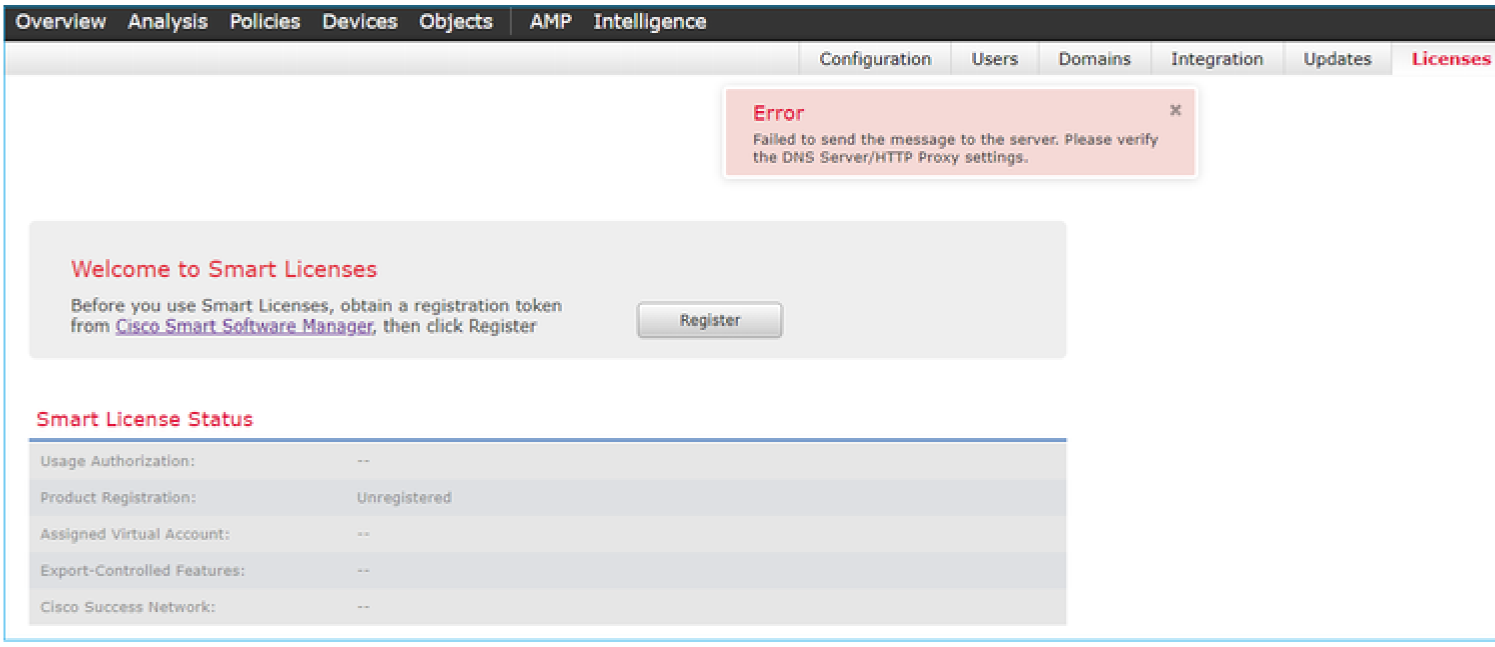
4. The firewall does not receive any TCP ACK packet from the server and retransmits the TLS Client Hello message. This is again due to TCP Proxy mode that the firewall activated.
5. After ~30 seconds the firewall gives up and sends a TCP RST towards the client.
6. The firewall sends a TCP RST towards the server.

For reference:

Firepower TLS/SSL Handshake Processing

Case 10. HTTPS Connectivity Problem (Scenario 2)

Problem Description: FMC Smart License registration fails.



Error

Failed to send the message to the server. Please verify the DNS Server/HTTP Proxy settings.

Welcome to Smart Licenses

Before you use Smart Licenses, obtain a registration token from Cisco Smart Software Manager, then click Register

Smart License Status

Usage Authorization:	--
Product Registration:	Unregistered
Assigned Virtual Account:	--
Export-Controlled Features:	--
Cisco Success Network:	--

This image shows the topology:

FMC
192.168.0.100
Cisco Licensing Portal

Affected Flow:

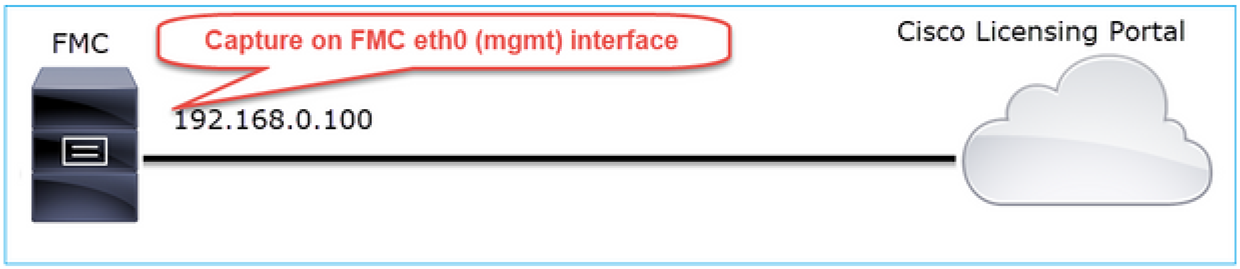
Src IP: 192.168.0.100

Dst: tools.cisco.com

Protocol: TCP 443 (HTTPS)

Capture Analysis

Enable capture on the FMC management interface:

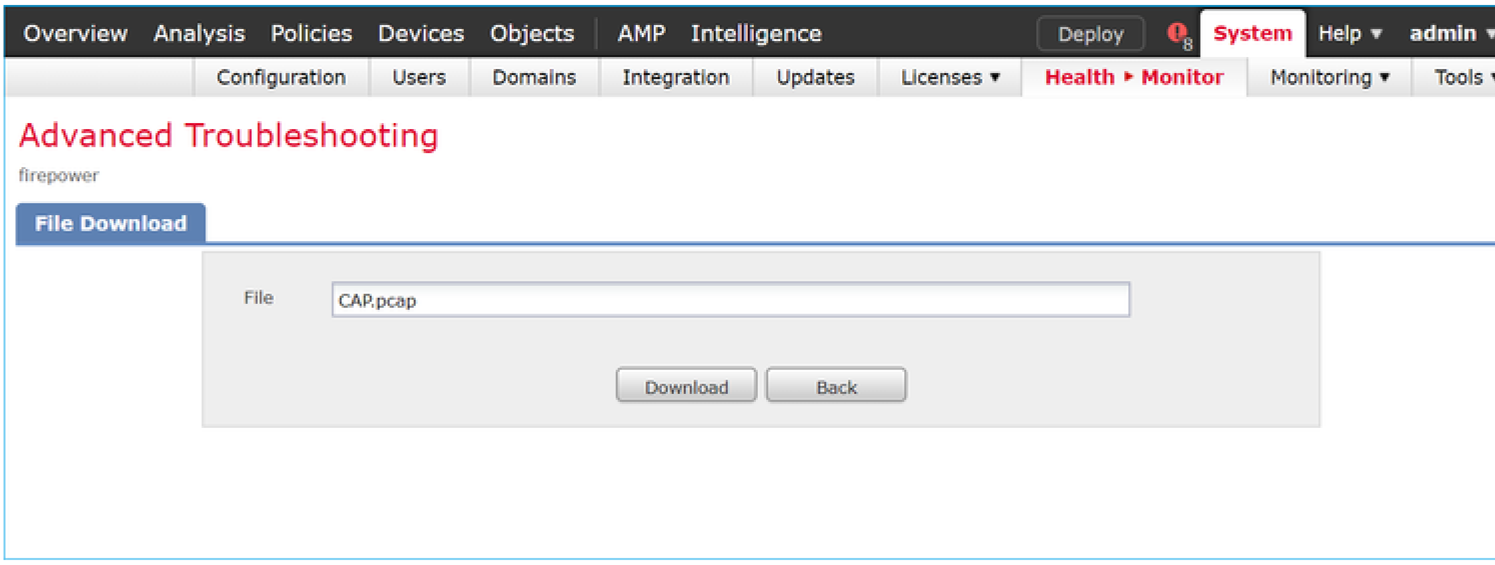


Try to register again. Once the Error message appears press CTRL-C to stop the capture:

<#root>

```
root@firepower:/Volume/home/admin#

tcpdump -i eth0 port 443 -s 0 -w CAP.pcap

HS_PACKET_BUFFER_SIZE is set to 4.
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes

^C

264 packets captured

<- CTRL-C

264 packets received by filter
0 packets dropped by kernel

root@firepower:/Volume/home/admin#
```

Collect the capture from the FMC (**System** > **Health** > **Monitor,** select the device and select **Advanced Troubleshooting**), as shown in the image:



Advanced Troubleshooting

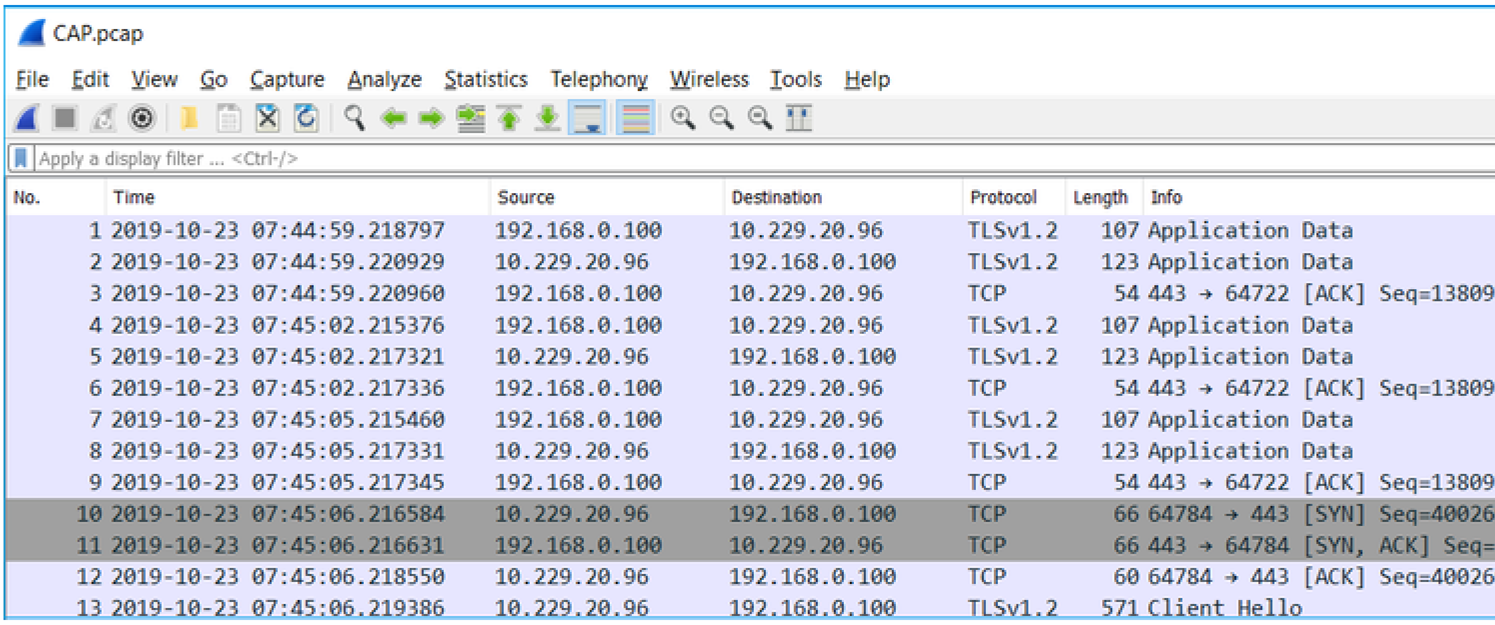
firepower

File Download

File: CAP.pcap

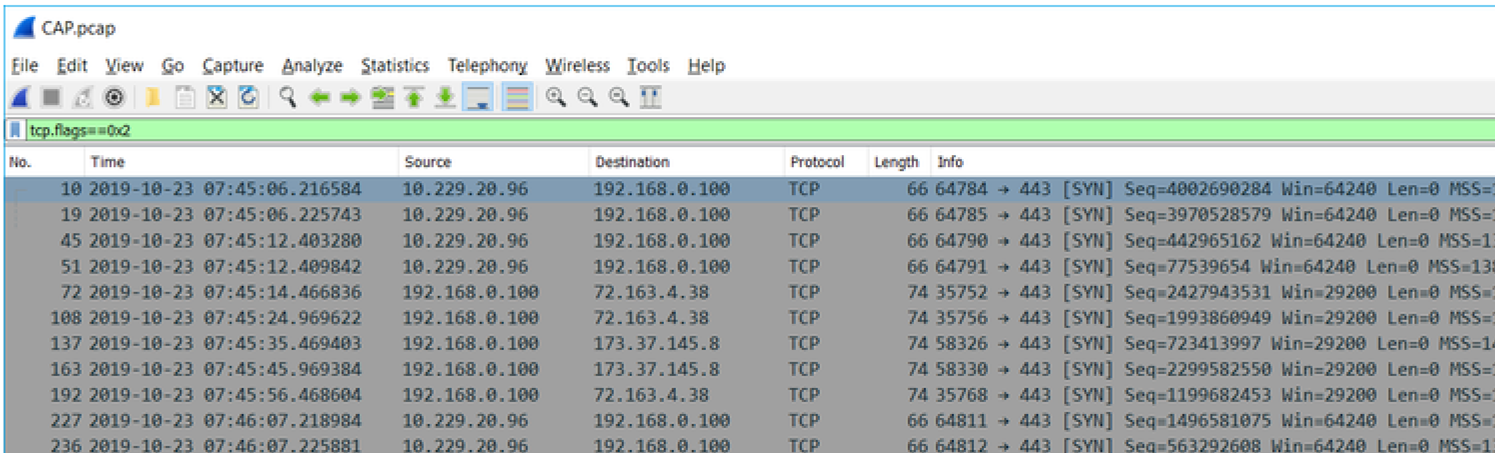
Download | Back

The image shows the FMC capture on Wireshark:



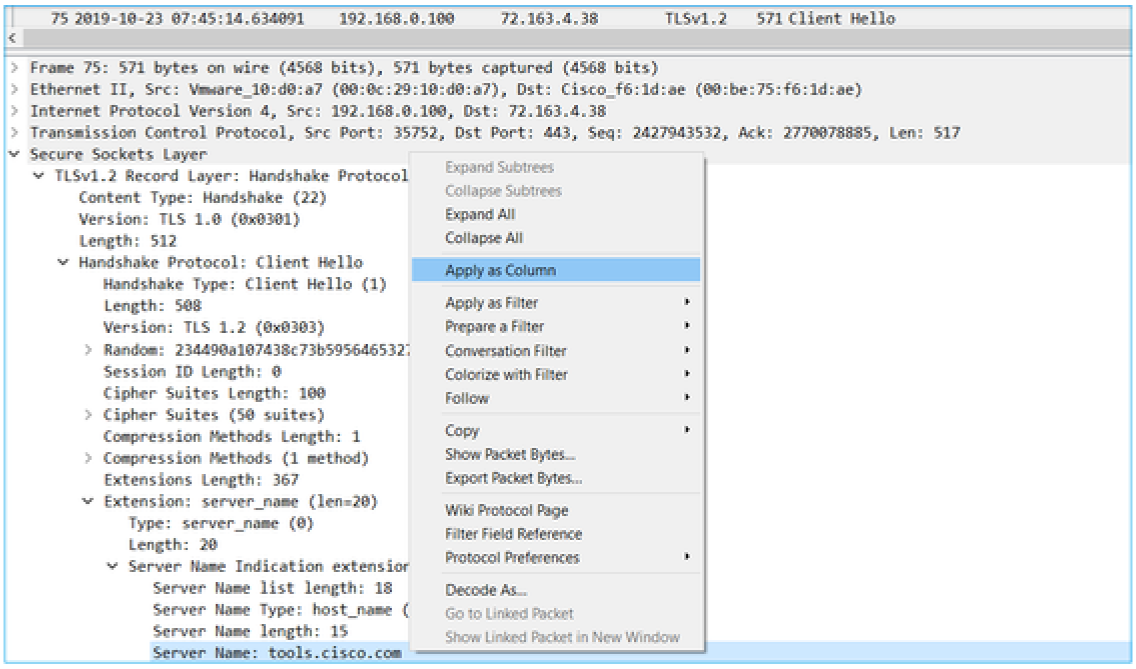
```
No.  Time                        Source          Destination     Protocol  Length Info
   1 2019-10-23 07:44:59.218797  192.168.0.100   10.229.20.96    TLSv1.2   107 Application Data
   2 2019-10-23 07:44:59.220929  10.229.20.96    192.168.0.100   TLSv1.2   123 Application Data
   3 2019-10-23 07:44:59.220960  192.168.0.100   10.229.20.96    TCP        54 443 → 64722 [ACK] Seq=13809
   4 2019-10-23 07:45:02.215376  192.168.0.100   10.229.20.96    TLSv1.2   107 Application Data
   5 2019-10-23 07:45:02.217321  10.229.20.96    192.168.0.100   TLSv1.2   123 Application Data
   6 2019-10-23 07:45:02.217336  192.168.0.100   10.229.20.96    TCP        54 443 → 64722 [ACK] Seq=13809
   7 2019-10-23 07:45:05.215460  192.168.0.100   10.229.20.96    TLSv1.2   107 Application Data
   8 2019-10-23 07:45:05.217331  10.229.20.96    192.168.0.100   TLSv1.2   123 Application Data
   9 2019-10-23 07:45:05.217345  192.168.0.100   10.229.20.96    TCP        54 443 → 64722 [ACK] Seq=13809
  10 2019-10-23 07:45:06.216584  10.229.20.96    192.168.0.100   TCP        66 64784 → 443 [SYN] Seq=40026
  11 2019-10-23 07:45:06.216631  192.168.0.100   10.229.20.96    TCP        66 443 → 64784 [SYN, ACK] Seq=
  12 2019-10-23 07:45:06.218550  10.229.20.96    192.168.0.100   TCP        60 64784 → 443 [ACK] Seq=40026
  13 2019-10-23 07:45:06.219386  10.229.20.96    192.168.0.100   TLSv1.2   571 Client Hello
```

Tip: In order to check for all new TCP sessions that were captured, use the **tcp.flags==0x2** display filter on Wireshark. This filters all the TCP SYN packets that were captured.



