

# 分析Firepower防火墙捕获以排除网络故障

## 目录

---

### [简介](#)

#### [先决条件](#)

[要求](#)

[使用的组件](#)

#### [背景信息](#)

#### [如何收集和导出NGFW产品系列的捕获信息？](#)

[收集FXOS捕获](#)

[启用并收集FTD Lina捕获](#)

[启用和收集FTD Snort捕获](#)

#### [故障排除](#)

##### [例 1.出口接口没有TCP SYN](#)

[捕获分析](#)

[推荐的操作](#)

[可能的原因和建议的操作摘要](#)

##### [案例 2.来自客户端的TCP SYN、来自服务器的TCP RST](#)

[捕获分析](#)

[推荐的操作](#)

##### [案例 3.来自一个终端的TCP三次握手+ RST](#)

[捕获分析](#)

[3.1-来自客户端的TCP三次握手+延迟RST](#)

[推荐的操作](#)

[3.2 - TCP三次握手+来自客户端的延迟FIN/ACK +来自服务器的延迟RST](#)

[推荐的操作](#)

[3.3 -来自客户端的TCP三次握手+延迟RST](#)

[推荐的操作](#)

[3.4 -来自服务器的TCP三次握手+即时RST](#)

[推荐的操作](#)

##### [案例 4.来自客户端的TCP RST](#)

[捕获分析](#)

[推荐的操作](#)

##### [案例 5.缓慢TCP传输 \(场景1\)](#)

[场景 1.缓慢传输](#)

[捕获分析](#)

[推荐的操作](#)

[场景 2：快速传输](#)

##### [案例 6.缓慢TCP传输 \(场景2\)](#)

[捕获分析](#)

[推荐的操作](#)

##### [导出捕获检查入口数据包与出口数据包之间的时间差案例7.TCP连接问题 \(数据包损坏\)](#)

[捕获分析](#)

[推荐的操作](#)

##### [案例 8.UDP连接问题 \(缺少数据包\)](#)

[捕获分析](#)

---

[推荐的操作](#)

[捕获分析](#)

[推荐的操作](#)

[案例 10.HTTPS连接问题 \( 场景2 \)](#)

[捕获分析](#)

[推荐的操作](#)

[捕获分析](#)

[推荐的操作](#)

[案例 12.间歇性连接问题 \( ARP毒化 \)](#)

[捕获分析](#)

[推荐的操作](#)

[案例 13.标识导致CPU占用的SNMP对象标识符\(OID\)](#)

[捕获分析](#)

[推荐的操作](#)

[相关信息](#)

---

## 简介

本文档介绍旨在有效排查网络问题的各种数据包捕获分析技术。

## 先决条件

### 要求

Cisco 建议您了解以下主题：

- Firepower平台架构
- NGFW日志
- NGFW packet-tracer

此外，在开始分析数据包捕获之前，强烈建议满足以下要求：

- 了解协议操作- 如果您不了解捕获的协议如何运行，请勿开始检查数据包捕获。
- 了解拓扑 -您必须了解端到端的传输设备。如果无法做到这一点，您必须至少了解上游和下游设备。
- 了解设备- 您必须了解设备如何处理数据包、涉及的接口（入口/出口）、设备架构是什么，以及各个捕获点是什么。
- 了解配置- 您必须了解数据包流应该如何由设备在以下方面处理：
  - 路由/出口接口
  - 应用的策略
  - 网络地址转换 (NAT)
- 了解可用工具- 除了捕获之外，建议准备好应用其他工具和技术（例如日志记录和跟踪程序），如果需要，请将其与捕获的数据包相关联。

### 使用的组件

本文档中的信息基于以下软件和硬件版本：

- 大多数场景基于运行FTD软件6.5.x的FP4140。
- FMC运行软件6.5.x。

本文档中的信息都是基于特定实验室环境中的设备编写的。本文档中使用的所有设备最初均采用原始（默认）配置。如果您的网络处于活动状态，请确保您了解所有命令的潜在影响。

## 背景信息

数据包捕获是当今最容易被忽视的故障排除工具之一。每天，Cisco TAC都可以通过分析捕获的数据来解决许多问题。

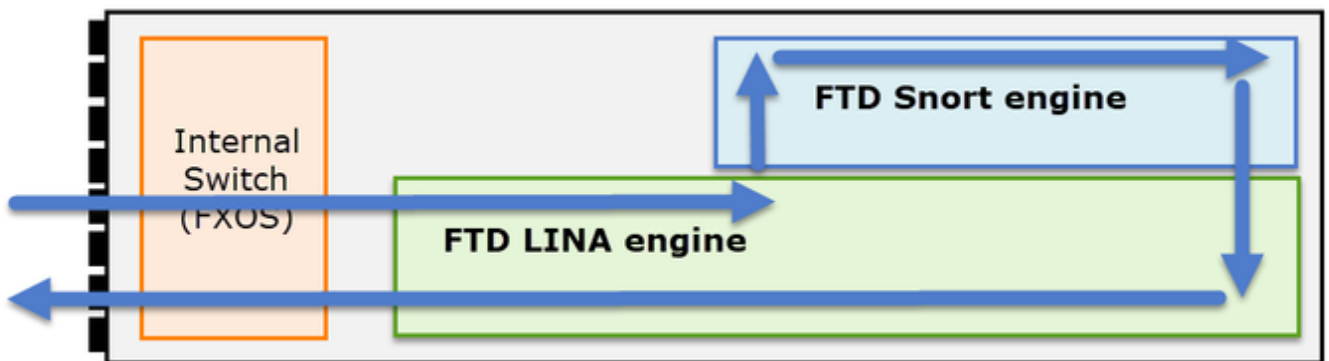
本文档的目标是帮助网络和安全工程师主要根据数据包捕获分析来识别和排除常见网络问题。