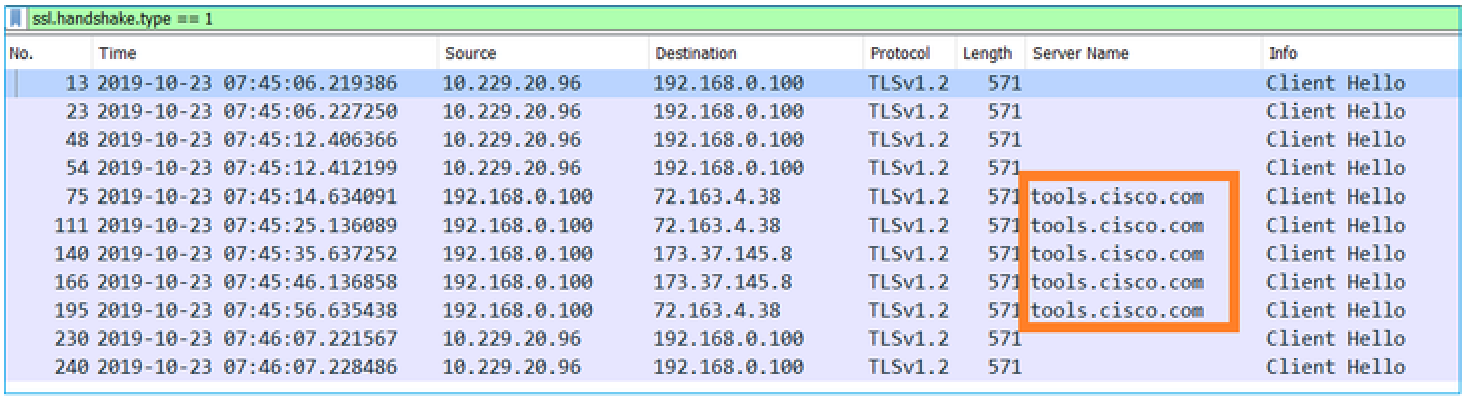
```
tcp.flags==0x2

No.  Time                        Source          Destination     Protocol  Length Info
  10 2019-10-23 07:45:06.216584  10.229.20.96    192.168.0.100   TCP        66 64784 → 443 [SYN] Seq=4002690284 Win=64240 Len=0 MSS=
  19 2019-10-23 07:45:06.225743  10.229.20.96    192.168.0.100   TCP        66 64785 → 443 [SYN] Seq=3970528579 Win=64240 Len=0 MSS=
  45 2019-10-23 07:45:12.403280  10.229.20.96    192.168.0.100   TCP        66 64790 → 443 [SYN] Seq=442965162 Win=64240 Len=0 MSS=1
  51 2019-10-23 07:45:12.409842  10.229.20.96    192.168.0.100   TCP        66 64791 → 443 [SYN] Seq=77539654 Win=64240 Len=0 MSS=13
  72 2019-10-23 07:45:14.466836  192.168.0.100   72.163.4.38     TCP        74 35752 → 443 [SYN] Seq=2427943531 Win=29200 Len=0 MSS=
 108 2019-10-23 07:45:24.969622  192.168.0.100   72.163.4.38     TCP        74 35756 → 443 [SYN] Seq=1993860949 Win=29200 Len=0 MSS=
 137 2019-10-23 07:45:35.469403  192.168.0.100   173.37.145.8    TCP        74 58326 → 443 [SYN] Seq=723413997 Win=29200 Len=0 MSS=1
 163 2019-10-23 07:45:45.969384  192.168.0.100   173.37.145.8    TCP        74 58330 → 443 [SYN] Seq=2299582550 Win=29200 Len=0 MSS=
 192 2019-10-23 07:45:56.468604  192.168.0.100   72.163.4.38     TCP        74 35768 → 443 [SYN] Seq=1199682453 Win=29200 Len=0 MSS=
 227 2019-10-23 07:46:07.218984  10.229.20.96    192.168.0.100   TCP        66 64811 → 443 [SYN] Seq=1496581075 Win=64240 Len=0 MSS=
 236 2019-10-23 07:46:07.225881  10.229.20.96    192.168.0.100   TCP        66 64812 → 443 [SYN] Seq=563292608 Win=64240 Len=0 MSS=1
```

Tip: Apply as Column the **Server Name** field from the SSL Client Hello.



```
   75 2019-10-23 07:45:14.634091    192.168.0.100      72.163.4.38        TLSv1.2    571 Client Hello

> Frame 75: 571 bytes on wire (4568 bits), 571 bytes captured (4568 bits)
> Ethernet II, Src: Vmware_10:d0:a7 (00:0c:29:10:d0:a7), Dst: Cisco_f6:1d:ae (00:be:75:f6:1d:ae)
> Internet Protocol Version 4, Src: 192.168.0.100, Dst: 72.163.4.38
> Transmission Control Protocol, Src Port: 35752, Dst Port: 443, Seq: 2427943532, Ack: 2770078885, Len: 517
v Secure Sockets Layer
  v TLSv1.2 Record Layer: Handshake Protocol
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 512
    v Handshake Protocol: Client Hello
        Handshake Type: Client Hello (1)
        Length: 508
        Version: TLS 1.2 (0x0303)
      > Random: 234490a107438c73b595646532
        Session ID Length: 0
        Cipher Suites Length: 100
      > Cipher Suites (50 suites)
        Compression Methods Length: 1
      > Compression Methods (1 method)
        Extensions Length: 367
      v Extension: server_name (len=20)
          Type: server_name (0)
          Length: 20
        v Server Name Indication extension
            Server Name list length: 18
            Server Name Type: host_name (
            Server Name length: 15
            Server Name: tools.cisco.com
```

Tip: Apply this display filter to see only the Client Hello messages **ssl.handshake.type == 1**

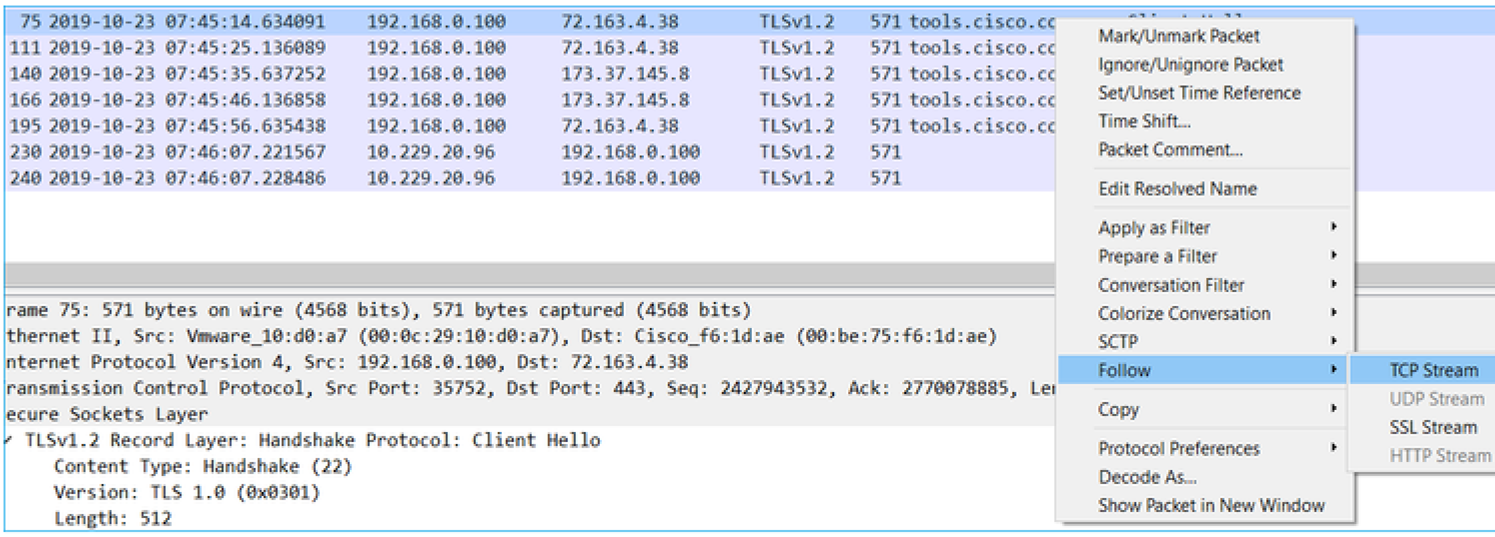


```
ssl.handshake.type == 1
```

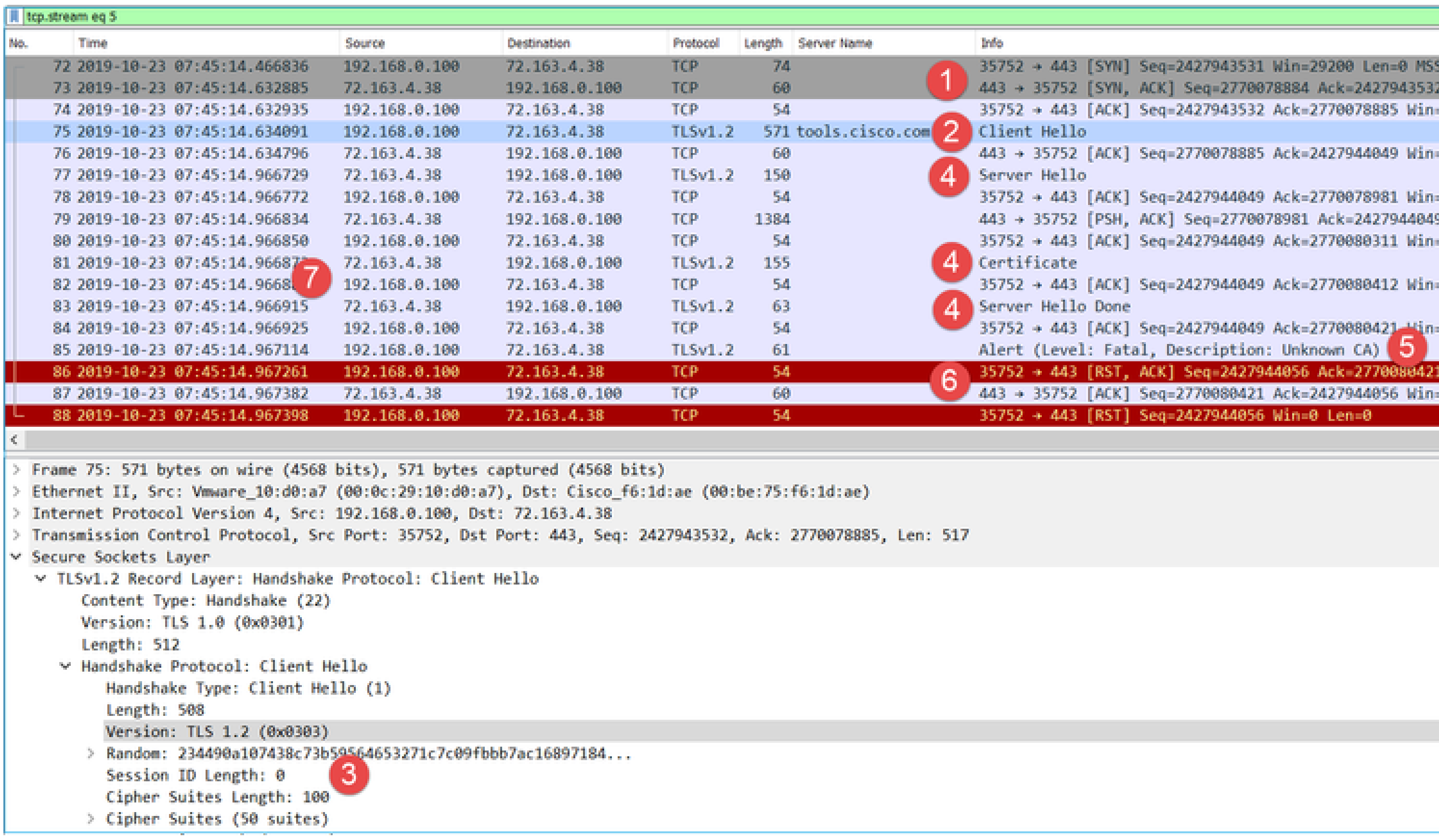
No.	Time	Source	Destination	Protocol	Length	Server Name	Info
13	2019-10-23 07:45:06.219386	10.229.20.96	192.168.0.100	TLSv1.2	571		Client Hello
23	2019-10-23 07:45:06.227250	10.229.20.96	192.168.0.100	TLSv1.2	571		Client Hello
48	2019-10-23 07:45:12.406366	10.229.20.96	192.168.0.100	TLSv1.2	571		Client Hello
54	2019-10-23 07:45:12.412199	10.229.20.96	192.168.0.100	TLSv1.2	571		Client Hello
75	2019-10-23 07:45:14.634091	192.168.0.100	72.163.4.38	TLSv1.2	571	tools.cisco.com	Client Hello
111	2019-10-23 07:45:25.136089	192.168.0.100	72.163.4.38	TLSv1.2	571	tools.cisco.com	Client Hello
140	2019-10-23 07:45:35.637252	192.168.0.100	173.37.145.8	TLSv1.2	571	tools.cisco.com	Client Hello
166	2019-10-23 07:45:46.136858	192.168.0.100	173.37.145.8	TLSv1.2	571	tools.cisco.com	Client Hello
195	2019-10-23 07:45:56.635438	192.168.0.100	72.163.4.38	TLSv1.2	571	tools.cisco.com	Client Hello
230	2019-10-23 07:46:07.221567	10.229.20.96	192.168.0.100	TLSv1.2	571		Client Hello
240	2019-10-23 07:46:07.228486	10.229.20.96	192.168.0.100	TLSv1.2	571		Client Hello

Note: At the time of this writing, the Smart Licensing portal (tools.cisco.com) uses these IPs: 72.163.4.38, 173.37.145.8

Follow one of the TCP flows (**Follow > TCP Stream)**, as shown in the image.



```
 75 2019-10-23 07:45:14.634091   192.168.0.100   72.163.4.38    TLSv1.2   571 tools.cisco.cc
111 2019-10-23 07:45:25.136089   192.168.0.100   72.163.4.38    TLSv1.2   571 tools.cisco.cc
140 2019-10-23 07:45:35.637252   192.168.0.100   173.37.145.8   TLSv1.2   571 tools.cisco.cc
166 2019-10-23 07:45:46.136858   192.168.0.100   173.37.145.8   TLSv1.2   571 tools.cisco.cc
195 2019-10-23 07:45:56.635438   192.168.0.100   72.163.4.38    TLSv1.2   571 tools.cisco.cc
230 2019-10-23 07:46:07.221567   10.229.20.96    192.168.0.100  TLSv1.2   571
240 2019-10-23 07:46:07.228486   10.229.20.96    192.168.0.100  TLSv1.2   571

rame 75: 571 bytes on wire (4568 bits), 571 bytes captured (4568 bits)
thernet II, Src: Vmware_10:d0:a7 (00:0c:29:10:d0:a7), Dst: Cisco_f6:1d:ae (00:be:75:f6:1d:ae)
nternet Protocol Version 4, Src: 192.168.0.100, Dst: 72.163.4.38
ransmission Control Protocol, Src Port: 35752, Dst Port: 443, Seq: 2427943532, Ack: 2770078885, Le
ecure Sockets Layer
 TLSv1.2 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 512
```

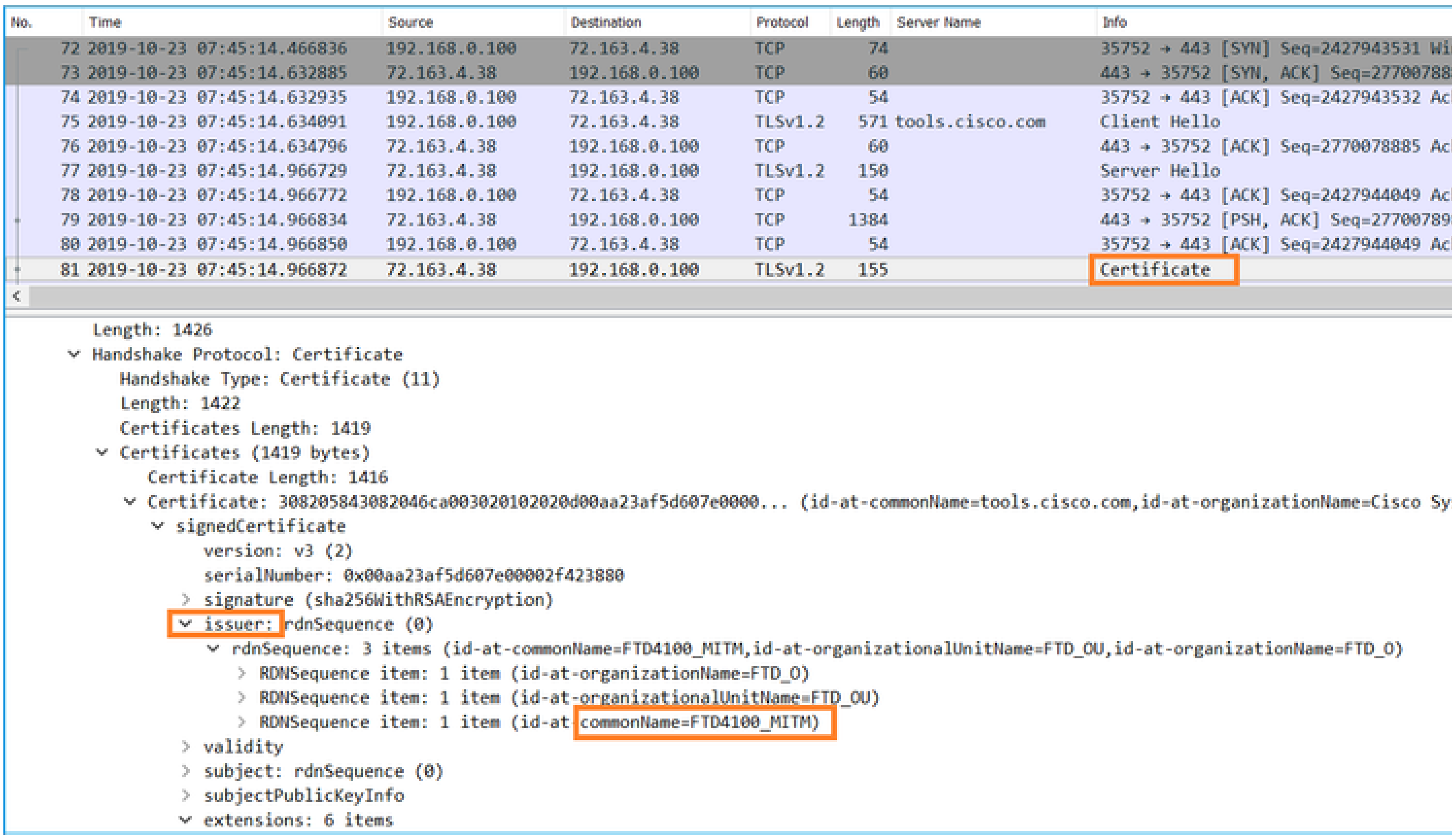


```
tcp.stream eq 5

No.  Time                        Source          Destination     Protocol  Length  Server Name       Info
 72 2019-10-23 07:45:14.466836   192.168.0.100   72.163.4.38     TCP       74                        35752 → 443 [SYN] Seq=2427943531 Win=29200 Len=0 MSS
 73 2019-10-23 07:45:14.632885   72.163.4.38     192.168.0.100   TCP       60                        443 → 35752 [SYN, ACK] Seq=2770078884 Ack=242794353
 74 2019-10-23 07:45:14.632935   192.168.0.100   72.163.4.38     TCP       54                        35752 → 443 [ACK] Seq=2427943532 Ack=2770078885 Win
 75 2019-10-23 07:45:14.634091   192.168.0.100   72.163.4.38     TLSv1.2   571     tools.cisco.com   Client Hello
 76 2019-10-23 07:45:14.634796   72.163.4.38     192.168.0.100   TCP       60                        443 → 35752 [ACK] Seq=2770078885 Ack=2427944049 Win
 77 2019-10-23 07:45:14.966729   72.163.4.38     192.168.0.100   TLSv1.2   150                       Server Hello
 78 2019-10-23 07:45:14.966772   192.168.0.100   72.163.4.38     TCP       54                        35752 → 443 [ACK] Seq=2427944049 Ack=2770078981 Win
 79 2019-10-23 07:45:14.966834   72.163.4.38     192.168.0.100   TCP       1384                      443 → 35752 [PSH, ACK] Seq=2770078981 Ack=242794404
 80 2019-10-23 07:45:14.966850   192.168.0.100   72.163.4.38     TCP       54                        35752 → 443 [ACK] Seq=2427944049 Ack=2770080311 Win
 81 2019-10-23 07:45:14.9668     72.163.4.38     192.168.0.100   TLSv1.2   155                       Certificate
 82 2019-10-23 07:45:14.9668     192.168.0.100   72.163.4.38     TCP       54                        35752 → 443 [ACK] Seq=2427944049 Ack=2770080412 Win
 83 2019-10-23 07:45:14.966915   72.163.4.38     192.168.0.100   TLSv1.2   63                        Server Hello Done
 84 2019-10-23 07:45:14.966925   192.168.0.100   72.163.4.38     TCP       54                        35752 → 443 [ACK] Seq=2427944049 Ack=2770080421 Win
 85 2019-10-23 07:45:14.967114   192.168.0.100   72.163.4.38     TLSv1.2   61                        Alert (Level: Fatal, Description: Unknown CA)
 86 2019-10-23 07:45:14.967261   192.168.0.100   72.163.4.38     TCP       54                        35752 → 443 [RST, ACK] Seq=2427944056 Ack=277008042
 87 2019-10-23 07:45:14.967382   72.163.4.38     192.168.0.100   TCP       60                        443 → 35752 [ACK] Seq=2770080421 Ack=2427944056 Win
 88 2019-10-23 07:45:14.967398   192.168.0.100   72.163.4.38     TCP       54                        35752 → 443 [RST] Seq=2427944056 Win=0 Len=0

> Frame 75: 571 bytes on wire (4568 bits), 571 bytes captured (4568 bits)
> Ethernet II, Src: Vmware_10:d0:a7 (00:0c:29:10:d0:a7), Dst: Cisco_f6:1d:ae (00:be:75:f6:1d:ae)
> Internet Protocol Version 4, Src: 192.168.0.100, Dst: 72.163.4.38
> Transmission Control Protocol, Src Port: 35752, Dst Port: 443, Seq: 2427943532, Ack: 2770078885, Len: 517
v Secure Sockets Layer
  v TLSv1.2 Record Layer: Handshake Protocol: Client Hello
      Content Type: Handshake (22)
      Version: TLS 1.0 (0x0301)
      Length: 512
    v Handshake Protocol: Client Hello
        Handshake Type: Client Hello (1)
        Length: 508
        Version: TLS 1.2 (0x0303)
      > Random: 234490a107438c73b59564653271c7c09fbbb7ac16897184...
        Session ID Length: 0
        Cipher Suites Length: 100
      > Cipher Suites (50 suites)
```

Key Points:

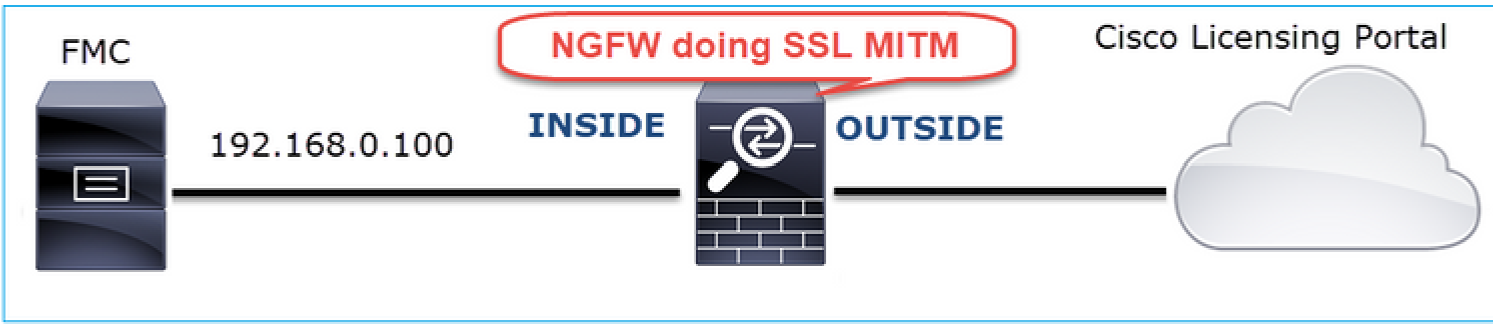
1. There is a TCP 3-way handshake.
2. The client (FMC) sends an SSL Client Hello message towards the Smart Licensing portal.
3. The SSL Session ID is 0. This means that it is not a resumed session.
4. The destination server replies with Server Hello, Certificate and Server Hello Done message.
5. The client sends an SSL Fatal Alert which regards an â€˜Unknown CAâ€™.
6. The client sends a TCP RST to close the session.
7. The whole TCP session duration (from establishment to closure) was ~0.5 sec.