本文档中介绍的所有场景均基于思科技术支持中心(TAC)中看到的实际用户案例。

本文档从思科下一代防火墙(NGFW)的角度介绍了数据包捕获，但相同的概念也适用于其他设备类型。

## 如何收集和导出NGFW产品系列的捕获信息？

对于Firepower设备(1xxx、21xx、41xx、93xx)和Firepower威胁防御(FTD)应用，数据包处理可视化，如图所示。



1. 数据包进入入口接口，由机箱内部交换机处理。
2. 数据包进入FTD Lina引擎，该引擎主要执行L3/L4检查。
3. 如果策略要求数据包由Snort引擎进行检查（主要是L7检查）。
4. Snort引擎返回数据包的判定。
5. LINA引擎根据Snort的判定丢弃或转发数据包。
6. 数据包通过内部机箱交换机离开机箱。

根据所示架构，FTD捕获可在三(3)个不同位置进行：

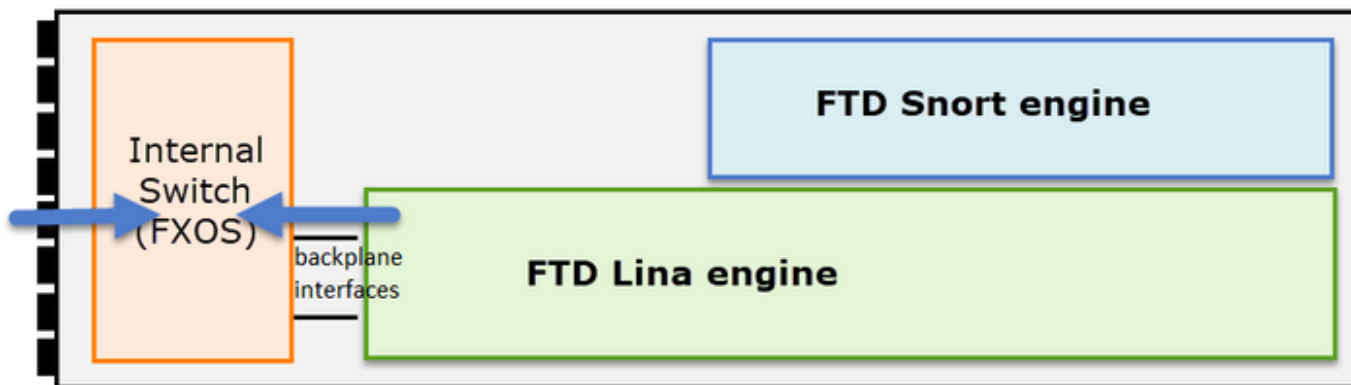
- FXOS
- FTD Lina引擎
- FTD Snort引擎

## 收集FXOS捕获

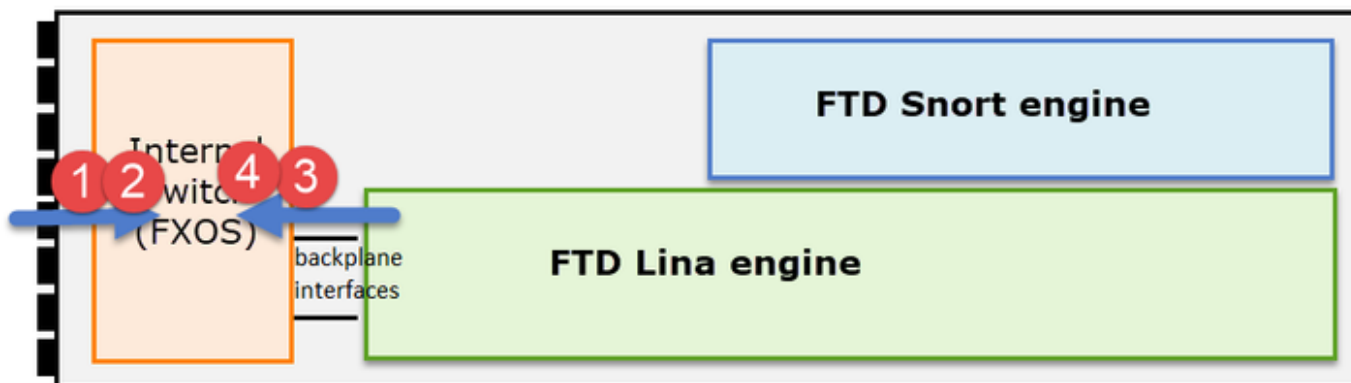
本文档介绍了该过程：

[https://www.cisco.com/c/en/us/td/docs/security/firepower/fxos/fxos271/web-guide/b\\_GUI\\_FXOS\\_ConfigGuide\\_271/troubleshooting.html#concept\\_E8823CC63C934A909BBC0DF12F](https://www.cisco.com/c/en/us/td/docs/security/firepower/fxos/fxos271/web-guide/b_GUI_FXOS_ConfigGuide_271/troubleshooting.html#concept_E8823CC63C934A909BBC0DF12F)


FXOS捕获只能从内部交换机的入口方向获取，如下图所示。



此处显示，由于内部交换机架构的原因，每个方向有两个捕获点。



在点2、3和4中捕获的数据包具有虚拟网络标记(VNTag)。

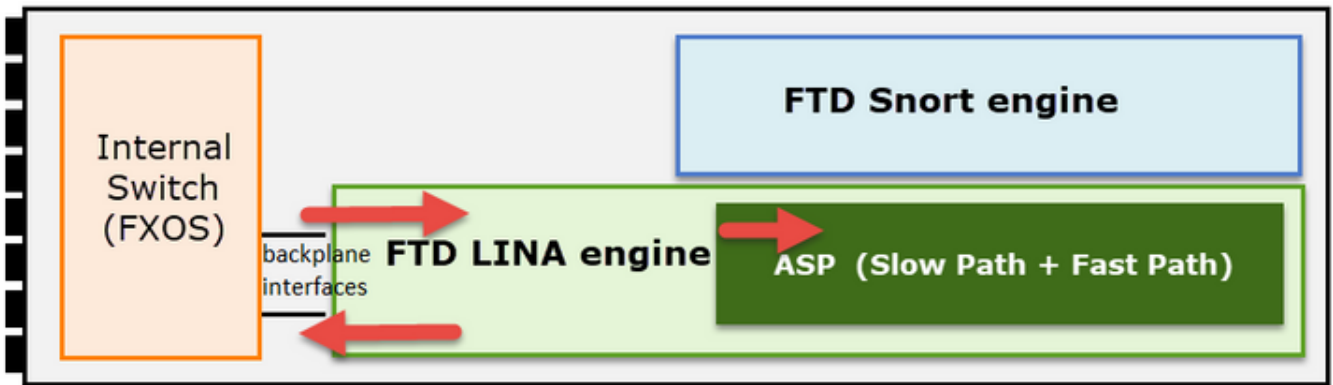
 **注意：**FXOS机箱级别捕获仅在FP41xx和FP93xx平台上可用。FP1xxx和FP21xx不提供此功能。

## 启用并收集FTD Lina捕获

主要捕获点：

- Ingress 接口
- Egress 接口
- 加速安全路径(ASP)





您可以使用Firepower管理中心用户界面(FMC UI)或FTD CLI启用和收集FTD Lina捕获。

在内部接口上从CLI启用捕获：

```
<#root>
```

```
firepower#
```

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

此捕获在两个方向上匹配IP 192.168.103.1和192.168.101.1之间的流量。

启用ASP捕获以查看FTD Lina引擎丢弃的所有数据包：

```
<#root>
```

```
firepower#
```

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

将FTD Lina捕获导出到FTP服务器：

```
<#root>
```

```
firepower#
```

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

将FTD Lina捕获导出到TFTP服务器：

```
<#root>
```

```
firepower#
```

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

从FMC 6.2.x版本开始，您可以从FMC UI启用并收集FTD Lina捕获。

从FMC管理的防火墙收集FTD捕获的另一种方法是。

## 第 1 步

对于LINA或ASP捕获，请将捕获复制到FTD磁盘。

```
<#root>
firepower#
copy /pcap capture:capin disk0:capin.pcap

Source capture name [capin]?

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

## 步骤 2

导航到专家模式，找到保存的捕获，然后将其复制到/ngfw/var/common位置：