Select the **Server Certificate** and expand the **issuer** field to see the commonName. In this case the Common Name reveals a device that does Man-in-the-middle (MITM).



```
No.  Time                        Source          Destination     Protocol  Length  Server Name       Info
  72 2019-10-23 07:45:14.466836  192.168.0.100   72.163.4.38     TCP           74                    35752 → 443 [SYN] Seq=2427943531 Wi
  73 2019-10-23 07:45:14.632885  72.163.4.38     192.168.0.100   TCP           60                    443 → 35752 [SYN, ACK] Seq=27700788
  74 2019-10-23 07:45:14.632935  192.168.0.100   72.163.4.38     TCP           54                    35752 → 443 [ACK] Seq=2427943532 Ac
  75 2019-10-23 07:45:14.634091  192.168.0.100   72.163.4.38     TLSv1.2      571 tools.cisco.com   Client Hello
  76 2019-10-23 07:45:14.634796  72.163.4.38     192.168.0.100   TCP           60                    443 → 35752 [ACK] Seq=2770078885 Ac
  77 2019-10-23 07:45:14.966729  72.163.4.38     192.168.0.100   TLSv1.2      150                    Server Hello
  78 2019-10-23 07:45:14.966772  192.168.0.100   72.163.4.38     TCP           54                    35752 → 443 [ACK] Seq=2427944049 Ac
  79 2019-10-23 07:45:14.966834  72.163.4.38     192.168.0.100   TCP         1384                    443 → 35752 [PSH, ACK] Seq=27700789
  80 2019-10-23 07:45:14.966850  192.168.0.100   72.163.4.38     TCP           54                    35752 → 443 [ACK] Seq=2427944049 Ac
  81 2019-10-23 07:45:14.966872  72.163.4.38     192.168.0.100   TLSv1.2      155                    Certificate

      Length: 1426
    ⌄ Handshake Protocol: Certificate
        Handshake Type: Certificate (11)
        Length: 1422
        Certificates Length: 1419
      ⌄ Certificates (1419 bytes)
          Certificate Length: 1416
        ⌄ Certificate: 308205843082046ca003020102020d00aa23af5d607e0000... (id-at-commonName=tools.cisco.com,id-at-organizationName=Cisco Sy
          ⌄ signedCertificate
              version: v3 (2)
              serialNumber: 0x00aa23af5d607e00002f423880
            › signature (sha256WithRSAEncryption)
            ⌄ issuer: rdnSequence (0)
              ⌄ rdnSequence: 3 items (id-at-commonName=FTD4100_MITM,id-at-organizationalUnitName=FTD_OU,id-at-organizationName=FTD_O)
                › RDNSequence item: 1 item (id-at-organizationName=FTD_O)
                › RDNSequence item: 1 item (id-at-organizationalUnitName=FTD_OU)
                › RDNSequence item: 1 item (id-at-commonName=FTD4100_MITM)
            › validity
            › subject: rdnSequence (0)
            › subjectPublicKeyInfo
            ⌄ extensions: 6 items
```

This is shown in this image:

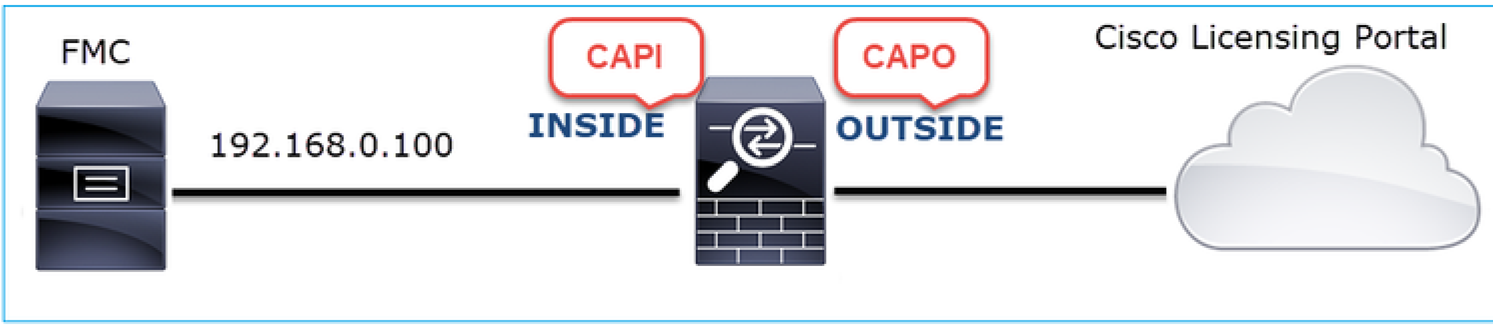


Recommended Actions

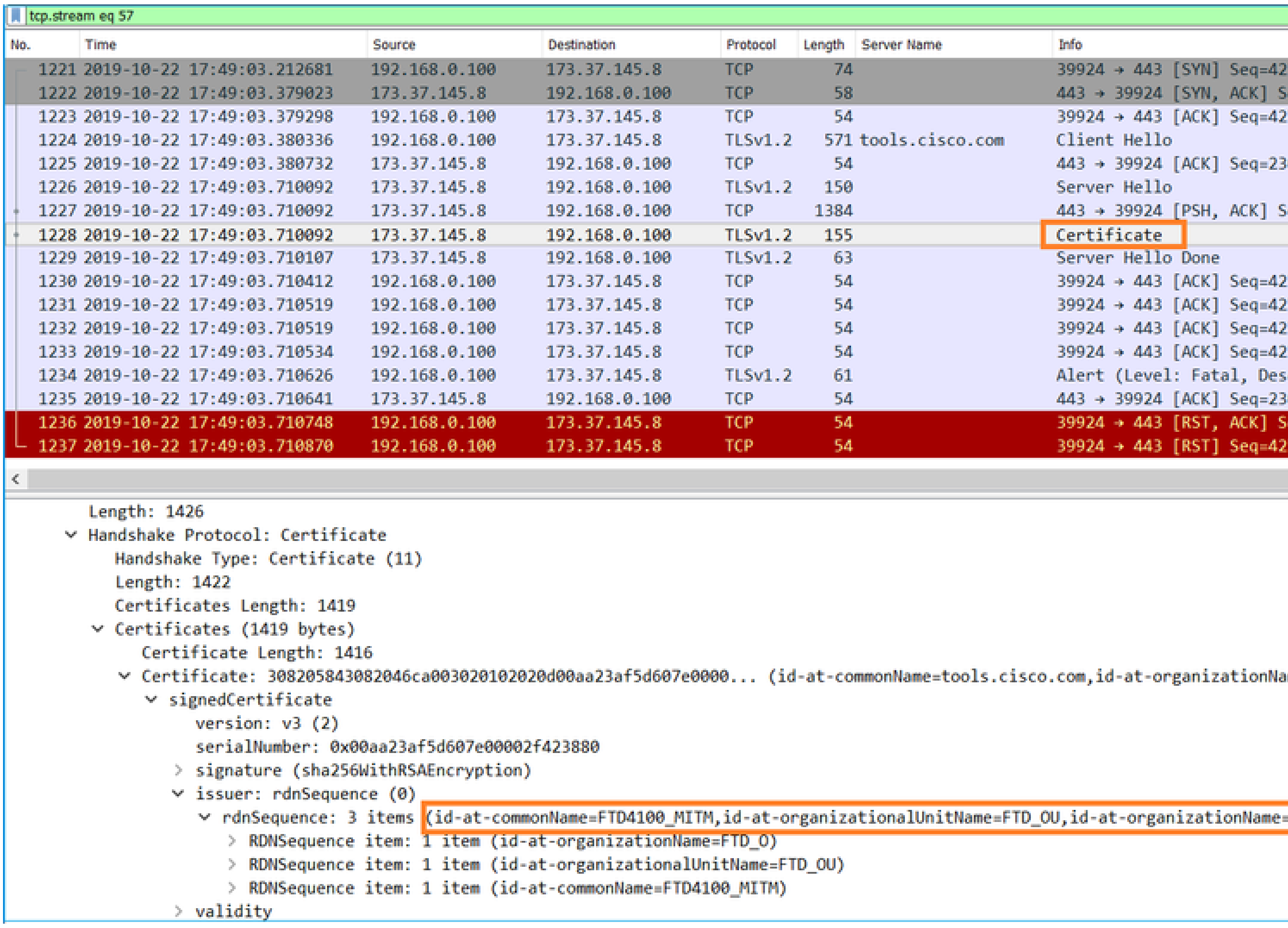
The actions listed in this section have as a goal to further narrow down the issue.

Action 1. Take additional captures.

Take captures on the transit firewall device:

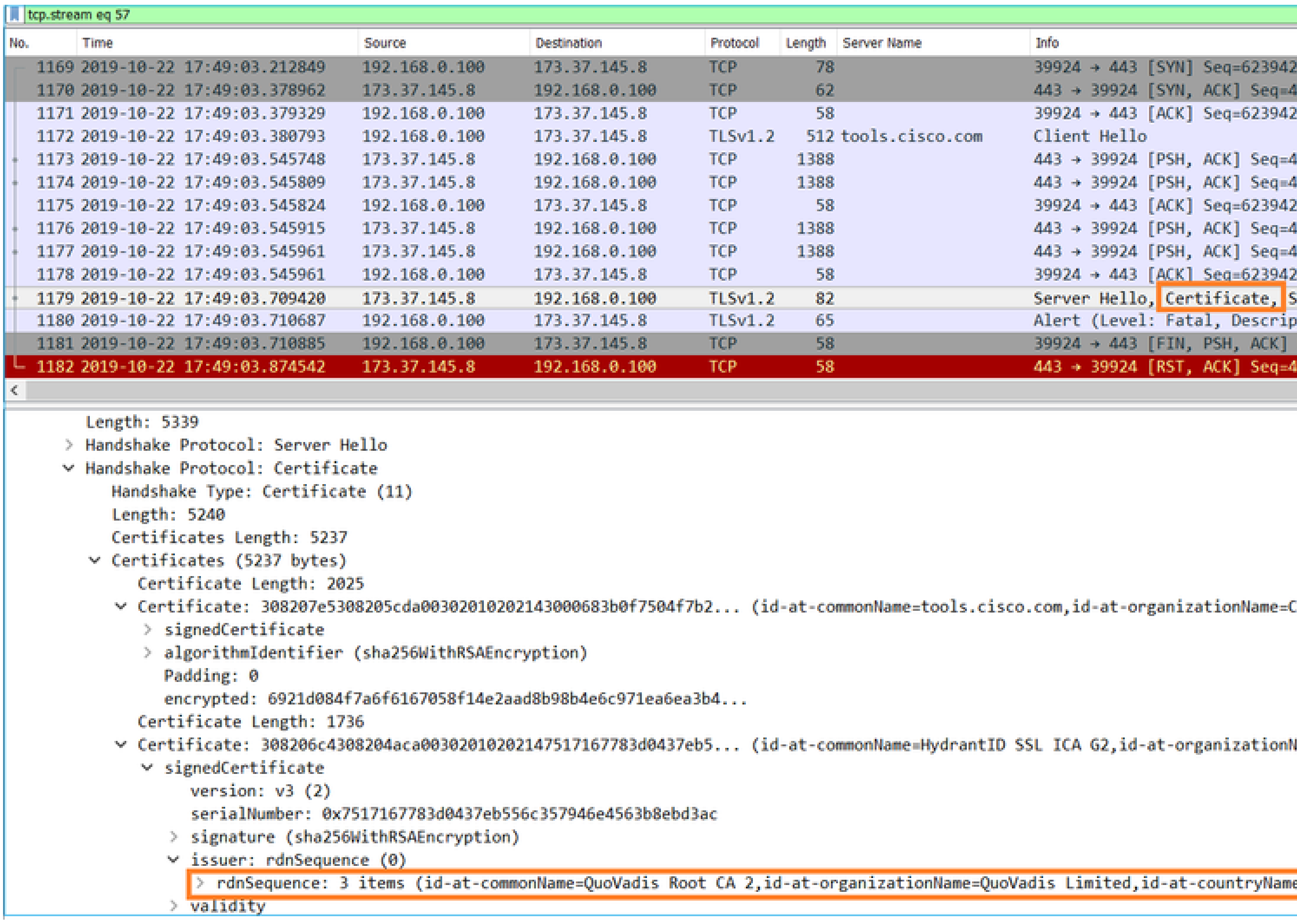


CAPI shows:



```
tcp.stream eq 57
No.   Time                        Source          Destination     Protocol  Length Server Name      Info
1221 2019-10-22 17:49:03.212681  192.168.0.100   173.37.145.8    TCP         74                     39924 → 443 [SYN] Seq=42
1222 2019-10-22 17:49:03.379023  173.37.145.8    192.168.0.100   TCP         58                     443 → 39924 [SYN, ACK] S
1223 2019-10-22 17:49:03.379298  192.168.0.100   173.37.145.8    TCP         54                     39924 → 443 [ACK] Seq=42
1224 2019-10-22 17:49:03.380336  192.168.0.100   173.37.145.8    TLSv1.2    571 tools.cisco.com     Client Hello
1225 2019-10-22 17:49:03.380732  173.37.145.8    192.168.0.100   TCP         54                     443 → 39924 [ACK] Seq=23
1226 2019-10-22 17:49:03.710092  173.37.145.8    192.168.0.100   TLSv1.2    150                     Server Hello
1227 2019-10-22 17:49:03.710092  173.37.145.8    192.168.0.100   TCP       1384                     443 → 39924 [PSH, ACK] S
1228 2019-10-22 17:49:03.710092  173.37.145.8    192.168.0.100   TLSv1.2    155                     Certificate
1229 2019-10-22 17:49:03.710107  173.37.145.8    192.168.0.100   TLSv1.2     63                     Server Hello Done
1230 2019-10-22 17:49:03.710412  192.168.0.100   173.37.145.8    TCP         54                     39924 → 443 [ACK] Seq=42
1231 2019-10-22 17:49:03.710519  192.168.0.100   173.37.145.8    TCP         54                     39924 → 443 [ACK] Seq=42
1232 2019-10-22 17:49:03.710519  192.168.0.100   173.37.145.8    TCP         54                     39924 → 443 [ACK] Seq=42
1233 2019-10-22 17:49:03.710534  192.168.0.100   173.37.145.8    TCP         54                     39924 → 443 [ACK] Seq=42
1234 2019-10-22 17:49:03.710626  192.168.0.100   173.37.145.8    TLSv1.2     61                     Alert (Level: Fatal, Des
1235 2019-10-22 17:49:03.710641  173.37.145.8    192.168.0.100   TCP         54                     443 → 39924 [ACK] Seq=23
1236 2019-10-22 17:49:03.710748  192.168.0.100   173.37.145.8    TCP         54                     39924 → 443 [RST, ACK] S
1237 2019-10-22 17:49:03.710870  192.168.0.100   173.37.145.8    TCP         54                     39924 → 443 [RST] Seq=42

      Length: 1426
    ˅ Handshake Protocol: Certificate
        Handshake Type: Certificate (11)
        Length: 1422
        Certificates Length: 1419
      ˅ Certificates (1419 bytes)
          Certificate Length: 1416
        ˅ Certificate: 308205843082046ca003020102020d00aa23af5d607e0000... (id-at-commonName=tools.cisco.com,id-at-organizationNa
          ˅ signedCertificate
              version: v3 (2)
              serialNumber: 0x00aa23af5d607e00002f423880
            > signature (sha256WithRSAEncryption)
            ˅ issuer: rdnSequence (0)
              ˅ rdnSequence: 3 items (id-at-commonName=FTD4100_MITM,id-at-organizationalUnitName=FTD_OU,id-at-organizationName=
                > RDNSequence item: 1 item (id-at-organizationName=FTD_O)
                > RDNSequence item: 1 item (id-at-organizationalUnitName=FTD_OU)
                > RDNSequence item: 1 item (id-at-commonName=FTD4100_MITM)
            > validity
```

CAPO shows:



```
tcp.stream eq 57
No.   Time                        Source          Destination     Protocol  Length Server Name      Info
1169 2019-10-22 17:49:03.212849  192.168.0.100   173.37.145.8    TCP         78                    39924 → 443 [SYN] Seq=623942
1170 2019-10-22 17:49:03.378962  173.37.145.8    192.168.0.100   TCP         62                    443 → 39924 [SYN, ACK] Seq=4
1171 2019-10-22 17:49:03.379329  192.168.0.100   173.37.145.8    TCP         58                    39924 → 443 [ACK] Seq=623942
1172 2019-10-22 17:49:03.380793  192.168.0.100   173.37.145.8    TLSv1.2    512 tools.cisco.com   Client Hello
1173 2019-10-22 17:49:03.545748  173.37.145.8    192.168.0.100   TCP       1388                    443 → 39924 [PSH, ACK] Seq=4
1174 2019-10-22 17:49:03.545809  173.37.145.8    192.168.0.100   TCP       1388                    443 → 39924 [PSH, ACK] Seq=4
1175 2019-10-22 17:49:03.545824  192.168.0.100   173.37.145.8    TCP         58                    39924 → 443 [ACK] Seq=623942
1176 2019-10-22 17:49:03.545915  173.37.145.8    192.168.0.100   TCP       1388                    443 → 39924 [PSH, ACK] Seq=4
1177 2019-10-22 17:49:03.545961  173.37.145.8    192.168.0.100   TCP       1388                    443 → 39924 [PSH, ACK] Seq=4
1178 2019-10-22 17:49:03.545961  192.168.0.100   173.37.145.8    TCP         58                    39924 → 443 [ACK] Seq=623942
1179 2019-10-22 17:49:03.709420  173.37.145.8    192.168.0.100   TLSv1.2     82                    Server Hello, Certificate, S
1180 2019-10-22 17:49:03.710687  192.168.0.100   173.37.145.8    TLSv1.2     65                    Alert (Level: Fatal, Descrip
1181 2019-10-22 17:49:03.710885  192.168.0.100   173.37.145.8    TCP         58                    39924 → 443 [FIN, PSH, ACK]
1182 2019-10-22 17:49:03.874542  173.37.145.8    192.168.0.100   TCP         58                    443 → 39924 [RST, ACK] Seq=4

      Length: 5339
    > Handshake Protocol: Server Hello
    v Handshake Protocol: Certificate
        Handshake Type: Certificate (11)
        Length: 5240
        Certificates Length: 5237
      v Certificates (5237 bytes)
          Certificate Length: 2025
        v Certificate: 308207e5308205cda00302010202143000683b0f7504f7b2... (id-at-commonName=tools.cisco.com,id-at-organizationName=C
          > signedCertificate
          > algorithmIdentifier (sha256WithRSAEncryption)
            Padding: 0
            encrypted: 6921d084f7a6f6167058f14e2aad8b98b4e6c971ea6ea3b4...
          Certificate Length: 1736
        v Certificate: 308206c4308204aca00302010202147517167783d0437eb5... (id-at-commonName=HydrantID SSL ICA G2,id-at-organizationN
          v signedCertificate
              version: v3 (2)
              serialNumber: 0x7517167783d0437eb556c357946e4563b8ebd3ac
            > signature (sha256WithRSAEncryption)
            v issuer: rdnSequence (0)
              > rdnSequence: 3 items (id-at-commonName=QuoVadis Root CA 2,id-at-organizationName=QuoVadis Limited,id-at-countryName
            > validity
```

These captures prove that the transit firewall modifies the server certificate (MITM)

Action 2. Check the device logs.