```
<#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

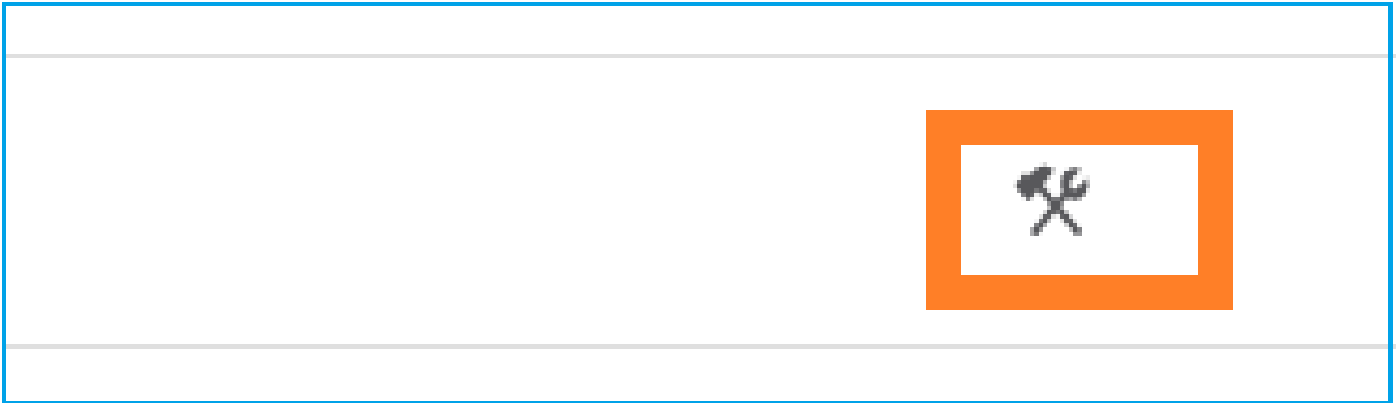
capin.pcap

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

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

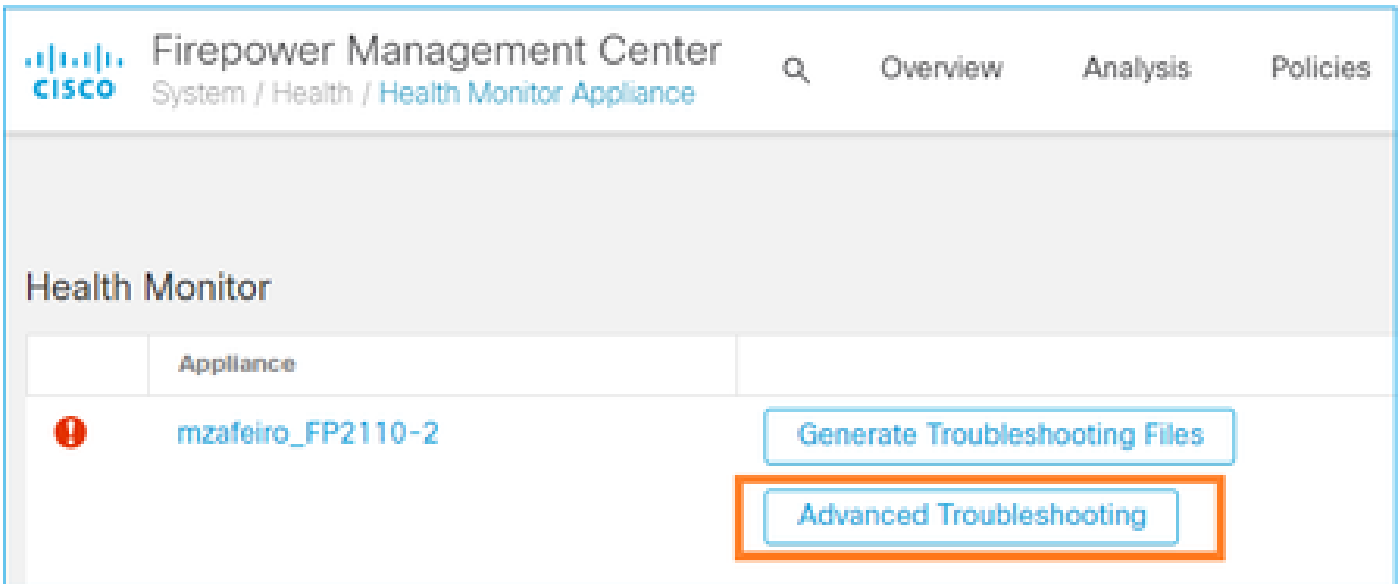
### 步骤 3

登录到管理FTD的FMC并导航到设备>设备管理。找到FTD设备并选择故障排除图标：

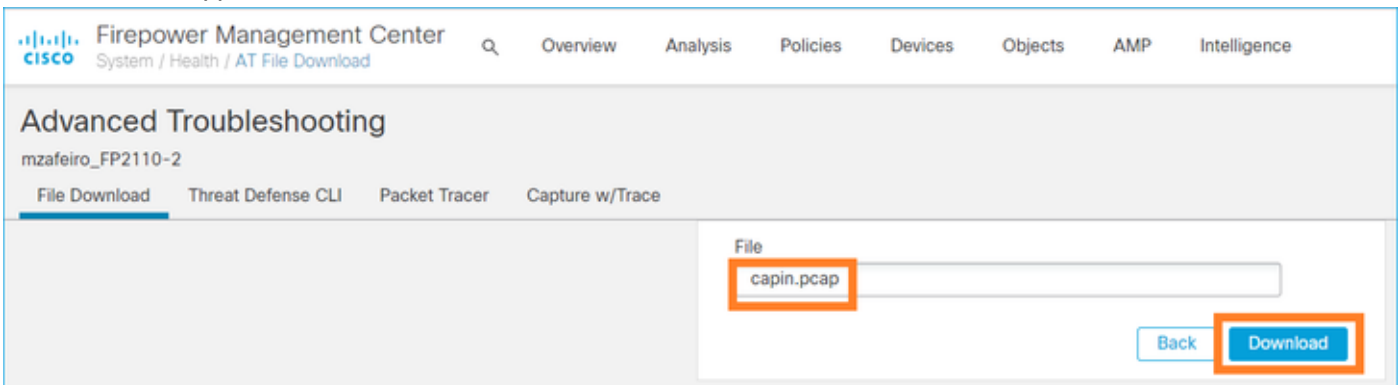


### 步骤 4

选择高级故障排除：



指定捕获文件名并选择Download：

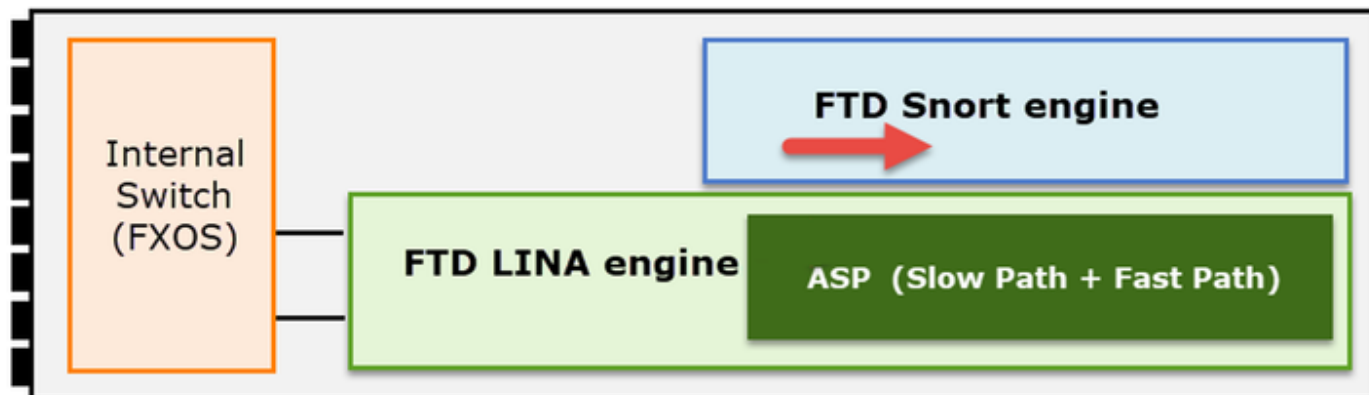


有关如何启用/收集FMC UI捕获的更多示例，请阅读本文档：

<https://www.cisco.com/c/en/us/support/docs/security/firepower-ngfw/212474-working-with->