You can collect the FMC TS bundle as described in this document:

https://www.cisco.com/c/en/us/support/docs/security/sourcefire-defense-center/117663-technote-SourceFire-00.html

In this case, the **/dir-archives/var-log/process_stdout.log** file show messages like this:

<#root>

```
SOUT: 10-23 05:45:14 2019-10-23 05:45:36 sla[10068]: *Wed .967 UTC: CH-LIB-ERROR: ch_pf_curl_send_msg[49

failed to perform, err code 60, err string "SSL peer certificate or SSH remote key was not OK"

...
SOUT: 10-23 05:45:14 2019-10-23 05:45:36 sla[10068]: *Wed .967 UTC: CH-LIB-TRACE: ch_pf_curl_is_cert_iss

cert issue checking, ret 60, url "https://tools.cisco.com/its/
```

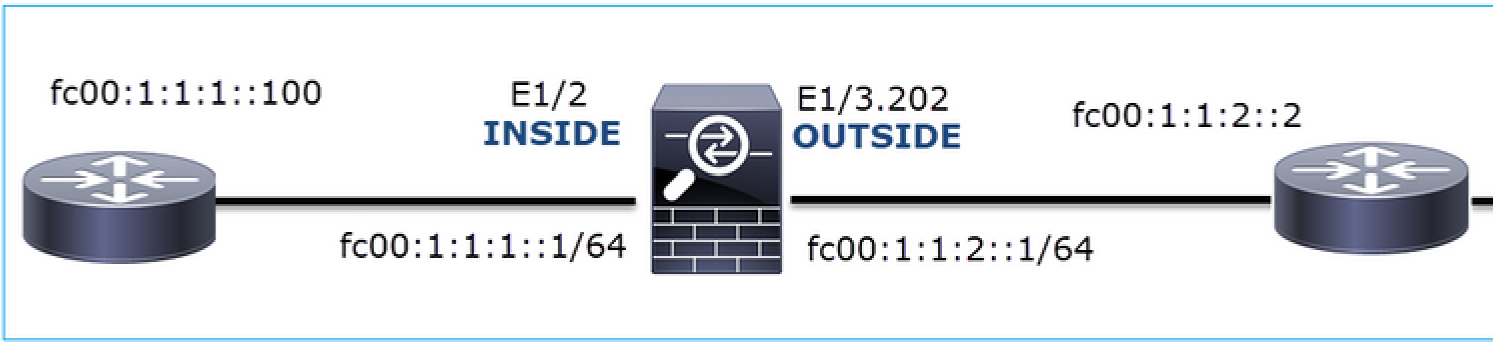
Recommended Solution

Disable the MITM for the specific flow so that FMC can successfully register to the Smart Licensing cloud.

Case 11. IPv6 Connectivity Problem

Problem Description: Internal hosts (located behind the firewallâ€™s INSIDE interface) cannot communicate with external hosts (hosts located behind firewallâ€™s OUTSIDE interface).

This image shows the topology:



Affected Flow:

Src IP: fc00:1:1:1::100

Dst IP: fc00:1:1:2::2

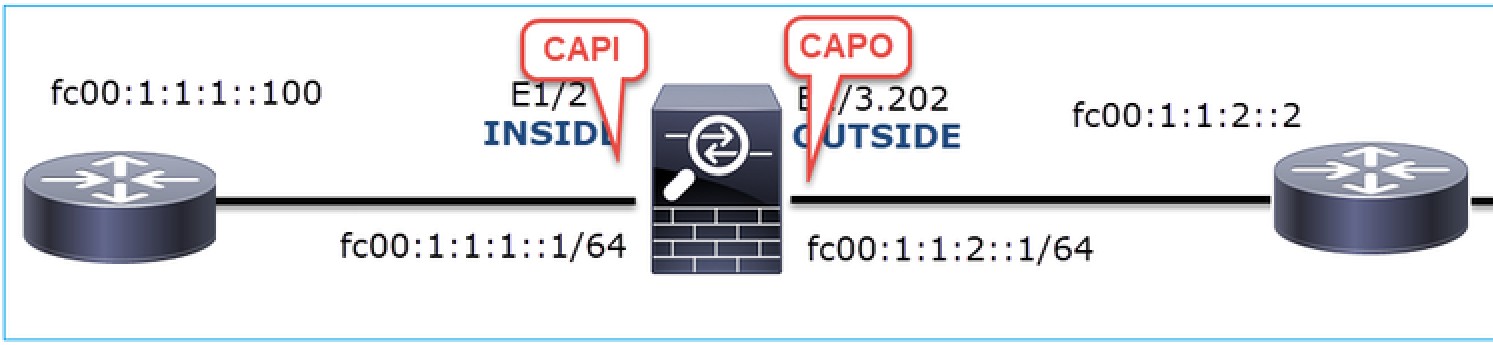
Protocol: any

Capture Analysis

Enable captures on FTD LINA engine.

<#root>

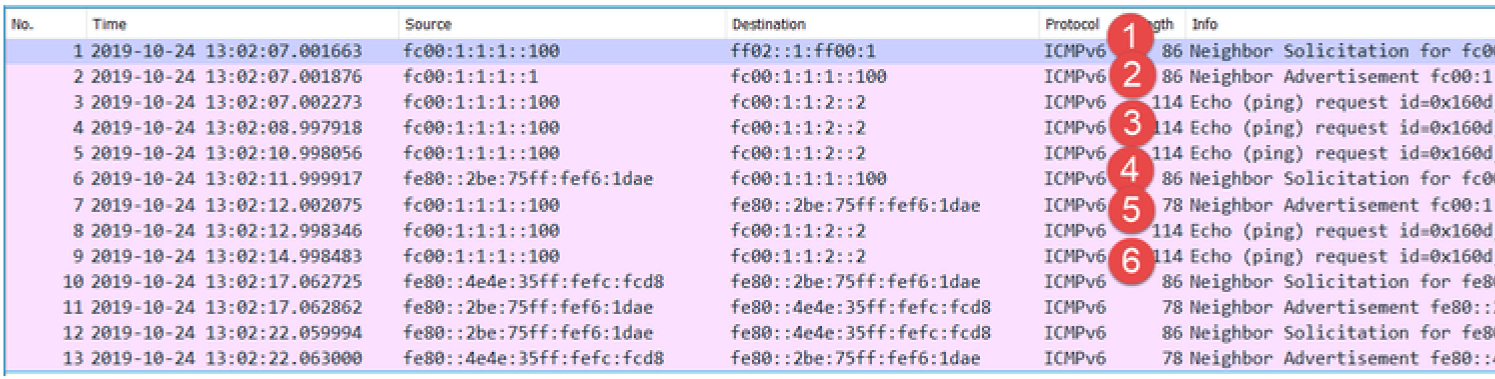
```
firepower#
capture CAPI int INSIDE match ip any6 any6
firepower#
capture CAPO int OUTSIDE match ip any6 any6
```



Captures - Non-functional Scenario

These captures were taken in parallel with an ICMP connectivity test from IP fc00:1:1:1::100 (inside router) to IP fc00:1:1:2::2 (upstream router).

The capture on firewall INSIDE interface contains:

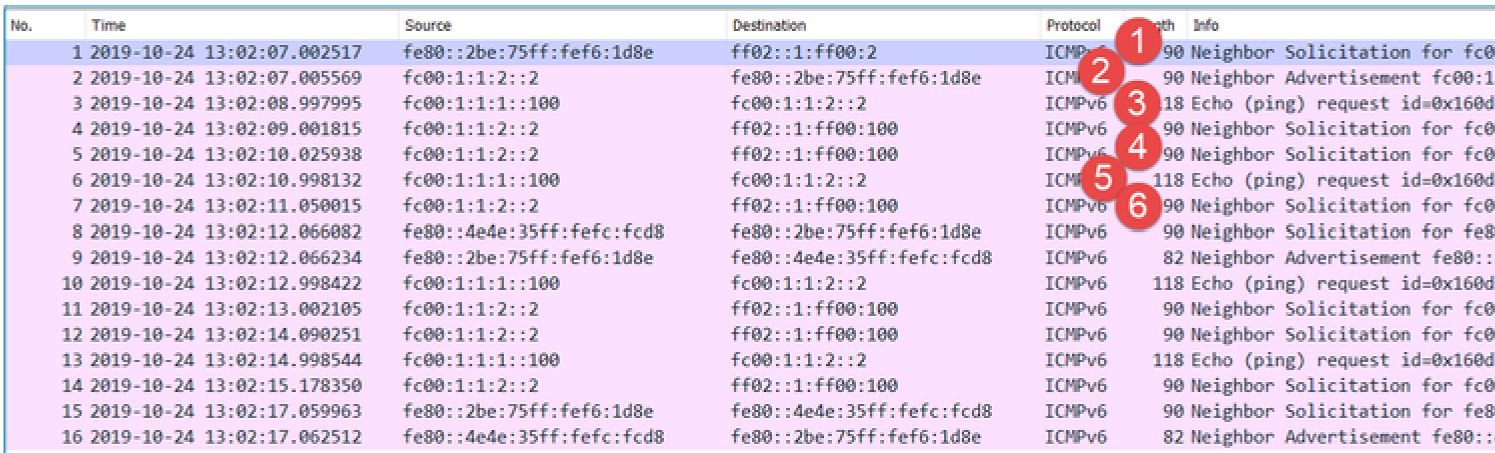


No.	Time	Source	Destination	Protocol	gth	Info
1	2019-10-24 13:02:07.001663	fc00:1:1:1::100	ff02::1:ff00:1	ICMPv6	86	Neighbor Solicitation for fc0
2	2019-10-24 13:02:07.001876	fc00:1:1:1::1	fc00:1:1:1::100	ICMPv6	86	Neighbor Advertisement fc00:1
3	2019-10-24 13:02:07.002273	fc00:1:1:1::100	fc00:1:1:2::2	ICMPv6	114	Echo (ping) request id=0x160d
4	2019-10-24 13:02:08.997918	fc00:1:1:1::100	fc00:1:1:2::2	ICMPv6	114	Echo (ping) request id=0x160d
5	2019-10-24 13:02:10.998056	fc00:1:1:1::100	fc00:1:1:2::2	ICMPv6	114	Echo (ping) request id=0x160d
6	2019-10-24 13:02:11.999917	fe80::2be:75ff:fef6:1dae	fc00:1:1:1::100	ICMPv6	86	Neighbor Solicitation for fc0
7	2019-10-24 13:02:12.002075	fc00:1:1:1::100	fe80::2be:75ff:fef6:1dae	ICMPv6	78	Neighbor Advertisement fc00:1
8	2019-10-24 13:02:12.998346	fc00:1:1:1::100	fc00:1:1:2::2	ICMPv6	114	Echo (ping) request id=0x160d
9	2019-10-24 13:02:14.998483	fc00:1:1:1::100	fc00:1:1:2::2	ICMPv6	114	Echo (ping) request id=0x160d
10	2019-10-24 13:02:17.062725	fe80::4e4e:35ff:fefc:fcd8	fe80::2be:75ff:fef6:1dae	ICMPv6	86	Neighbor Solicitation for fe8
11	2019-10-24 13:02:17.062862	fe80::2be:75ff:fef6:1dae	fe80::4e4e:35ff:fefc:fcd8	ICMPv6	78	Neighbor Advertisement fe80::
12	2019-10-24 13:02:22.059994	fe80::2be:75ff:fef6:1dae	fe80::4e4e:35ff:fefc:fcd8	ICMPv6	86	Neighbor Solicitation for fe8
13	2019-10-24 13:02:22.063000	fe80::4e4e:35ff:fefc:fcd8	fe80::2be:75ff:fef6:1dae	ICMPv6	78	Neighbor Advertisement fe80::

Key Points:

1. The router sends an IPv6 Neighbor Solicitation message and asks for the MAC address of the upstream device (IP fc00:1:1:1::1).
2. The firewall replies with an IPv6 Neighbor Advertisement.
3. The router sends an ICMP Echo Request.
4. The firewall sends an IPv6 Neighbor Solicitation message and asks for the MAC address of the downstream device (fc00:1:1:1::100).
5. The router replies with an IPv6 Neighbor Advertisement.
6. The router sends additional IPv6 ICMP Echo Requests.

The capture on firewall OUTSIDE interface contains:



No.	Time	Source	Destination	Protocol	gth	Info
1	2019-10-24 13:02:07.002517	fe80::2be:75ff:fef6:1d8e	ff02::1:ff00:2	ICMPv6	90	Neighbor Solicitation for fc0
2	2019-10-24 13:02:07.005569	fc00:1:1:2::2	fe80::2be:75ff:fef6:1d8e	ICMPv6	90	Neighbor Advertisement fc00:1
3	2019-10-24 13:02:08.997995	fc00:1:1:1::100	fc00:1:1:2::2	ICMPv6	118	Echo (ping) request id=0x160d
4	2019-10-24 13:02:09.001815	fc00:1:1:2::2	ff02::1:ff00:100	ICMPv6	90	Neighbor Solicitation for fc0
5	2019-10-24 13:02:10.025938	fc00:1:1:2::2	ff02::1:ff00:100	ICMPv6	90	Neighbor Solicitation for fc0
6	2019-10-24 13:02:10.998132	fc00:1:1:1::100	fc00:1:1:2::2	ICMPv6	118	Echo (ping) request id=0x160d
7	2019-10-24 13:02:11.050015	fc00:1:1:2::2	ff02::1:ff00:100	ICMPv6	90	Neighbor Solicitation for fc0
8	2019-10-24 13:02:12.066082	fe80::4e4e:35ff:fefc:fcd8	fe80::2be:75ff:fef6:1d8e	ICMPv6	90	Neighbor Solicitation for fe8
9	2019-10-24 13:02:12.066234	fe80::2be:75ff:fef6:1d8e	fe80::4e4e:35ff:fefc:fcd8	ICMPv6	82	Neighbor Advertisement fe80::
10	2019-10-24 13:02:12.998422	fc00:1:1:1::100	fc00:1:1:2::2	ICMPv6	118	Echo (ping) request id=0x160d
11	2019-10-24 13:02:13.002105	fc00:1:1:2::2	ff02::1:ff00:100	ICMPv6	90	Neighbor Solicitation for fc0
12	2019-10-24 13:02:14.090251	fc00:1:1:2::2	ff02::1:ff00:100	ICMPv6	90	Neighbor Solicitation for fc0
13	2019-10-24 13:02:14.998544	fc00:1:1:1::100	fc00:1:1:2::2	ICMPv6	118	Echo (ping) request id=0x160d
14	2019-10-24 13:02:15.178350	fc00:1:1:2::2	ff02::1:ff00:100	ICMPv6	90	Neighbor Solicitation for fc0
15	2019-10-24 13:02:17.059963	fe80::2be:75ff:fef6:1d8e	fe80::4e4e:35ff:fefc:fcd8	ICMPv6	90	Neighbor Solicitation for fe8
16	2019-10-24 13:02:17.062512	fe80::4e4e:35ff:fefc:fcd8	fe80::2be:75ff:fef6:1d8e	ICMPv6	82	Neighbor Advertisement fe80::

Key Points:

1. The firewall sends an IPv6 Neighbor Solicitation message which asks for the MAC address of the upstream device (IP fc00:1:1:2::2).
2. The router replies with an IPv6 Neighbor Advertisement.
3. The firewall sends an IPv6 ICMP Echo Request.
4. The upstream device (router fc00:1:1:2::2) sends an IPv6 Neighbor Solicitation message which asks for the MAC address of the IPv6 address fc00:1:1:1::100.
5. The firewall sends an additional IPv6 ICMP Echo Request.
6. The upstream router sends an additional IPv6 Neighbor Solicitation message which asks for the MAC address of the IPv6 address fc00:1:1:1::100.

Point 4 is very interesting. Normally the upstream router asks for the MAC of the firewall OUTSIDE interface (fc00:1:1:2::2), but instead, it asks for the fc00:1:1:1::100. This is an indication of a misconfiguration.

Recommended Actions

The actions listed in this section have as a goal to further narrow down the issue.

Action 1. Check the IPv6 Neighbor Table.

The firewall IPv6 Neighbor Table is properly populated.

<#root>

```
firepower#
```

`show ipv6 neighbor | i fc00`

```
fc00:1:1:2::2                              58 4c4e.35fc.fcd8  STALE OUTSIDE
fc00:1:1:1::100                            58 4c4e.35fc.fcd8  STALE INSIDE
```

Action 2. Check the IPv6 Configuration.

This is the firewall configuration.

<#root>

```
firewall#
```

`show run int e1/2`

```
!
interface Ethernet1/2
 nameif INSIDE
 cts manual
  propagate sgt preserve-untag
  policy static sgt disabled trusted
 security-level 0
 ip address 192.168.0.1 255.255.255.0
 ipv6 address
```

`fc00:1:1:1::1/64`

```
 ipv6 enable
```

```
firewall#
```

```
show run int e1/3.202
```

```
!
interface Ethernet1/3.202
 vlan 202
 nameif OUTSIDE
 cts manual
  propagate sgt preserve-untag
  policy static sgt disabled trusted
 security-level 0
 ip address 192.168.103.96 255.255.255.0
 ipv6 address
```

```
fc00:1:1:2::1/64
```

```
 ipv6 enable
```

The upstream device configuration reveals the misconfiguration:

<#root>

```
Router#
```

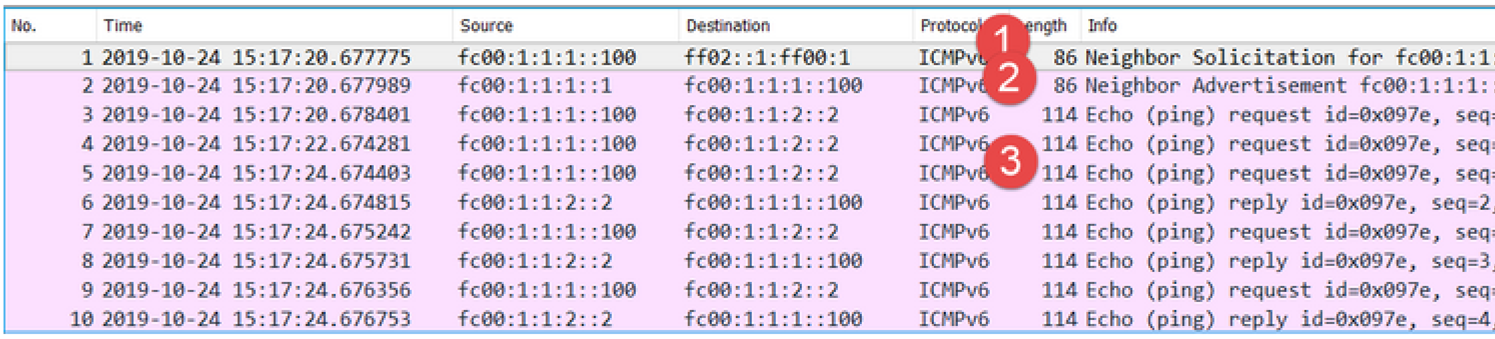
```
show run interface g0/0.202
```

```
!
interface GigabitEthernet0/0.202
 encapsulation dot1Q 202
 vrf forwarding VRF202
 ip address 192.168.2.72 255.255.255.0
 ipv6 address FC00:1:1:2::2
```

```
/48
```

Captures - Functional Scenario

The subnet mask change (from /48 to /64) fixed the issue. This is the CAPI capture in the functional scenario.