## 启用和收集FTD Snort捕获

捕获点显示在此处的图像中。



启用Snort级捕获：

```
<#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
```

要将捕获写入名称为capture.pcap的文件并通过FTP将其复制到远程服务器，请执行以下操作：

```
<#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.

>

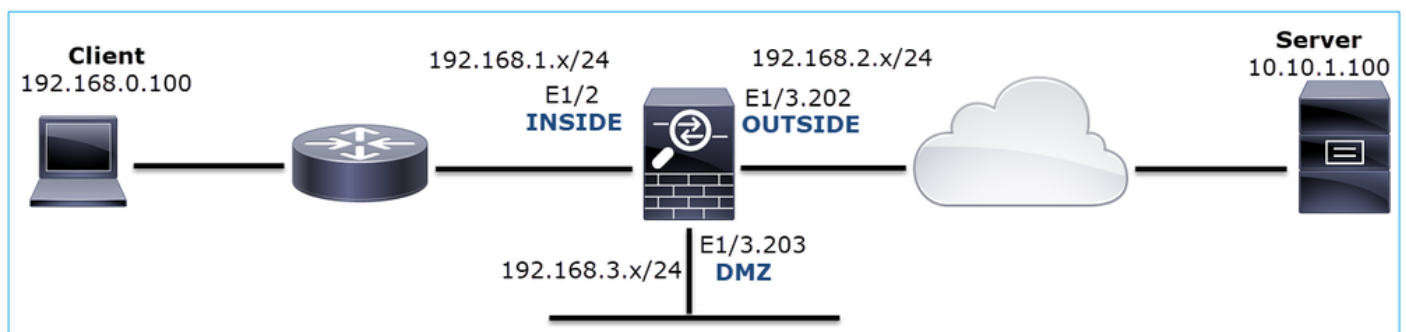
有关包含不同捕获过滤器的更多Snort级捕获示例，请查看以下文档：

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

## 故障排除

### 例 1. 出口接口没有TCP SYN

拓扑如图所示：



问题说明：HTTP不起作用

受影响的流：

源IP：192.168.0.100

DST IP : 10.10.1.100

协议 : TCP 80

### 捕获分析

在FTD LINA引擎上启用捕获：

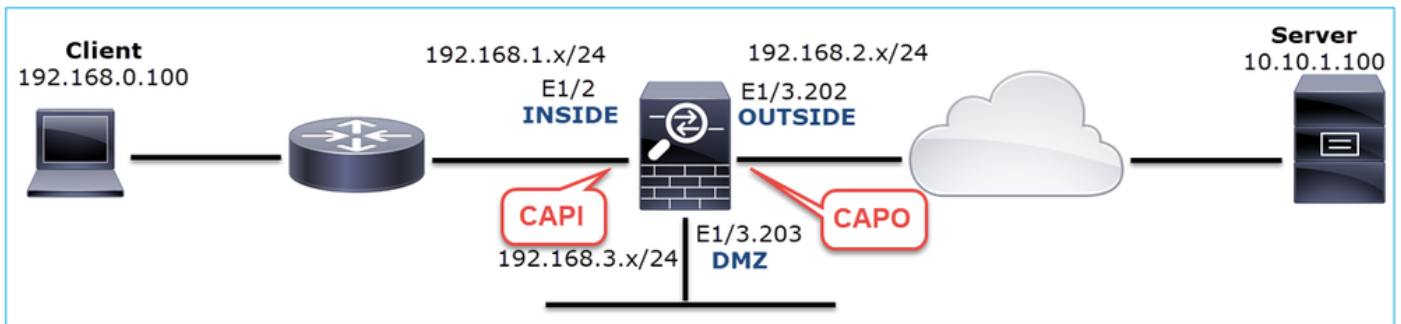
```
<#root>
```

```
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
```



捕获-功能场景：

作为基准，从功能场景获取捕获信息始终非常有用。

捕获在NGFW内部接口上获取，如图所示：

The screenshot shows a Wireshark capture of a TCP stream. The capture shows a SYN exchange, followed by a GET request and a 200 OK response. The source IP is 192.168.0.100 and the destination IP is 10.10.1.100. Red boxes and numbers highlight key details: 1. The SYN packet (No. 2), 2. The GET request (No. 5), 3. The 200 OK response (No. 6), and 4. The Ethernet II header showing the source MAC address Cisco\_fc:fc:d8 (4c:4e:35:fc:fc:d8).

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 SACK_PERM=1
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 MSS=1380 WS=256 SACK_PERM=1
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 [TCP segment of a reassembled PDU]
9	0.000000	10.10.1.100	192.168.0.100	TCP	1434	80 → 1779 [ACK] Seq=2293 Ack=593 Win=65792 Len=1380 [TCP segment of a reassembled PDU]
10	0.000626	192.168.0.100	10.10.1.100	TCP	54	1779 → 80 [ACK] Seq=593 Ack=3673 Win=66240 Len=0

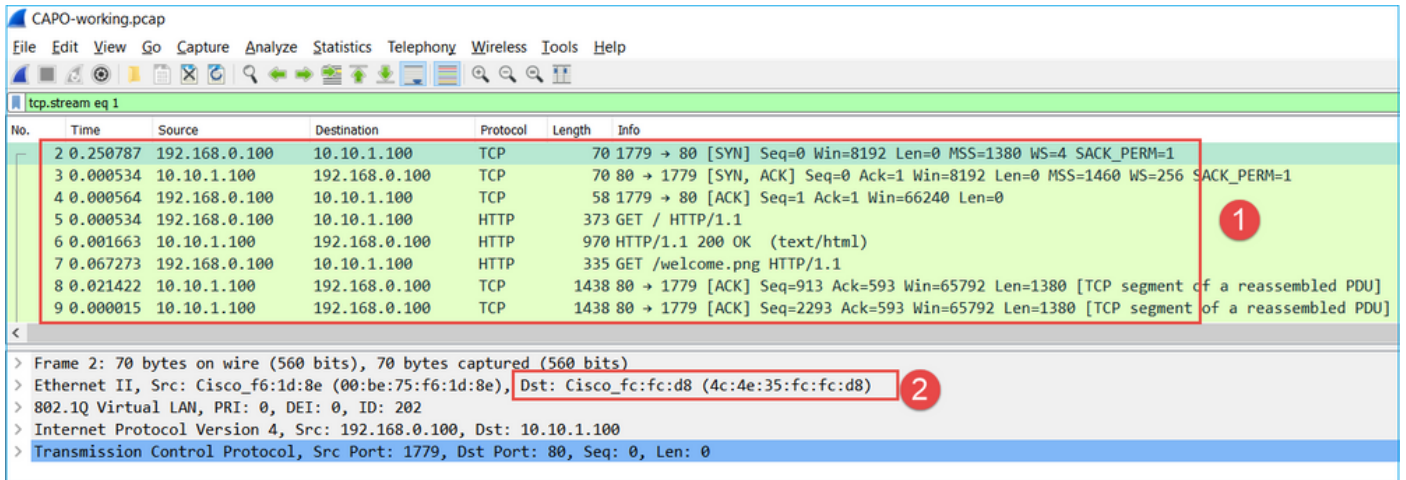
> Frame 2: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)  
> Ethernet II, Src: Cisco\_fc:fc:d8 (4c:4e:35:fc:fc:d8), Dst: Cisco\_f6:1d:ae (00:be:75:f6:1d:ae)  
> Internet Protocol Version 4, Src: 192.168.0.100, Dst: 10.10.1.100  
> Transmission Control Protocol, Src Port: 1779, Dst Port: 80, Seq: 0, Len: 0

要点:

1. TCP三次握手。
2. 双向数据交换。

3. 数据包之间无延迟（基于数据包之间的时间差）。
4. 源MAC是正确的下游设备。

在NGFW外部接口上捕获的流量如下图所示：



要点：

1. 与CAPI捕获中的数据相同。
2. 目的MAC是正确的上游设备。

捕获-非功能场景

从设备CLI捕获结果如下所示：