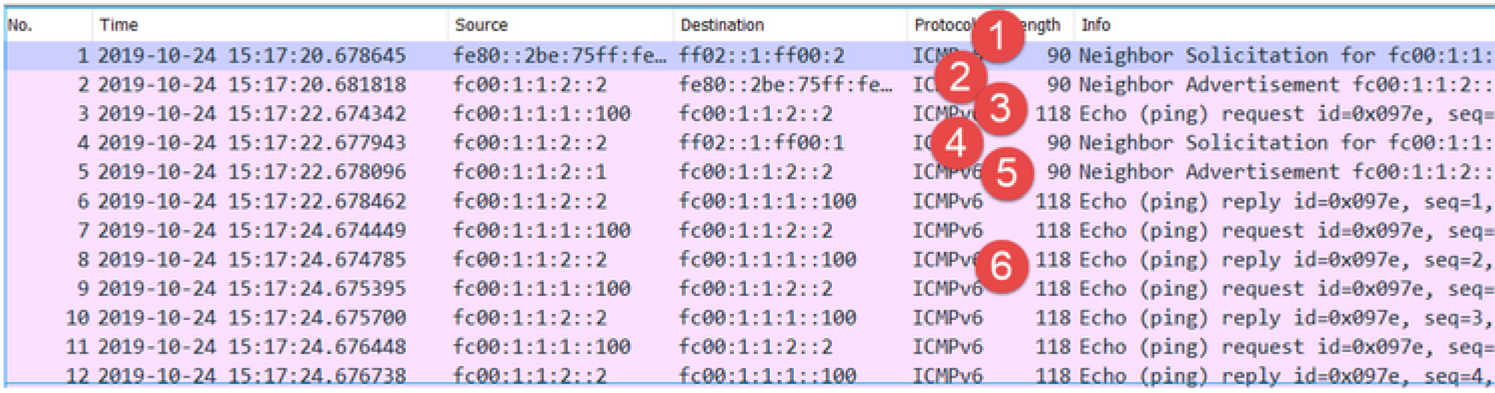
```
No.   Time                          Source              Destination         Protocol Length Info
    1 2019-10-24 15:17:20.677775   fc00:1:1:1::100     ff02::1:ff00:1      ICMPv6    86 Neighbor Solicitation for fc00:1:1
    2 2019-10-24 15:17:20.677989   fc00:1:1:1::1       fc00:1:1:1::100     ICMPv6    86 Neighbor Advertisement fc00:1:1:1:
    3 2019-10-24 15:17:20.678401   fc00:1:1:1::100     fc00:1:1:2::2       ICMPv6   114 Echo (ping) request id=0x097e, seq
    4 2019-10-24 15:17:22.674281   fc00:1:1:1::100     fc00:1:1:2::2       ICMPv6   114 Echo (ping) request id=0x097e, seq
    5 2019-10-24 15:17:24.674403   fc00:1:1:1::100     fc00:1:1:2::2       ICMPv6   114 Echo (ping) request id=0x097e, seq
    6 2019-10-24 15:17:24.674815   fc00:1:1:2::2       fc00:1:1:1::100     ICMPv6   114 Echo (ping) reply id=0x097e, seq=2
    7 2019-10-24 15:17:24.675242   fc00:1:1:1::100     fc00:1:1:2::2       ICMPv6   114 Echo (ping) request id=0x097e, seq
    8 2019-10-24 15:17:24.675731   fc00:1:1:2::2       fc00:1:1:1::100     ICMPv6   114 Echo (ping) reply id=0x097e, seq=3
    9 2019-10-24 15:17:24.676356   fc00:1:1:1::100     fc00:1:1:2::2       ICMPv6   114 Echo (ping) request id=0x097e, seq
   10 2019-10-24 15:17:24.676753   fc00:1:1:2::2       fc00:1:1:1::100     ICMPv6   114 Echo (ping) reply id=0x097e, seq=4
```

Key Point:

1. The router sends an IPv6 Neighbor Solicitation message which asks for the MAC address of the

upstream device (IP fc00:1:1:1::1).
2. The firewall replies with an IPv6 Neighbor Advertisement.
3. The router sends ICMP Echo Requests and gets Echo Replies.

CAPO contents:



```
No.  Time                        Source              Destination         Protocol Length Info
   1 2019-10-24 15:17:20.678645  fe80::2be:75ff:fe…  ff02::1:ff00:2      ICMP         90 Neighbor Solicitation for fc00:1:1:
   2 2019-10-24 15:17:20.681818  fc00:1:1:2::2       fe80::2be:75ff:fe…  IC           90 Neighbor Advertisement fc00:1:1:2::
   3 2019-10-24 15:17:22.674342  fc00:1:1:1::100     fc00:1:1:2::2       ICMPv       118 Echo (ping) request id=0x097e, seq=
   4 2019-10-24 15:17:22.677943  fc00:1:1:2::2       ff02::1:ff00:1      IC           90 Neighbor Solicitation for fc00:1:1:
   5 2019-10-24 15:17:22.678096  fc00:1:1:2::1       fc00:1:1:2::2       ICMPv6       90 Neighbor Advertisement fc00:1:1:2::
   6 2019-10-24 15:17:22.678462  fc00:1:1:2::2       fc00:1:1:1::100     ICMPv6      118 Echo (ping) reply id=0x097e, seq=1,
   7 2019-10-24 15:17:24.674449  fc00:1:1:1::100     fc00:1:1:2::2       ICMPv6      118 Echo (ping) request id=0x097e, seq=
   8 2019-10-24 15:17:24.674785  fc00:1:1:2::2       fc00:1:1:1::100     ICMPv6      118 Echo (ping) reply id=0x097e, seq=2,
   9 2019-10-24 15:17:24.675395  fc00:1:1:1::100     fc00:1:1:2::2       ICMPv6      118 Echo (ping) request id=0x097e, seq=
  10 2019-10-24 15:17:24.675700  fc00:1:1:2::2       fc00:1:1:1::100     ICMPv6      118 Echo (ping) reply id=0x097e, seq=3,
  11 2019-10-24 15:17:24.676448  fc00:1:1:1::100     fc00:1:1:2::2       ICMPv6      118 Echo (ping) request id=0x097e, seq=
  12 2019-10-24 15:17:24.676738  fc00:1:1:2::2       fc00:1:1:1::100     ICMPv6      118 Echo (ping) reply id=0x097e, seq=4,
```

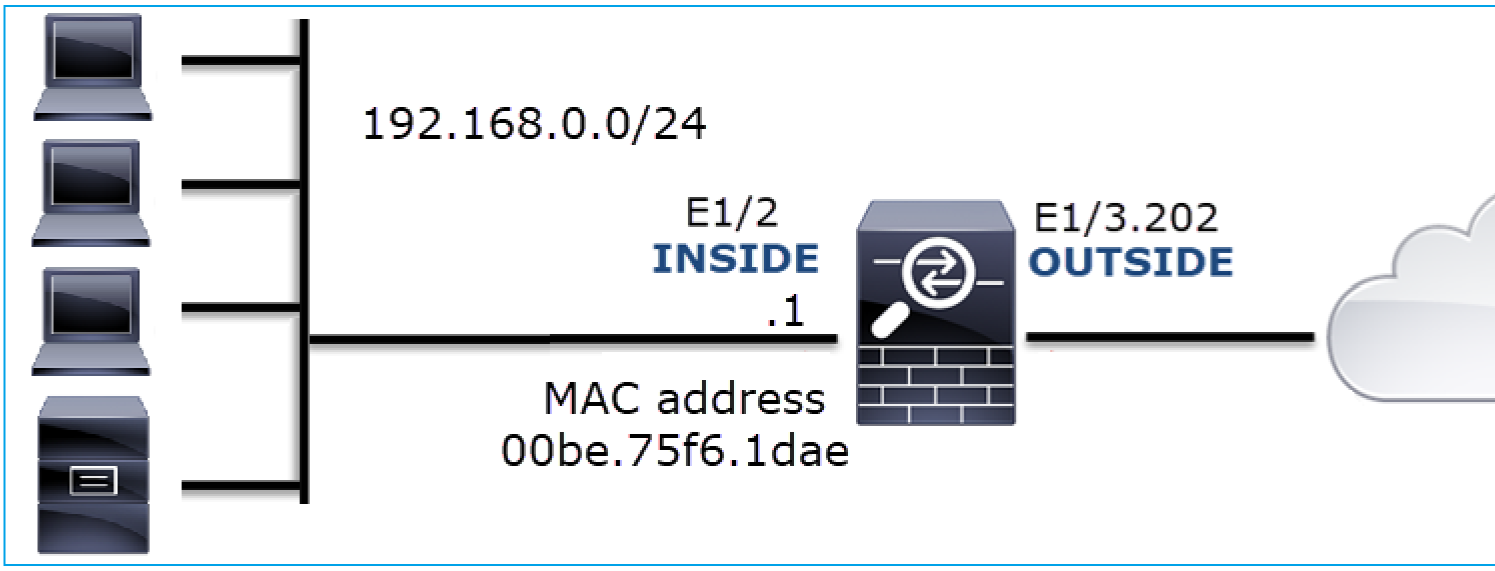
Key Points:

1. The firewall sends an IPv6 Neighbor Solicitation message which asks for the MAC address of the upstream device (IP fc00:1:1:2::2).
2. The firewall replies with an IPv6 Neighbor Advertisement.
3. The firewall sends an ICMP Echo Request.
4. The router sends an IPv6 Neighbor Solicitation message which asks for the MAC address of the downstream device (IP fc00:1:1:1::1).
5. The firewall replies with an IPv6 Neighbor Advertisement.
6. The firewall sends ICMP Echo Requests and gets Echo Replies.

Case 12. Intermittent Connectivity Problem (ARP Poisoning)

Problem Description: Internal hosts (192.168.0.x/24) have intermittent connectivity issues with hosts in the same subnet

This image shows the topology:



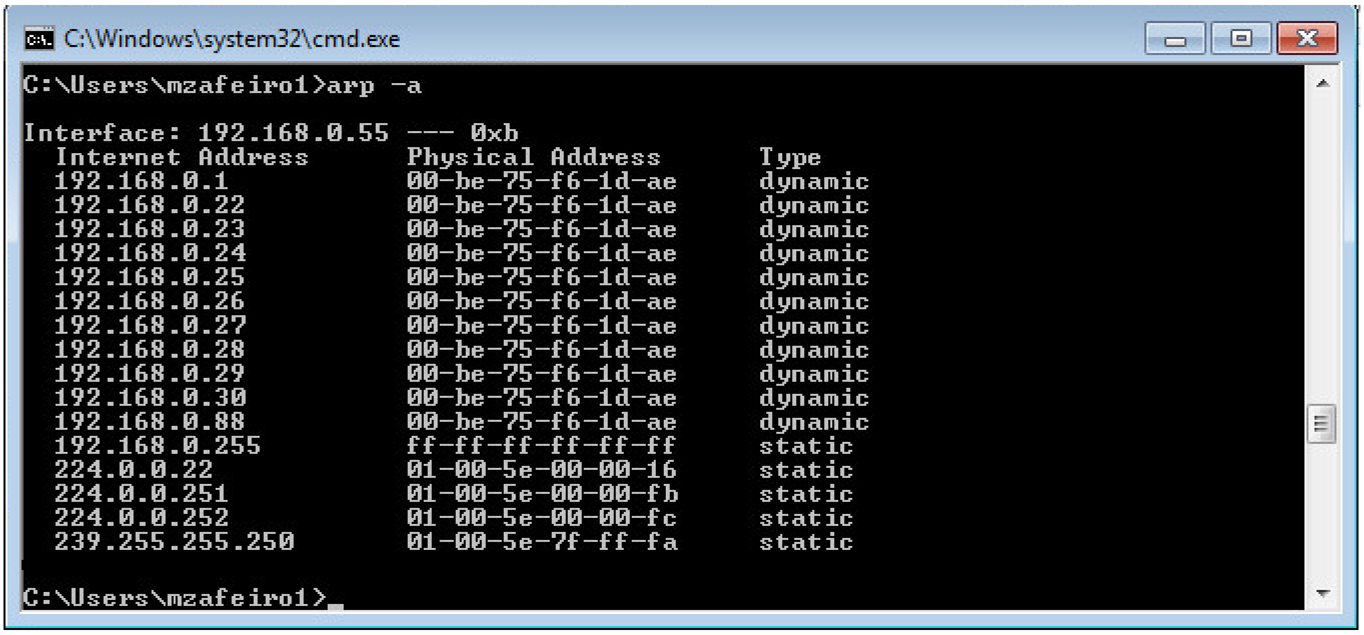
Affected Flow:

Src IP: 192.168.0.x/24

Dst IP: 192.168.0.x/24

Protocol: any

The ARP cache of an internal host seems to be poisoned:



```
C:\Users\mzafeiro1>arp -a

Interface: 192.168.0.55 --- 0xb
  Internet Address      Physical Address      Type
  192.168.0.1           00-be-75-f6-1d-ae     dynamic
  192.168.0.22          00-be-75-f6-1d-ae     dynamic
  192.168.0.23          00-be-75-f6-1d-ae     dynamic
  192.168.0.24          00-be-75-f6-1d-ae     dynamic
  192.168.0.25          00-be-75-f6-1d-ae     dynamic
  192.168.0.26          00-be-75-f6-1d-ae     dynamic
  192.168.0.27          00-be-75-f6-1d-ae     dynamic
  192.168.0.28          00-be-75-f6-1d-ae     dynamic
  192.168.0.29          00-be-75-f6-1d-ae     dynamic
  192.168.0.30          00-be-75-f6-1d-ae     dynamic
  192.168.0.88          00-be-75-f6-1d-ae     dynamic
  192.168.0.255         ff-ff-ff-ff-ff-ff     static
  224.0.0.22            01-00-5e-00-00-16     static
  224.0.0.251           01-00-5e-00-00-fb     static
  224.0.0.252           01-00-5e-00-00-fc     static
  239.255.255.250       01-00-5e-7f-ff-fa     static

C:\Users\mzafeiro1>_
```

Capture Analysis

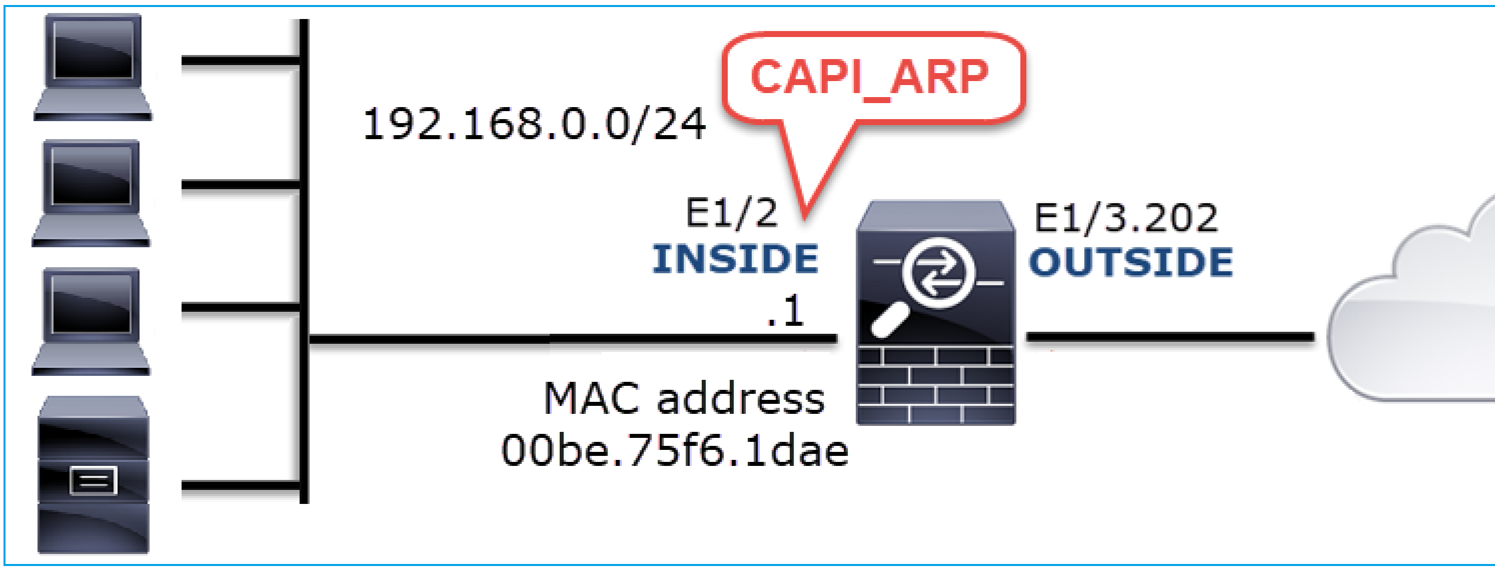
Enable a capture on FTD LINA engine

This capture only captures ARP packets on the INSIDE interface:

<#root>

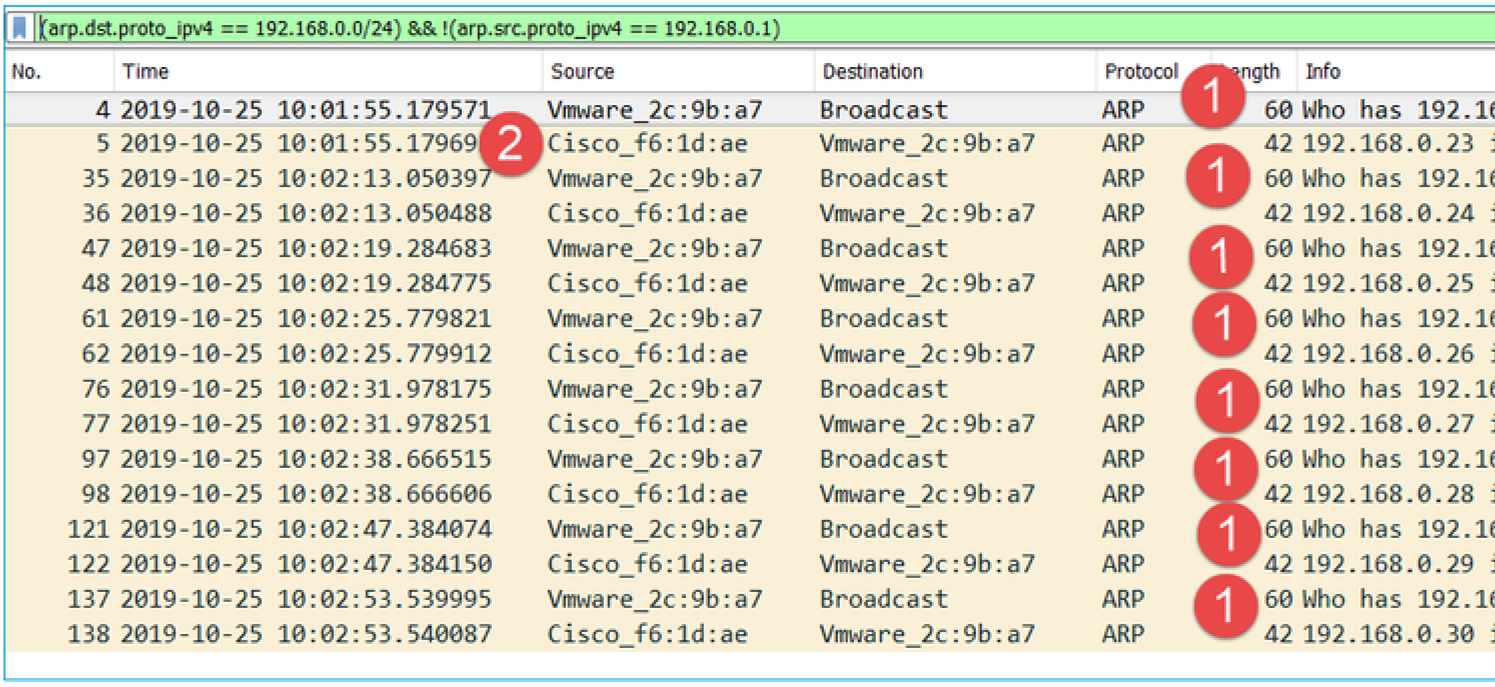
```
firepower#
```

```
capture CAPI_ARP interface INSIDE ethernet-type arp
```



Captures - Non-functional Scenario:

The capture on the firewall INSIDE interface contains.



```
(arp.dst.proto_ipv4 == 192.168.0.0/24) && !(arp.src.proto_ipv4 == 192.168.0.1)

No.   Time                        Source            Destination       Protocol  Length Info
    4 2019-10-25 10:01:55.179571  Vmware_2c:9b:a7   Broadcast         ARP    1    60 Who has 192.16
    5 2019-10-25 10:01:55.17969 2 Cisco_f6:1d:ae    Vmware_2c:9b:a7   ARP         42 192.168.0.23 i
   35 2019-10-25 10:02:13.050397  Vmware_2c:9b:a7   Broadcast         ARP    1    60 Who has 192.16
   36 2019-10-25 10:02:13.050488  Cisco_f6:1d:ae    Vmware_2c:9b:a7   ARP         42 192.168.0.24 i
   47 2019-10-25 10:02:19.284683  Vmware_2c:9b:a7   Broadcast         ARP    1    60 Who has 192.16
   48 2019-10-25 10:02:19.284775  Cisco_f6:1d:ae    Vmware_2c:9b:a7   ARP         42 192.168.0.25 i
   61 2019-10-25 10:02:25.779821  Vmware_2c:9b:a7   Broadcast         ARP    1    60 Who has 192.16
   62 2019-10-25 10:02:25.779912  Cisco_f6:1d:ae    Vmware_2c:9b:a7   ARP         42 192.168.0.26 i
   76 2019-10-25 10:02:31.978175  Vmware_2c:9b:a7   Broadcast         ARP    1    60 Who has 192.16
   77 2019-10-25 10:02:31.978251  Cisco_f6:1d:ae    Vmware_2c:9b:a7   ARP         42 192.168.0.27 i
   97 2019-10-25 10:02:38.666515  Vmware_2c:9b:a7   Broadcast         ARP    1    60 Who has 192.16
   98 2019-10-25 10:02:38.666606  Cisco_f6:1d:ae    Vmware_2c:9b:a7   ARP         42 192.168.0.28 i
  121 2019-10-25 10:02:47.384074  Vmware_2c:9b:a7   Broadcast         ARP    1    60 Who has 192.16
  122 2019-10-25 10:02:47.384150  Cisco_f6:1d:ae    Vmware_2c:9b:a7   ARP         42 192.168.0.29 i
  137 2019-10-25 10:02:53.539995  Vmware_2c:9b:a7   Broadcast         ARP    1    60 Who has 192.16
  138 2019-10-25 10:02:53.540087  Cisco_f6:1d:ae    Vmware_2c:9b:a7   ARP         42 192.168.0.30 i
```

Key Points:

1. The firewall receives various ARP requests for IPs within 192.168.0.x/24 network
2. The firewall replies to all of them (proxy-ARP) with its own MAC address

Recommended Actions

The actions listed in this section have as a goal to further narrow down the issue.

Action 1. Check the NAT configuration.

With regard to the NAT configuration, there are cases where the **no-proxy-arp** keyword can prevent the earlier behavior:

<#root>

```
firepower#

show run nat

nat (INSIDE,OUTSIDE) source static NET_1.1.1.0 NET_2.2.2.0 destination static NET_192.168.0.0 NET_4.4.4.

no-proxy-arp
```

Action 2. Disable the proxy-arp functionality on the firewall interface.

If the â€˜no-proxy-arpâ€™ keyword does not solve the problem, try to disable proxy ARP on the interface itself. In case of FTD, at the time of this writing, you have to use FlexConfig and deploy the command (specify the appropriate interface name).

```
sysopt noproxyarp INSIDE
```

Case 13. Identify SNMP Object Identifiers (OIDs) that cause CPU Hogs

This case demonstrates how certain SNMP OIDs for memory polling were identified as the root cause of CPU hogs (performance issue) based on the analysis of SNMP version 3 (SNMPv3) packet captures.

Problem Description: Overruns on data interfaces continuously increase. Further research revealed that there are also CPU hogs (caused by the SNMP process) which are the root cause of the interface overruns.

Next step in the troubleshoot process was to identify the root cause of the CPU hogs caused by the SNMP process and in particular, narrow down the scope of the issue to identify the SNMP Object Identifiers (OID) which, when polled, could potentially result in CPU hogs.

Currently, the FTD LINA engine does not provide a 'show' command for SNMP OIDs that are polled in real-time.

The list of SNMP OIDs for polling can be retrieved from the SNMP monitoring tool, however, in this case, there were these preventive factors:

- The FTD administrator did not have access to the SNMP monitoring tool
- SNMP version 3 with authentication and data encryption for privacy was configured on FTD

Capture Analysis

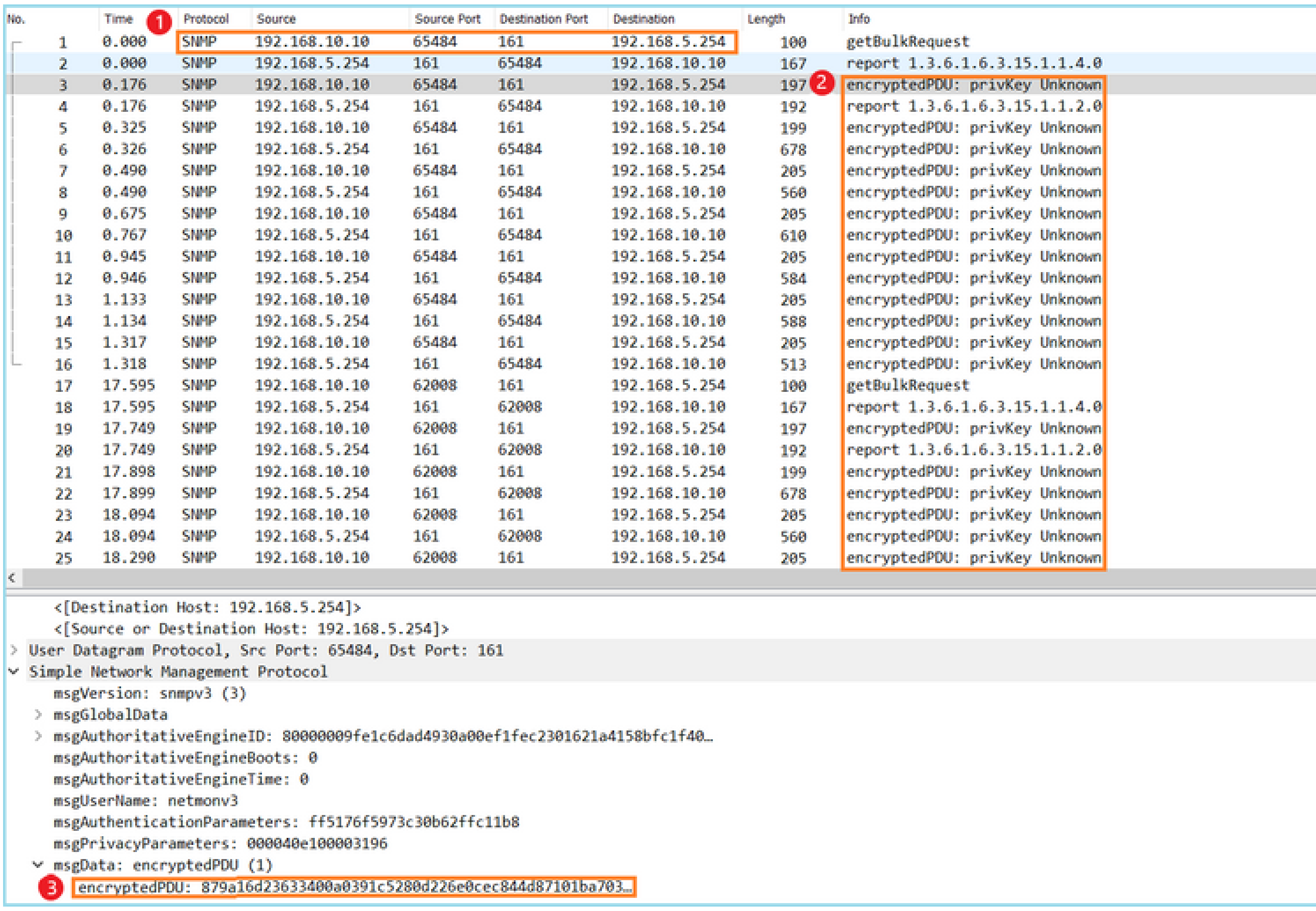
Since the FTD administrator had the credentials for the SNMP version 3 authentication and data encryption, this action plan was proposed:

1. Take SNMP packet captures
2. Save the captures and use Wireshark SNMP protocol preferences to specify the SNMP version 3 credentials to decrypt the SNMP version 3 packets. The decrypted captures are used for the analysis and retrieval of SNMP OIDs

Configure SNMP packet captures on the interface that is used in snmp-server host configuration:

<#root>

```
firepower#

show run snmp-server  | include host

snmp-server host management 192.168.10.10 version 3 netmonv3


firepower#

show ip address management

System IP Address:
Interface                Name                     IP address      Subnet mask     Method
Management0/0            management               192.168.5.254   255.255.255.0   CONFIG
Current IP Address:
Interface                Name                     IP address      Subnet mask     Method
Management0/0            management               192.168.5.254   255.255.255.0   CONFIG

firepower#

capture capsnmp interface management buffer 10000000 match udp host 192.168.10.10  host 192.168.5.254 ec


firepower#

show capture capsnmp


capture capsnmp type raw-data buffer 10000000 interface outside [Capturing -

9512

 bytes]
  match udp host 192.168.10.10  host 192.168.5.254 eq snmp
```



```
No.  Time    Protocol  Source         Source Port  Destination Port  Destination    Length  Info
1    0.000   SNMP      192.168.10.10  65484        161               192.168.5.254  100     getBulkRequest
2    0.000   SNMP      192.168.5.254  161          65484             192.168.10.10  167     report 1.3.6.1.6.3.15.1.1.4.0
3    0.176   SNMP      192.168.10.10  65484        161               192.168.5.254  197     encryptedPDU: privKey Unknown
4    0.176   SNMP      192.168.5.254  161          65484             192.168.10.10  192     report 1.3.6.1.6.3.15.1.1.2.0
5    0.325   SNMP      192.168.10.10  65484        161               192.168.5.254  199     encryptedPDU: privKey Unknown
6    0.326   SNMP      192.168.5.254  161          65484             192.168.10.10  678     encryptedPDU: privKey Unknown
7    0.490   SNMP      192.168.10.10  65484        161               192.168.5.254  205     encryptedPDU: privKey Unknown
8    0.490   SNMP      192.168.5.254  161          65484             192.168.10.10  560     encryptedPDU: privKey Unknown
9    0.675   SNMP      192.168.10.10  65484        161               192.168.5.254  205     encryptedPDU: privKey Unknown
10   0.767   SNMP      192.168.5.254  161          65484             192.168.10.10  610     encryptedPDU: privKey Unknown
11   0.945   SNMP      192.168.10.10  65484        161               192.168.5.254  205     encryptedPDU: privKey Unknown
12   0.946   SNMP      192.168.5.254  161          65484             192.168.10.10  584     encryptedPDU: privKey Unknown
13   1.133   SNMP      192.168.10.10  65484        161               192.168.5.254  205     encryptedPDU: privKey Unknown
14   1.134   SNMP      192.168.5.254  161          65484             192.168.10.10  588     encryptedPDU: privKey Unknown
15   1.317   SNMP      192.168.10.10  65484        161               192.168.5.254  205     encryptedPDU: privKey Unknown
16   1.318   SNMP      192.168.5.254  161          65484             192.168.10.10  513     encryptedPDU: privKey Unknown
17   17.595  SNMP      192.168.10.10  62008        161               192.168.5.254  100     getBulkRequest
18   17.595  SNMP      192.168.5.254  161          62008             192.168.10.10  167     report 1.3.6.1.6.3.15.1.1.4.0
19   17.749  SNMP      192.168.10.10  62008        161               192.168.5.254  197     encryptedPDU: privKey Unknown
20   17.749  SNMP      192.168.5.254  161          62008             192.168.10.10  192     report 1.3.6.1.6.3.15.1.1.2.0
21   17.898  SNMP      192.168.10.10  62008        161               192.168.5.254  199     encryptedPDU: privKey Unknown
22   17.899  SNMP      192.168.5.254  161          62008             192.168.10.10  678     encryptedPDU: privKey Unknown
23   18.094  SNMP      192.168.10.10  62008        161               192.168.5.254  205     encryptedPDU: privKey Unknown
24   18.094  SNMP      192.168.5.254  161          62008             192.168.10.10  560     encryptedPDU: privKey Unknown
25   18.290  SNMP      192.168.10.10  62008        161               192.168.5.254  205     encryptedPDU: privKey Unknown

    <[Destination Host: 192.168.5.254]>
    <[Source or Destination Host: 192.168.5.254]>
> User Datagram Protocol, Src Port: 65484, Dst Port: 161
v Simple Network Management Protocol
    msgVersion: snmpv3 (3)
  > msgGlobalData
  > msgAuthoritativeEngineID: 80000009fe1c6dad4930a00ef1fec2301621a4158bfc1f40…
    msgAuthoritativeEngineBoots: 0
    msgAuthoritativeEngineTime: 0
    msgUserName: netmonv3
    msgAuthenticationParameters: ff5176f5973c30b62ffc11b8
    msgPrivacyParameters: 000040e100003196
  v msgData: encryptedPDU (1)
      encryptedPDU: 879a16d23633400a0391c5280d226e0cec844d87101ba703…
```

Key points:

1. SNMP source and destination addresses/ports.
2. The SNMP protocol PDU could not be decoded because privKey is unknown to Wireshark.
3. The value of the encryptedPDU primitive.

Recommended Actions

The actions listed in this section have as a goal to further narrow down the issue.

Action 1. Decrypt the SNMP captures.

Save the captures and edit the Wireshark SNMP protocol preferences to specify the SNMP version 3 credentials to decrypt the packets.

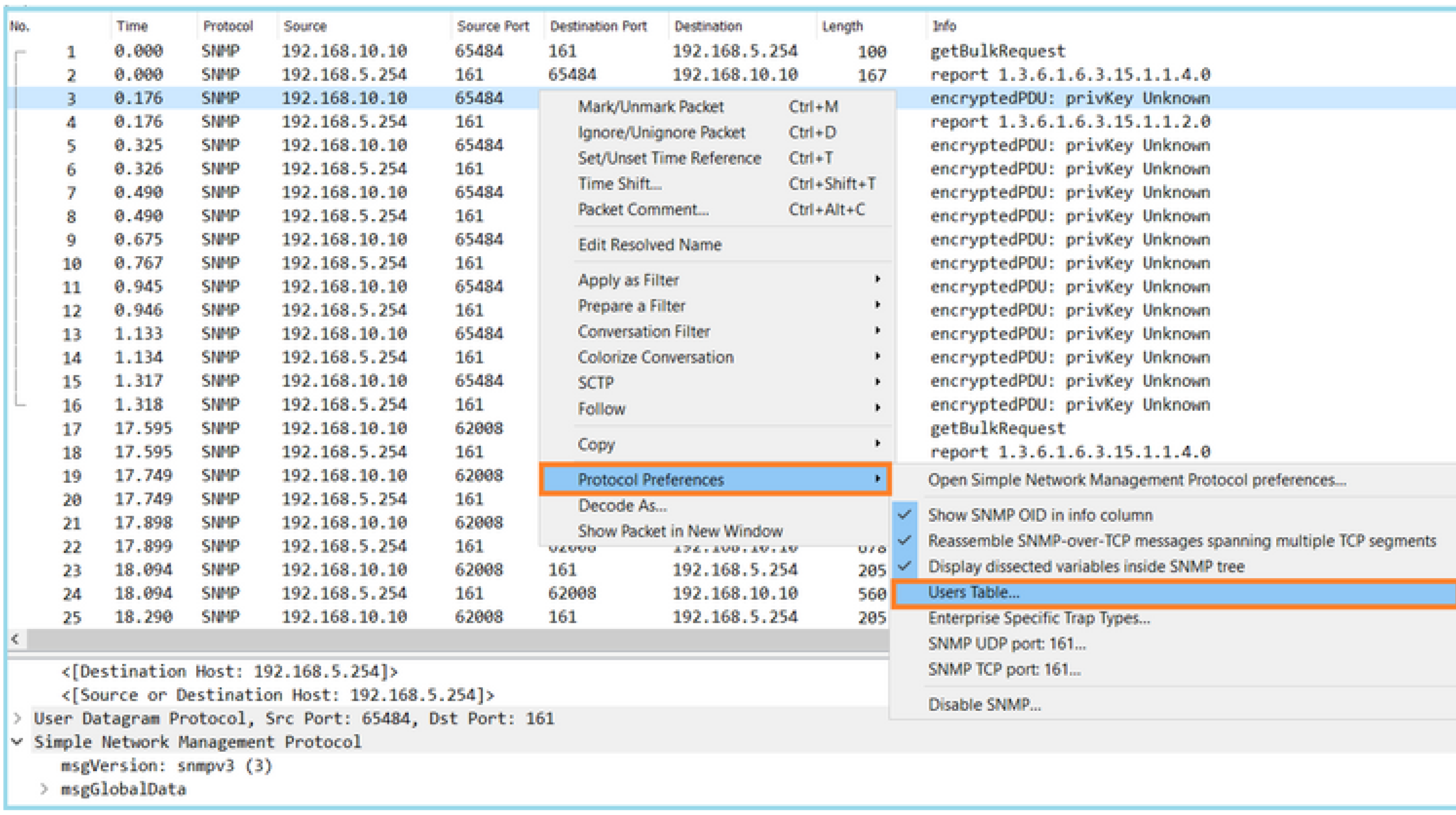
<#root>

```
firepower#
copy /pcap capture: tftp:


Source capture name [capsnmp]?

Address or name of remote host []? 192.168.10.253
```

```
Destination filename [capsnmp]? capsnmp.pcap
```

```
!!!!!!
64 packets copied in 0.40 secs
```

Open the capture file on Wireshark, select an SNMP packet and navigate to **Protocol Preferences > Users Table**, as shown in the image:

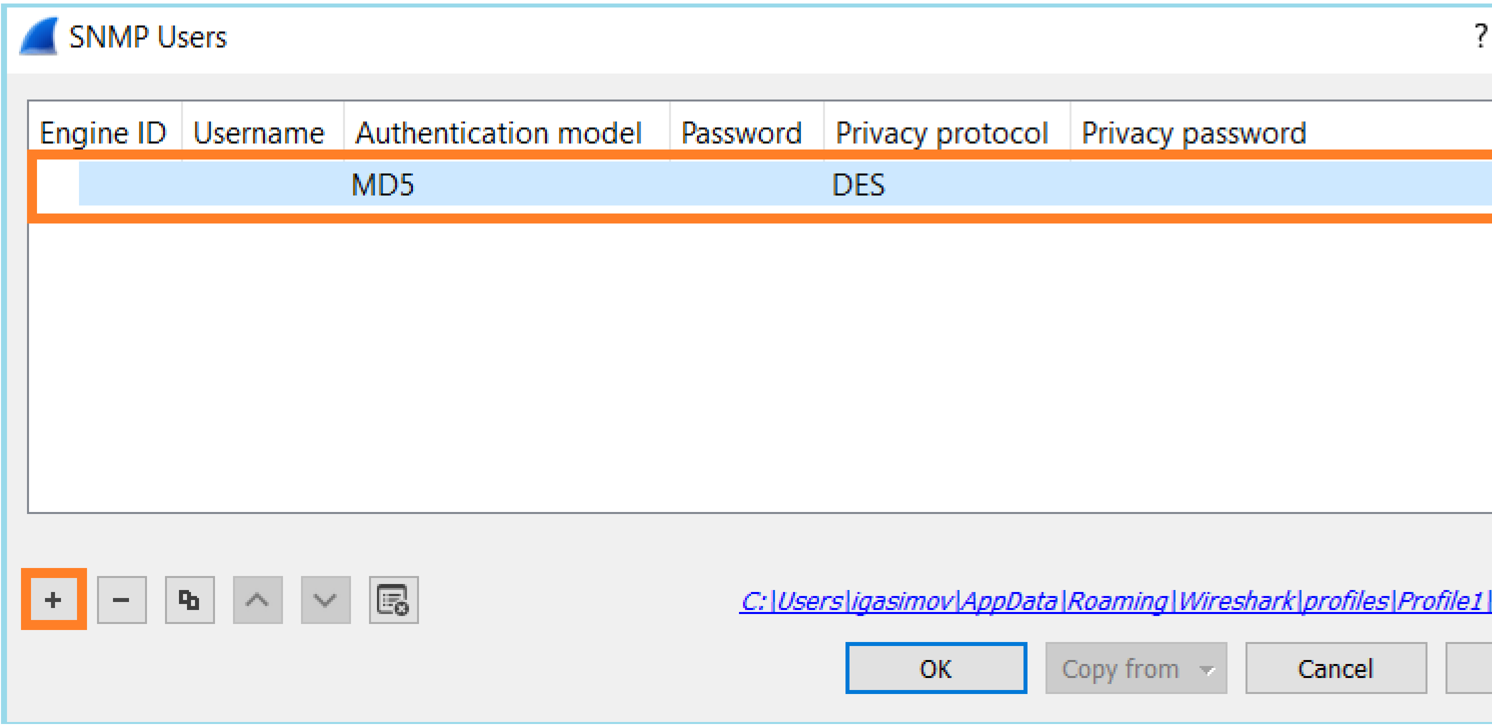


Mark/Unmark Packet Ctrl+M
Ignore/Unignore Packet Ctrl+D
Set/Unset Time Reference Ctrl+T
Time Shift... Ctrl+Shift+T
Packet Comment... Ctrl+Alt+C
Edit Resolved Name
Apply as Filter
Prepare a Filter
Conversation Filter
Colorize Conversation
SCTP
Follow
Copy
Protocol Preferences
Decode As...
Show Packet in New Window
Open Simple Network Management Protocol preferences...
Show SNMP OID in info column
Reassemble SNMP-over-TCP messages spanning multiple TCP segments
Display dissected variables inside SNMP tree
Users Table...
Enterprise Specific Trap Types...
SNMP UDP port: 161...
SNMP TCP port: 161...
Disable SNMP...