```
<#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内容：

```
<#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

这是CAPI捕获在Wireshark中的图像：

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 WS=4 SACK_PERM=1
2	0.250420	192.168.0.100	10.10.1.100	TCP	66	3172 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
3	2.745781	192.168.0.100	10.10.1.100	TCP	66	[TCP Retransmission] 3171 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
4	0.255074	192.168.0.100	10.10.1.100	TCP	66	[TCP Retransmission] 3172 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
5	5.751883	192.168.0.100	10.10.1.100	TCP	62	[TCP Retransmission] 3171 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 SACK_PERM=1
6	0.250070	192.168.0.100	10.10.1.100	TCP	62	[TCP Retransmission] 3172 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 SACK_PERM=1

Annotations in the image:  
1: Points to the 'MSS=1460' field in the first packet's info.  
2: Points to the 'Seq=0' field in the second packet's info.  
3: Points to the packet number '1' in the packet list.  
4: Points to the 'Src: 192.168.0.100' field in the packet details pane.

要点:



1. 只看到TCP SYN数据包 ( 无TCP三次握手 )。
2. 无法建立2个TCP会话 ( 源端口3171和3172 )。源客户端重新发送TCP SYN数据包。这些重新传输的数据包由Wireshark标识为TCP重新传输。
3. TCP重新传输每-3到6秒进行一次。
4. 源MAC地址来自正确的下游设备。

根据这2条捕获信息，可以得出以下结论：

- 特定5元组 ( src/dst IP、src/dst port、协议 ) 的数据包到达预期接口(INSIDE)上的防火墙。
- 数据包不会离开预期接口 ( 外部 ) 上的防火墙。

## 推荐的操作

本部分列出的操作旨在进一步缩小问题范围。

行动1.检查模拟数据包的跟踪。

使用packet-tracer工具查看防火墙如何处理数据包。如果数据包被防火墙访问策略丢弃，则模拟数据包的跟踪类似于以下输出：

```
<#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

行动2.检查实时数据包的跟踪。

启用数据包跟踪以检查防火墙如何处理实际TCP SYN数据包。默认情况下，仅跟踪前50个入口数据包：

```
<#root>
```

```
firepower#
```

```
capture CAPI trace
```

清除捕获缓冲区：

```
<#root>
```

```
firepower#
```

```
clear capture /all
```

如果数据包被防火墙访问策略丢弃，跟踪将类似于以下输出：

```
<#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 <m
```

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

行动3.检查FTD Lina日志。

要通过FMC在FTD上配置系统日志，请检查本文档：

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

强烈建议为FTD Lina日志配置外部系统日志服务器。如果没有配置远程系统日志服务器，请在故障排除时启用防火墙上的本地缓冲区日志。本示例中显示的日志配置是一个很好的起点：

```
<#root>
firepower#
show run logging

...
logging enable
logging timestamp
logging buffer-size 1000000
logging buffered informational
```

将终端寻呼机设置为24行，以便控制终端寻呼机：

```
<#root>
firepower#
terminal pager 24
```

清除捕获缓冲区：

```
<#root>
firepower#
clear logging buffer
```

测试连接并使用解析器过滤器检查日志。在本示例中，防火墙访问策略丢弃数据包：

```
<#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
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
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
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
```

行动4.检查防火墙ASP丢弃。

如果您怀疑防火墙丢弃了数据包，您可以在软件级别看到防火墙丢弃的所有数据包的计数器：

```
<#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
```

您可以启用捕获以查看所有ASP软件级别丢弃：

```
<#root>
firepower#
capture ASP type asp-drop all buffer 33554432 headers-only
```