No.	Time	Protocol	Source	Source Port	Destination Port	Destination	Length	Info
1	0.000	SNMP	192.168.10.10	65484	161	192.168.5.254	100	getBulkRequest
2	0.000	SNMP	192.168.5.254	161	65484	192.168.10.10	167	report 1.3.6.1.6.3.15.1.1.4.0
3	0.176	SNMP	192.168.10.10	65484				encryptedPDU: privKey Unknown
4	0.176	SNMP	192.168.5.254	161				report 1.3.6.1.6.3.15.1.1.2.0
5	0.325	SNMP	192.168.10.10	65484				encryptedPDU: privKey Unknown
6	0.326	SNMP	192.168.5.254	161				encryptedPDU: privKey Unknown
7	0.490	SNMP	192.168.10.10	65484				encryptedPDU: privKey Unknown
8	0.490	SNMP	192.168.5.254	161				encryptedPDU: privKey Unknown
9	0.675	SNMP	192.168.10.10	65484				encryptedPDU: privKey Unknown
10	0.767	SNMP	192.168.5.254	161				encryptedPDU: privKey Unknown
11	0.945	SNMP	192.168.10.10	65484				encryptedPDU: privKey Unknown
12	0.946	SNMP	192.168.5.254	161				encryptedPDU: privKey Unknown
13	1.133	SNMP	192.168.10.10	65484				encryptedPDU: privKey Unknown
14	1.134	SNMP	192.168.5.254	161				encryptedPDU: privKey Unknown
15	1.317	SNMP	192.168.10.10	65484				encryptedPDU: privKey Unknown
16	1.318	SNMP	192.168.5.254	161				encryptedPDU: privKey Unknown
17	17.595	SNMP	192.168.10.10	62008				getBulkRequest
18	17.595	SNMP	192.168.5.254	161				report 1.3.6.1.6.3.15.1.1.4.0
19	17.749	SNMP	192.168.10.10	62008				
20	17.749	SNMP	192.168.5.254	161				
21	17.898	SNMP	192.168.10.10	62008				
22	17.899	SNMP	192.168.5.254	161	62008	192.168.10.10	678	
23	18.094	SNMP	192.168.10.10	62008	161	192.168.5.254	205	
24	18.094	SNMP	192.168.5.254	161	62008	192.168.10.10	560	
25	18.290	SNMP	192.168.10.10	62008	161	192.168.5.254	205	

```
    <[Destination Host: 192.168.5.254]>
    <[Source or Destination Host: 192.168.5.254]>
> User Datagram Protocol, Src Port: 65484, Dst Port: 161
v Simple Network Management Protocol
    msgVersion: snmpv3 (3)
  > msgGlobalData
```

In the SNMP Users table the SNMP version 3 Username, Authentication model, Authentication Password, Privacy protocol and the Privacy password were specified (actual credentials are not shown below):

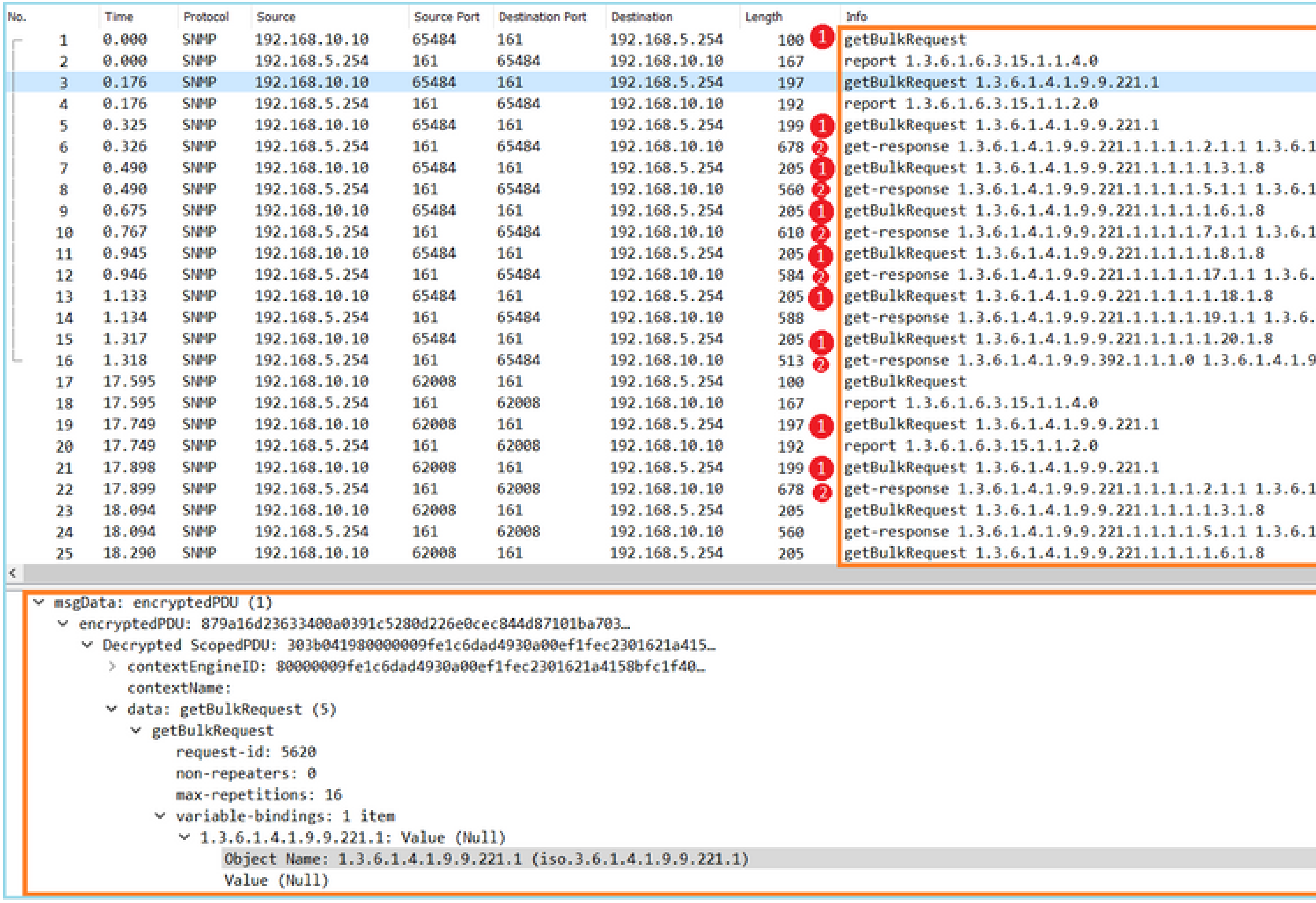


SNMP Users ?

Engine ID	Username	Authentication model	Password	Privacy protocol	Privacy password
		MD5		DES	

C:\Users\igasimov\AppData\Roaming\Wireshark\profiles\Profile1\

Once SNMP Users settings were applied Wireshark showed decrypted SNMP PDUs:



```
No.   Time    Protocol Source        Source Port Destination Port Destination   Length   Info
  1   0.000   SNMP     192.168.10.10 65484       161              192.168.5.254   100 1  getBulkRequest
  2   0.000   SNMP     192.168.5.254 161         65484            192.168.10.10   167    report 1.3.6.1.6.3.15.1.1.4.0
  3   0.176   SNMP     192.168.10.10 65484       161              192.168.5.254   197    getBulkRequest 1.3.6.1.4.1.9.9.221.1
  4   0.176   SNMP     192.168.5.254 161         65484            192.168.10.10   192    report 1.3.6.1.6.3.15.1.1.2.0
  5   0.325   SNMP     192.168.10.10 65484       161              192.168.5.254   199 1  getBulkRequest 1.3.6.1.4.1.9.9.221.1
  6   0.326   SNMP     192.168.5.254 161         65484            192.168.10.10   678 2  get-response 1.3.6.1.4.1.9.9.221.1.1.1.1.2.1.1 1.3.6.1
  7   0.490   SNMP     192.168.10.10 65484       161              192.168.5.254   205 1  getBulkRequest 1.3.6.1.4.1.9.9.221.1.1.1.1.3.1.8
  8   0.490   SNMP     192.168.5.254 161         65484            192.168.10.10   560 2  get-response 1.3.6.1.4.1.9.9.221.1.1.1.1.5.1.1 1.3.6.1
  9   0.675   SNMP     192.168.10.10 65484       161              192.168.5.254   205 1  getBulkRequest 1.3.6.1.4.1.9.9.221.1.1.1.1.6.1.8
 10   0.767   SNMP     192.168.5.254 161         65484            192.168.10.10   610 2  get-response 1.3.6.1.4.1.9.9.221.1.1.1.1.7.1.1 1.3.6.1
 11   0.945   SNMP     192.168.10.10 65484       161              192.168.5.254   205 1  getBulkRequest 1.3.6.1.4.1.9.9.221.1.1.1.1.8.1.8
 12   0.946   SNMP     192.168.5.254 161         65484            192.168.10.10   584 2  get-response 1.3.6.1.4.1.9.9.221.1.1.1.1.17.1.1 1.3.6.
 13   1.133   SNMP     192.168.10.10 65484       161              192.168.5.254   205 1  getBulkRequest 1.3.6.1.4.1.9.9.221.1.1.1.1.18.1.8
 14   1.134   SNMP     192.168.5.254 161         65484            192.168.10.10   588    get-response 1.3.6.1.4.1.9.9.221.1.1.1.1.19.1.1 1.3.6.
 15   1.317   SNMP     192.168.10.10 65484       161              192.168.5.254   205 1  getBulkRequest 1.3.6.1.4.1.9.9.221.1.1.1.1.20.1.8
 16   1.318   SNMP     192.168.5.254 161         65484            192.168.10.10   513 2  get-response 1.3.6.1.4.1.9.9.392.1.1.1.0 1.3.6.1.4.1.9
 17  17.595   SNMP     192.168.10.10 62008       161              192.168.5.254   100    getBulkRequest
 18  17.595   SNMP     192.168.5.254 161         62008            192.168.10.10   167    report 1.3.6.1.6.3.15.1.1.4.0
 19  17.749   SNMP     192.168.10.10 62008       161              192.168.5.254   197 1  getBulkRequest 1.3.6.1.4.1.9.9.221.1
 20  17.749   SNMP     192.168.5.254 161         62008            192.168.10.10   192    report 1.3.6.1.6.3.15.1.1.2.0
 21  17.898   SNMP     192.168.10.10 62008       161              192.168.5.254   199 1  getBulkRequest 1.3.6.1.4.1.9.9.221.1
 22  17.899   SNMP     192.168.5.254 161         62008            192.168.10.10   678 2  get-response 1.3.6.1.4.1.9.9.221.1.1.1.1.2.1.1 1.3.6.1
 23  18.094   SNMP     192.168.10.10 62008       161              192.168.5.254   205    getBulkRequest 1.3.6.1.4.1.9.9.221.1.1.1.1.3.1.8
 24  18.094   SNMP     192.168.5.254 161         62008            192.168.10.10   560    get-response 1.3.6.1.4.1.9.9.221.1.1.1.1.5.1.1 1.3.6.1
 25  18.290   SNMP     192.168.10.10 62008       161              192.168.5.254   205    getBulkRequest 1.3.6.1.4.1.9.9.221.1.1.1.1.6.1.8
```

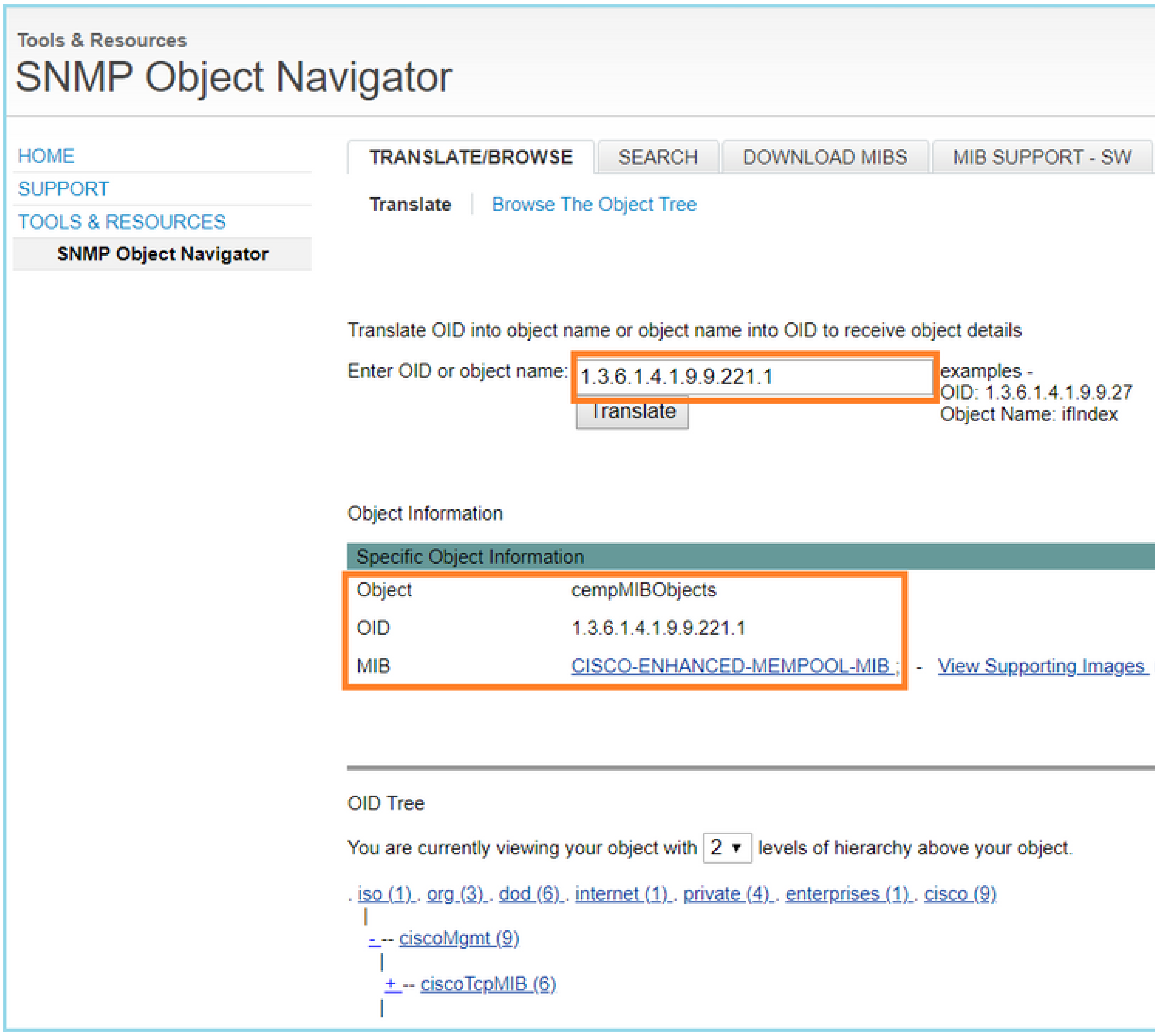
```
v msgData: encryptedPDU (1)
  v encryptedPDU: 879a16d23633400a0391c5280d226e0cec844d87101ba703…
    v Decrypted ScopedPDU: 303b041980000009fe1c6dad4930a00ef1fec2301621a415…
      > contextEngineID: 80000009fe1c6dad4930a00ef1fec2301621a4158bfc1f40…
        contextName:
      v data: getBulkRequest (5)
        v getBulkRequest
            request-id: 5620
            non-repeaters: 0
            max-repetitions: 16
          v variable-bindings: 1 item
            v 1.3.6.1.4.1.9.9.221.1: Value (Null)
                Object Name: 1.3.6.1.4.1.9.9.221.1 (iso.3.6.1.4.1.9.9.221.1)
                Value (Null)
```

Key points:

1. The SNMP monitoring tools used SNMP getBulkRequest to query and walk over the parent OID 1.3.6.1.4.1.9.9.221.1 and related OIDs.
2. The FTD responded to each getBulkRequest with get-response that contain OIDs related to 1.3.6.1.4.1.9.9.221.1.

Action 2. Identify the SNMP OIDs.

SNMP Object Navigator showed that OID 1.3.6.1.4.1.9.9.221.1 belongs to the management information base (MIB) named **CISCO-ENHANCED-MEMPOOL-MIB**, as shown in the image:



Translate OID into object name or object name into OID to receive object details

Enter OID or object name: 1.3.6.1.4.1.9.9.221.1

examples -
OID: 1.3.6.1.4.1.9.9.27
Object Name: ifIndex

Object Information

Specific Object Information

Object	cempMIBObjects
OID	1.3.6.1.4.1.9.9.221.1
MIB	CISCO-ENHANCED-MEMPOOL-MIB ; - View Supporting Images

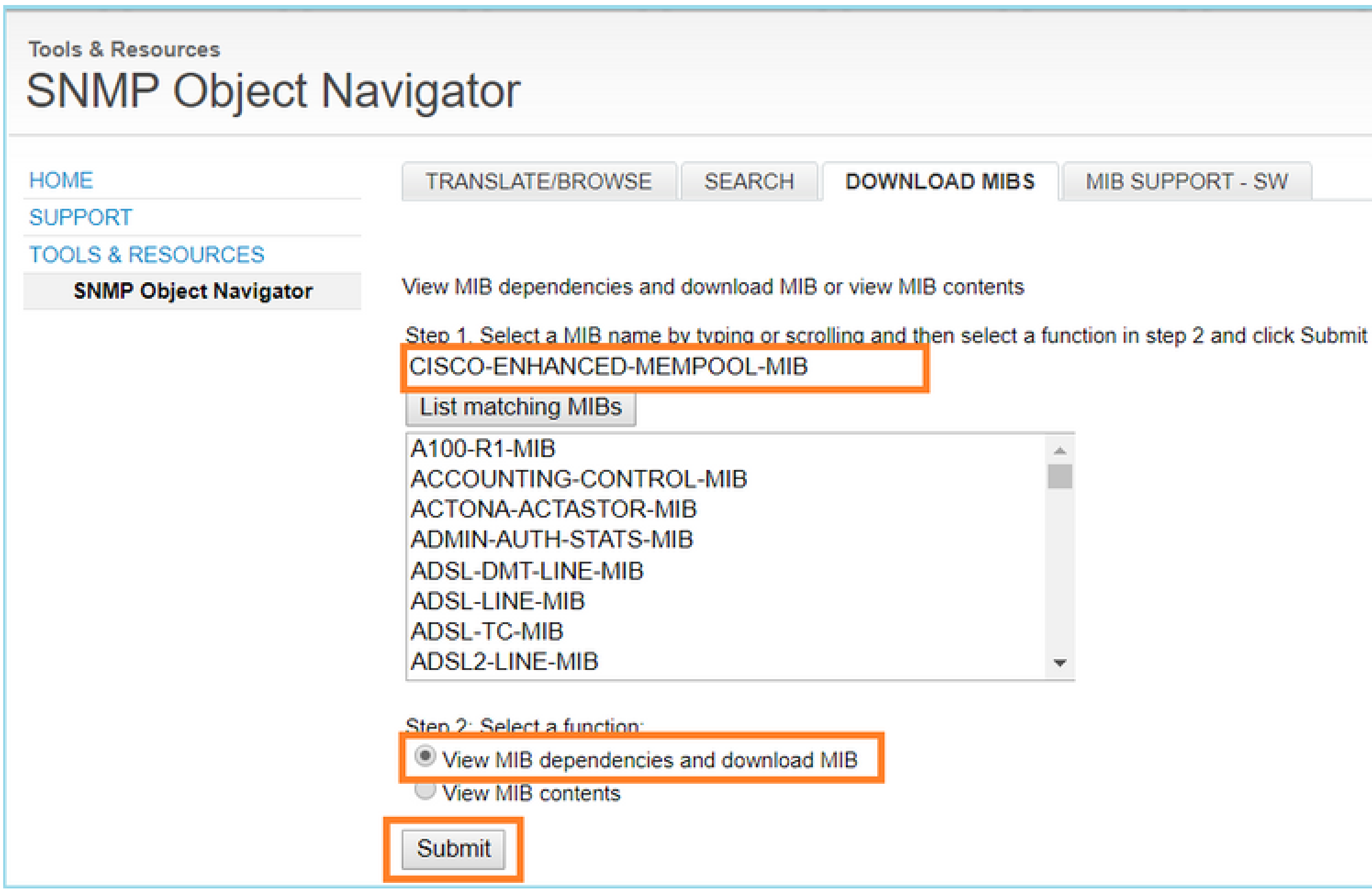
OID Tree

You are currently viewing your object with 2 levels of hierarchy above your object.

```
. iso (1) . org (3) . dod (6) . internet (1) . private (4) . enterprises (1) . cisco (9)
  |
  - -- ciscoMgmt (9)
     |
     + -- ciscoTcpMIB (6)
     |
```

To display the OIDs in human-readable format in Wireshark:

1. Download the MIB **CISCO-ENHANCED-MEMPOOL-MIB** and its dependencies, as shown in the image:


Tools & Resources

SNMP Object Navigator

HOME
SUPPORT
TOOLS & RESOURCES
SNMP Object Navigator

TRANSLATE/BROWSE | SEARCH | DOWNLOAD MIBS | MIB SUPPORT - SW

View MIB dependencies and download MIB or view MIB contents

Step 1. Select a MIB name by typing or scrolling and then select a function in step 2 and click Submit

CISCO-ENHANCED-MEMPOOL-MIB

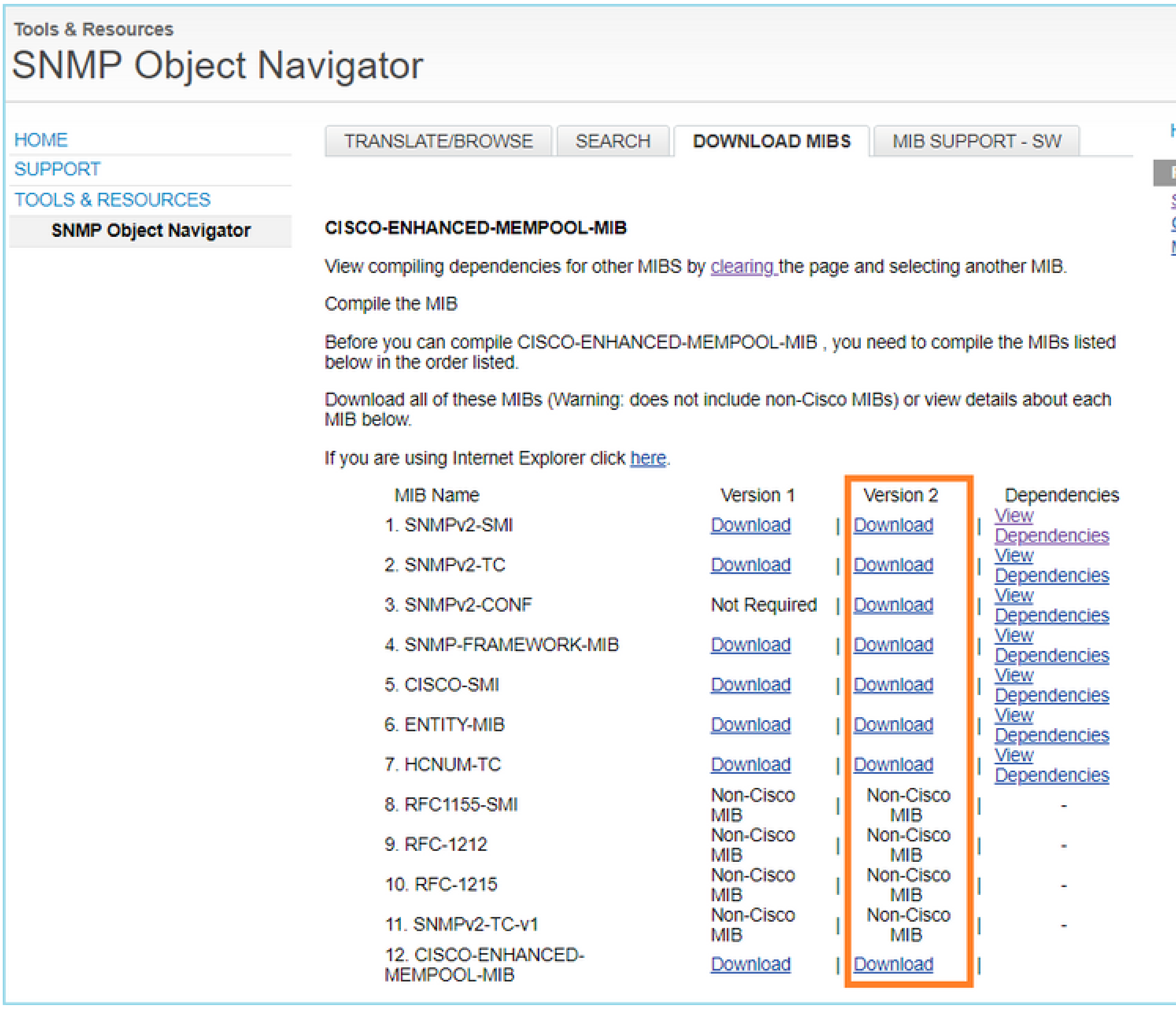
List matching MIBs

A100-R1-MIB
ACCOUNTING-CONTROL-MIB
ACTONA-ACTASTOR-MIB
ADMIN-AUTH-STATS-MIB
ADSL-DMT-LINE-MIB
ADSL-LINE-MIB
ADSL-TC-MIB
ADSL2-LINE-MIB

Step 2: Select a function:

[✓ View MIB dependencies and download MIB]
View MIB contents

Submit


Tools & Resources

SNMP Object Navigator

HOME
SUPPORT
TOOLS & RESOURCES
SNMP Object Navigator

TRANSLATE/BROWSE | SEARCH | DOWNLOAD MIBS | MIB SUPPORT - SW

CISCO-ENHANCED-MEMPOOL-MIB

View compiling dependencies for other MIBS by clearing the page and selecting another MIB.

Compile the MIB

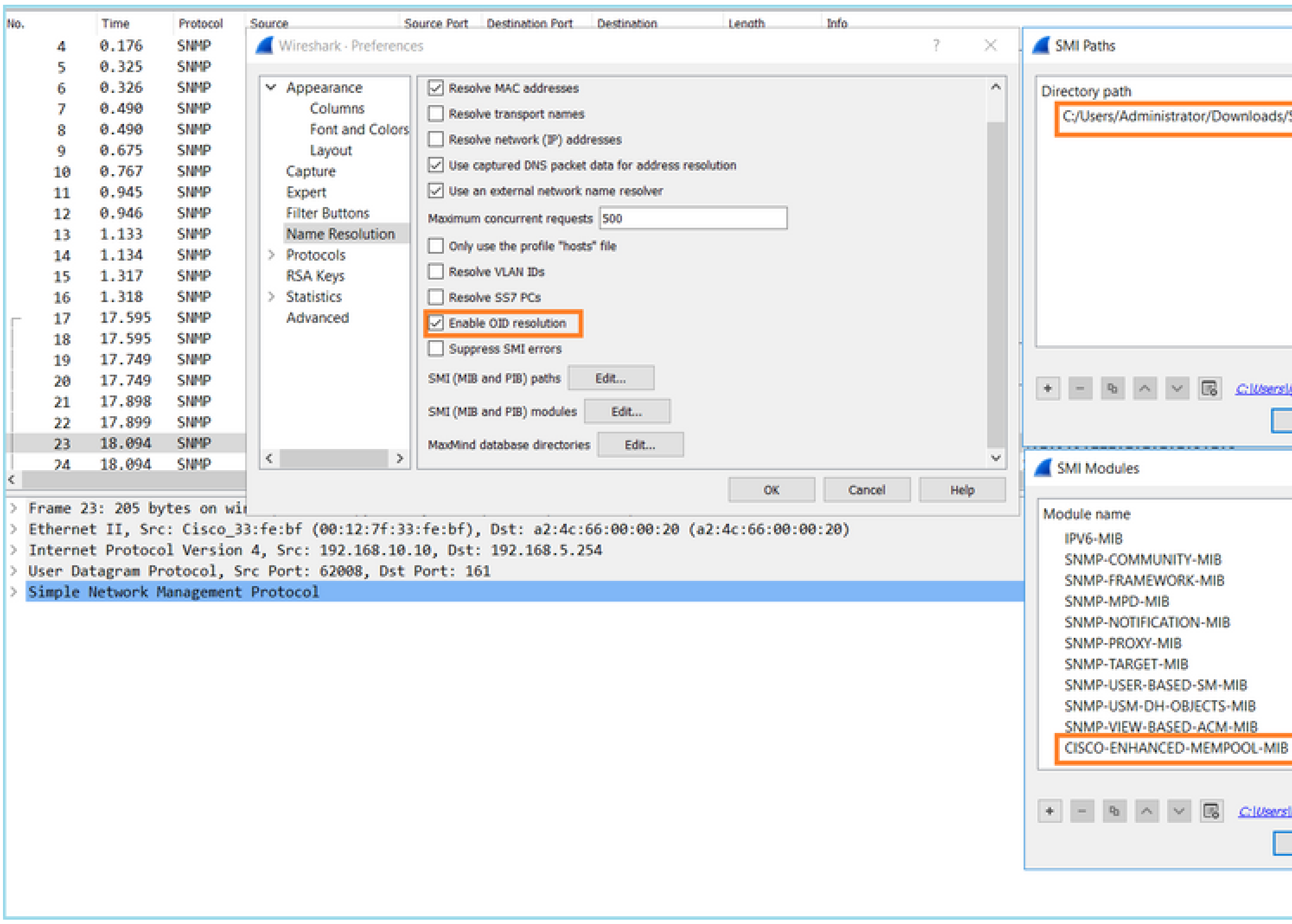
Before you can compile CISCO-ENHANCED-MEMPOOL-MIB , you need to compile the MIBs listed below in the order listed.

Download all of these MIBs (Warning: does not include non-Cisco MIBs) or view details about each MIB below.

If you are using Internet Explorer click here.

MIB Name	Version 1	Version 2	Dependencies
1. SNMPv2-SMI	Download	Download	View Dependencies
2. SNMPv2-TC	Download	Download	View Dependencies
3. SNMPv2-CONF	Not Required	Download	View Dependencies
4. SNMP-FRAMEWORK-MIB	Download	Download	View Dependencies
5. CISCO-SMI	Download	Download	View Dependencies
6. ENTITY-MIB	Download	Download	View Dependencies
7. HCNUM-TC	Download	Download	View Dependencies
8. RFC1155-SMI	Non-Cisco MIB	Non-Cisco MIB	-
9. RFC-1212	Non-Cisco MIB	Non-Cisco MIB	-
10. RFC-1215	Non-Cisco MIB	Non-Cisco MIB	-
11. SNMPv2-TC-v1	Non-Cisco MIB	Non-Cisco MIB	-
12. CISCO-ENHANCED-MEMPOOL-MIB	Download	Download	

2. In Wireshark in **Edit > Preferences > Name Resolution** window the **Enable OID Resolution** is checked. In **SMI (MIB and PIB paths)** window specify the folder with the downloaded MIBs and in **SMI (MIB and PIB modules).** The CISCO-ENHANCED-MEMPOOL-MIB is added automatically to the list of modules:



```
No.  Time    Protocol  Source  Source Port  Destination Port  Destination  Length  Info
 4   0.176   SNMP
 5   0.325   SNMP
 6   0.326   SNMP
 7   0.490   SNMP
 8   0.490   SNMP
 9   0.675   SNMP
10   0.767   SNMP
11   0.945   SNMP
12   0.946   SNMP
13   1.133   SNMP
14   1.134   SNMP
15   1.317   SNMP
16   1.318   SNMP
17   17.595  SNMP
18   17.595  SNMP
19   17.749  SNMP
20   17.749  SNMP
21   17.898  SNMP
22   17.899  SNMP
23   18.094  SNMP
24   18.094  SNMP
```

```
> Frame 23: 205 bytes on wi
> Ethernet II, Src: Cisco_33:fe:bf (00:12:7f:33:fe:bf), Dst: a2:4c:66:00:00:20 (a2:4c:66:00:00:20)
> Internet Protocol Version 4, Src: 192.168.10.10, Dst: 192.168.5.254
> User Datagram Protocol, Src Port: 62008, Dst Port: 161
> Simple Network Management Protocol
```

Wireshark · Preferences

- [✓] Resolve MAC addresses
- [ ] Resolve transport names
- [ ] Resolve network (IP) addresses
- [✓] Use captured DNS packet data for address resolution
- [✓] Use an external network name resolver
- Maximum concurrent requests: 500
- [ ] Only use the profile "hosts" file
- [ ] Resolve VLAN IDs
- [ ] Resolve SS7 PCs
- [✓] Enable OID resolution
- [ ] Suppress SMI errors
- SMI (MIB and PIB) paths: Edit...
- SMI (MIB and PIB) modules: Edit...
- MaxMind database directories: Edit...

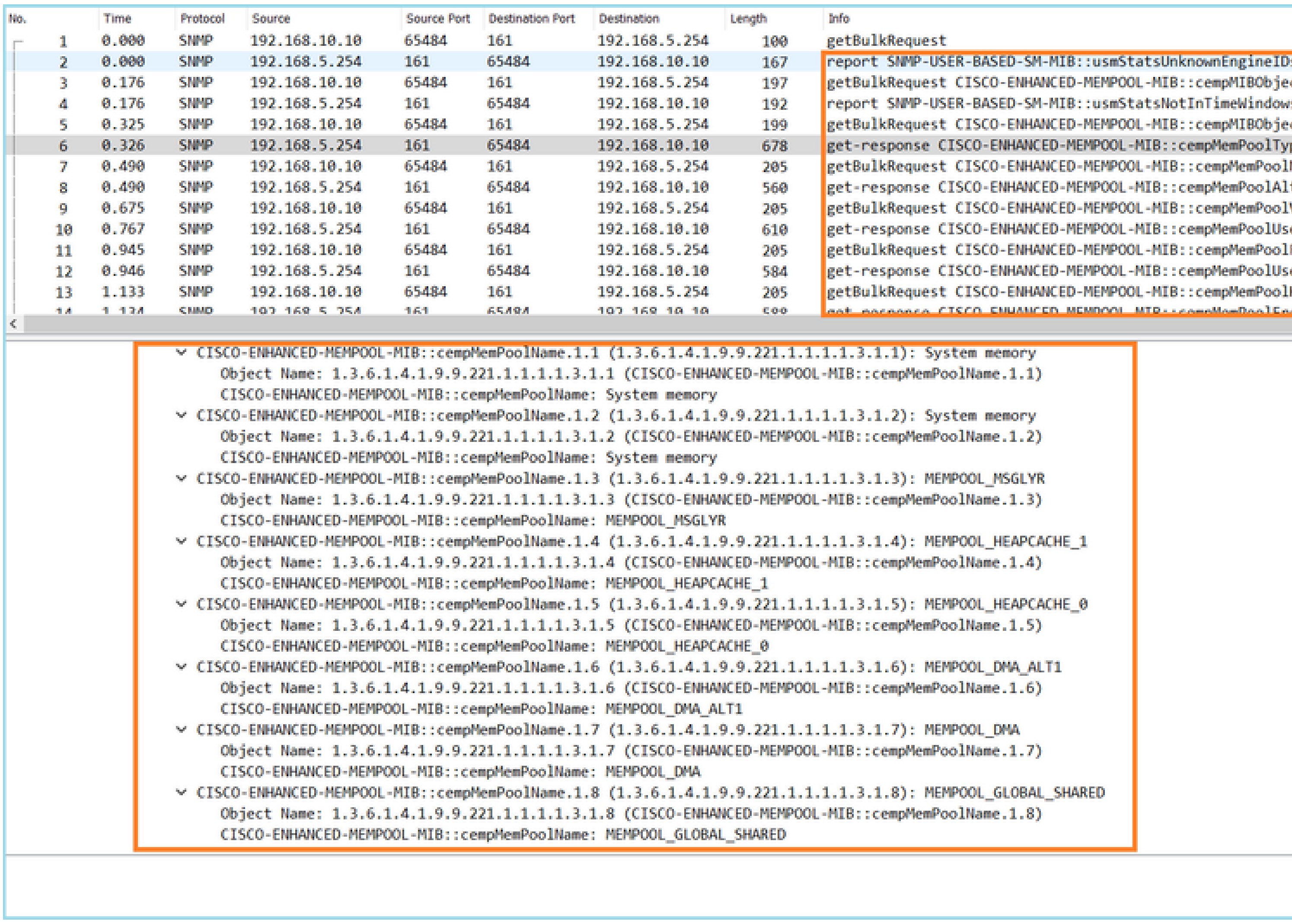
SMI Paths

Directory path
C:/Users/Administrator/Downloads/

SMI Modules

Module name
IPV6-MIB
SNMP-COMMUNITY-MIB
SNMP-FRAMEWORK-MIB
SNMP-MPD-MIB
SNMP-NOTIFICATION-MIB
SNMP-PROXY-MIB
SNMP-TARGET-MIB
SNMP-USER-BASED-SM-MIB
SNMP-USM-DH-OBJECTS-MIB
SNMP-VIEW-BASED-ACM-MIB
CISCO-ENHANCED-MEMPOOL-MIB

3. Once Wireshark is restarted, OID resolution is activated:



```
No.   Time   Protocol Source         Source Port Destination Port Destination    Length Info
  1   0.000  SNMP     192.168.10.10  65484       161              192.168.5.254   100   getBulkRequest
  2   0.000  SNMP     192.168.5.254  161         65484            192.168.10.10   167   report SNMP-USER-BASED-SM-MIB::usmStatsUnknownEngineIDs
  3   0.176  SNMP     192.168.10.10  65484       161              192.168.5.254   197   getBulkRequest CISCO-ENHANCED-MEMPOOL-MIB::cempMIBObje
  4   0.176  SNMP     192.168.5.254  161         65484            192.168.10.10   192   report SNMP-USER-BASED-SM-MIB::usmStatsNotInTimeWindow
  5   0.325  SNMP     192.168.10.10  65484       161              192.168.5.254   199   getBulkRequest CISCO-ENHANCED-MEMPOOL-MIB::cempMIBObje
  6   0.326  SNMP     192.168.5.254  161         65484            192.168.10.10   678   get-response CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolTyp
  7   0.490  SNMP     192.168.10.10  65484       161              192.168.5.254   205   getBulkRequest CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolN
  8   0.490  SNMP     192.168.5.254  161         65484            192.168.10.10   560   get-response CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolAl
  9   0.675  SNMP     192.168.10.10  65484       161              192.168.5.254   205   getBulkRequest CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolV
 10   0.767  SNMP     192.168.5.254  161         65484            192.168.10.10   610   get-response CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolUs
 11   0.945  SNMP     192.168.10.10  65484       161              192.168.5.254   205   getBulkRequest CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolI
 12   0.946  SNMP     192.168.5.254  161         65484            192.168.10.10   584   get-response CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolUs
 13   1.133  SNMP     192.168.10.10  65484       161              192.168.5.254   205   getBulkRequest CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolI
 14   1.134  SNMP     192.168.5.254  161         65484            192.168.10.10   588   get-response CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolFr
```

```
v CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.1 (1.3.6.1.4.1.9.9.221.1.1.1.1.3.1.1): System memory
    Object Name: 1.3.6.1.4.1.9.9.221.1.1.1.1.3.1.1 (CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.1)
    CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName: System memory
v CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.2 (1.3.6.1.4.1.9.9.221.1.1.1.1.3.1.2): System memory
    Object Name: 1.3.6.1.4.1.9.9.221.1.1.1.1.3.1.2 (CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.2)
    CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName: System memory
v CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.3 (1.3.6.1.4.1.9.9.221.1.1.1.1.3.1.3): MEMPOOL_MSGLYR
    Object Name: 1.3.6.1.4.1.9.9.221.1.1.1.1.3.1.3 (CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.3)
    CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName: MEMPOOL_MSGLYR
v CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.4 (1.3.6.1.4.1.9.9.221.1.1.1.1.3.1.4): MEMPOOL_HEAPCACHE_1
    Object Name: 1.3.6.1.4.1.9.9.221.1.1.1.1.3.1.4 (CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.4)
    CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName: MEMPOOL_HEAPCACHE_1
v CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.5 (1.3.6.1.4.1.9.9.221.1.1.1.1.3.1.5): MEMPOOL_HEAPCACHE_0
    Object Name: 1.3.6.1.4.1.9.9.221.1.1.1.1.3.1.5 (CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.5)
    CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName: MEMPOOL_HEAPCACHE_0
v CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.6 (1.3.6.1.4.1.9.9.221.1.1.1.1.3.1.6): MEMPOOL_DMA_ALT1
    Object Name: 1.3.6.1.4.1.9.9.221.1.1.1.1.3.1.6 (CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.6)
    CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName: MEMPOOL_DMA_ALT1
v CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.7 (1.3.6.1.4.1.9.9.221.1.1.1.1.3.1.7): MEMPOOL_DMA
    Object Name: 1.3.6.1.4.1.9.9.221.1.1.1.1.3.1.7 (CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.7)
    CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName: MEMPOOL_DMA
v CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.8 (1.3.6.1.4.1.9.9.221.1.1.1.1.3.1.8): MEMPOOL_GLOBAL_SHARED
    Object Name: 1.3.6.1.4.1.9.9.221.1.1.1.1.3.1.8 (CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.8)
    CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName: MEMPOOL_GLOBAL_SHARED
```

Based on the decrypted output of the capture file the SNMP monitoring tool was periodically (10 seconds interval) polling data about the utilization of memory pools on the FTD. As explained in the TechNote article [ASA SNMP Polling for Memory-Related Statistics](), polling the Global Shared Pool (GSP) utilization with SNMP results in high CPU usage. In this case from the captures, it was clear that the Global Shared Pool utilization was periodically polled as part of SNMP getBulkRequest primitive.

In order to minimize the CPU hogs caused by the SNMP process, it was recommended to follow the mitigation steps for the CPU Hogs for SNMP mentioned in the article and avoid to poll the OIDs related to GSP. Without the SNMP poll for the OIDs that relate to GSP no CPU hogs caused by the SNMP process were observed and the rate of overruns significantly decreased.

Related Information

- Cisco Firepower Management Center Configuration Guides
- Clarify Firepower Threat Defense Access Control Policy Rule Actions
- Work with Firepower Threat Defense Captures and Packet Tracer
- Learn Wireshark