---

 提示：如果您对数据包内容不感兴趣，则只能捕获数据包报头（仅报头选项）。这样，您可以在捕获缓冲区中捕获更多数据包。此外，可以将捕获缓冲区的大小（默认情况下为500KB）增加到最大32 MB的值（缓冲区选项）。最后，从FTD版本6.3开始，使用file-size选项可以配置捕获文件最多10GB。在这种情况下，只能看到pcap格式的捕获内容。

---

要检查捕获内容，可以使用过滤器缩小搜索范围：

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

在本例中，由于已在接口级别跟踪数据包，因此在ASP捕获中不会提及丢弃原因。请记住，只能在一个位置跟踪数据包（入口接口或ASP丢弃）。在这种情况下，建议使用多个ASP丢弃并设置特定ASP丢弃原因。下面是推荐方法：

1. 清除当前ASP丢弃计数器：

```
<#root>
firepower#
clear asp drop
```

2. 通过防火墙发送故障排除流程（运行测试）。

3. 再次检查ASP下拉计数器并记下增加的。

```
<#root>
firepower#
show asp drop
Frame drop:
  No route to host (
no-route
)
  Flow is denied by configured rule (
acl-drop
)
  234
  71
```

4. 为发现的特定丢包启用ASP捕获：

```
<#root>
firepower#
capture ASP_NO_ROUTE type asp-drop no-route
firepower#
capture ASP_ACL_DROP type asp-drop acl-drop
```

5. 通过防火墙发送故障排除流程（运行测试）。

6. 检查ASP捕获。在本例中，由于缺少路由，数据包被丢弃：

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

行动5.检查FTD Lina连接表。

有时您可能希望数据包输出接口“X”，但无论出于何种原因都会输出接口“Y”。防火墙出口接口确定基于以下操作顺序：

1. 已建立的连接查找
2. 网络地址转换(NAT)查找- UN-NAT (目标NAT) 阶段的优先级高于PBR和路由查找。
3. 基于策略的路由 (PBR)
4. 路由表查找

检查FTD连接表：

```
<#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

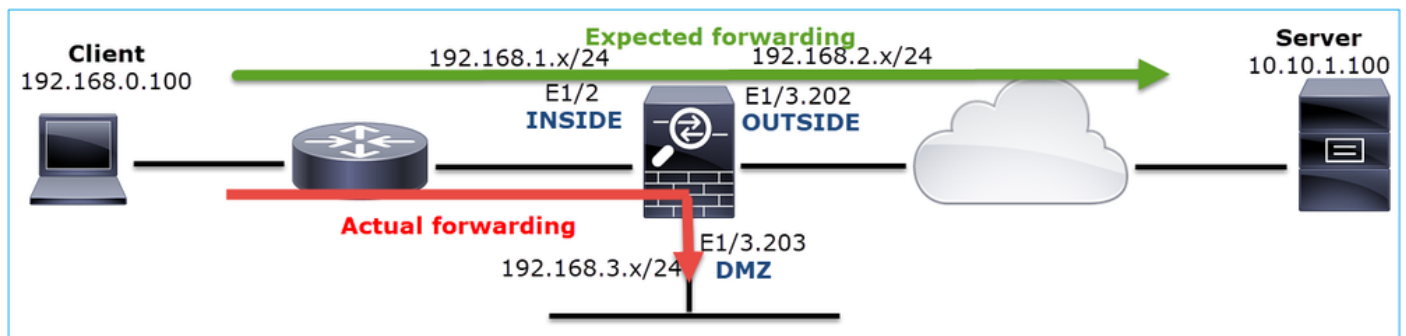
, idle 0:00:01, bytes 0, flags


aA N1

要点:

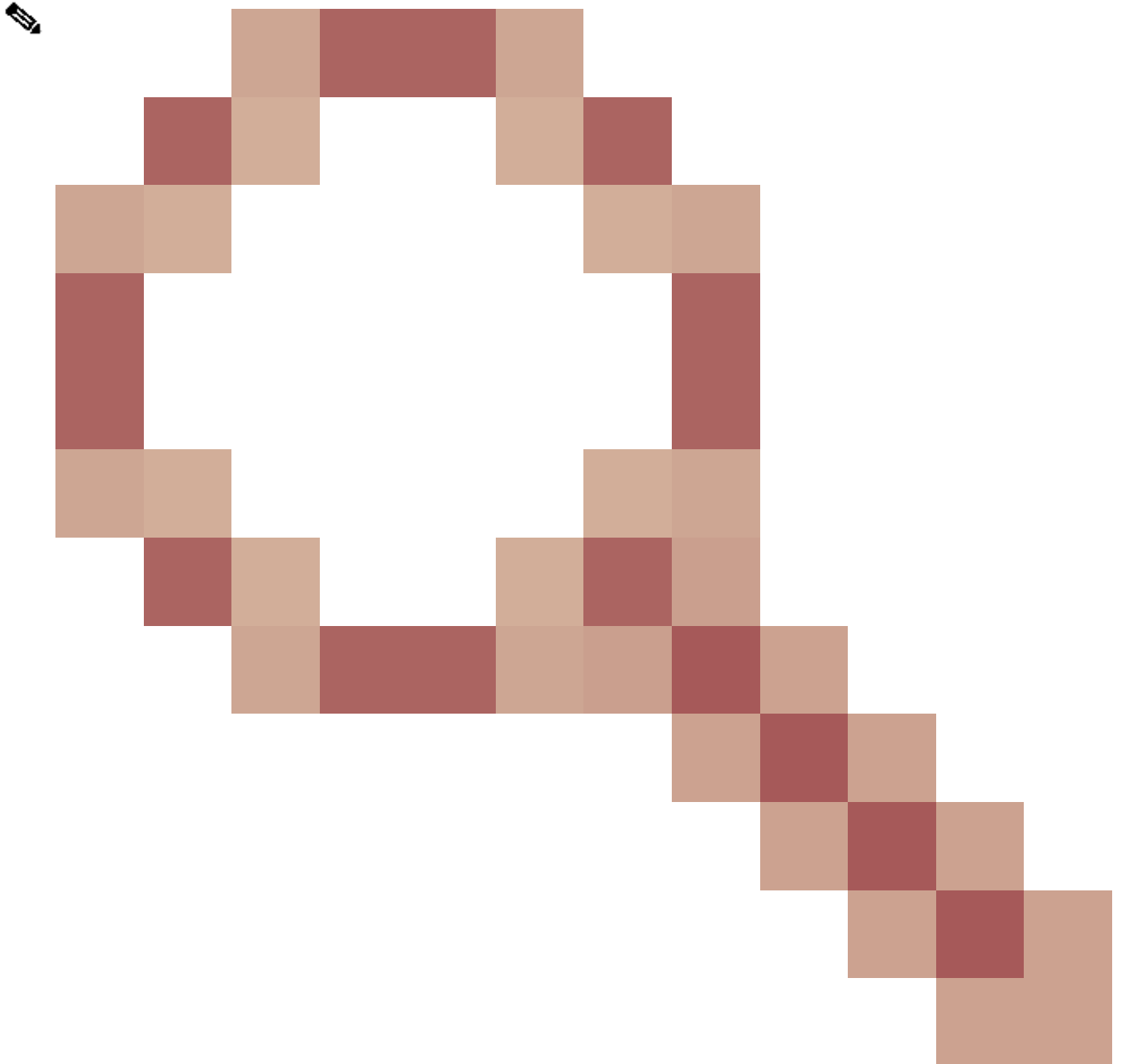
- 根据标志(Aa), 连接处于初期状态 ( 半打开-防火墙仅看到TCP SYN ) 。
- 根据源/目标端口, 入口接口为INSIDE, 出口接口为DMZ。

您可以在以下图片中看到此内容 :



 注意：由于所有FTD接口的安全级别都为0，因此show conn输出中的接口顺序基于接口号。具体而言，将具有较高vpif-num（虚拟平台接口编号）的接口选择为内部，而将具有较低vpif-num的接口选择为外部。您可以使用show interface detail命令来查看接口vpif值。相关增强，思科漏洞ID [CSCvi15290](https://cisco.com/security/center/content/CiscoSecurityAdvisory/CSCvi15290)





---

## ENH : FTD显示FTD“show conn”输出中的连接方向性


---

```
<#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

---

 注意：从Firepower软件版本6.5到ASA版本9.13.x，show conn long和show conn detail命令输出都提供有关连接发起方和响应方的信息

---

输出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), fla
```

```
Initiator: 192.168.1.100, Responder: 192.168.2.200
```

```
Connection lookup keyid: 228982375
```

输出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
```

此外，如果进行网络地址转换，show conn long 还会在括号内显示NAT转换后的IP：

```
<#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), fla
```

```
Initiator: 192.168.1.100, Responder: 192.168.2.222
```

```
Connection lookup keyid: 262895
```

行动6.检查防火墙地址解析协议(ARP)缓存。

如果防火墙无法解析下一跳，则防火墙将以静默方式丢弃原始数据包（本例中为TCP SYN）并继续发送ARP请求，直到解析下一跳。

要查看防火墙ARP缓存，请使用命令：

```
<#root>
firepower#
show arp
```

此外，要检查是否存在未解析的主机，您可以使用命令：

```
<#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
```

如果要进一步检查ARP操作，可以启用特定于ARP的捕获：

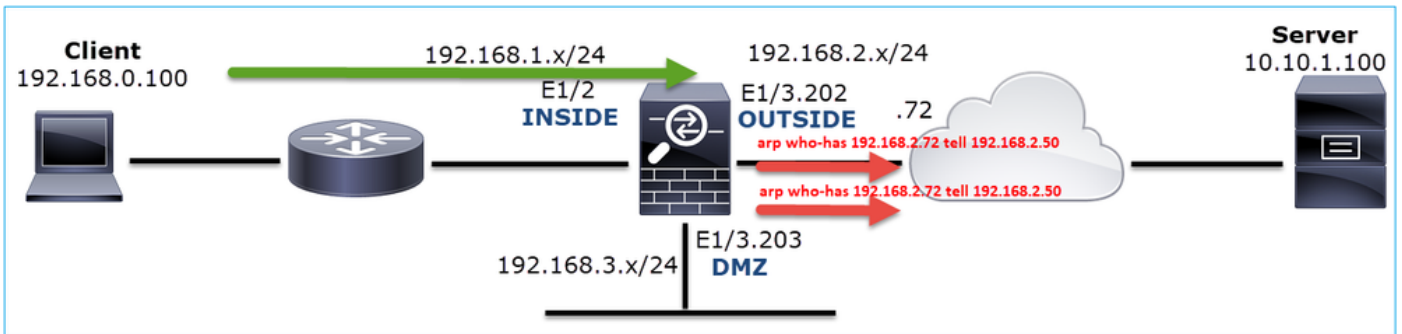
```
<#root>
firepower#
capture ARP ethernet-type arp interface OUTSIDE
```

```
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
```

在此输出中，防火墙(192.168.2.50)尝试解析下一跳(192.168.2.72)，但没有ARP应答



此处的输出显示具有正确ARP解析的功能场景：

```
<#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
```

如果没有ARP条目，则实时TCP SYN数据包的跟踪结果将显示：

<#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

...

Phase: 14

Type: FLOW-CREATION

Subtype:

Result: ALLOW

Config:

Additional Information:

New flow created with id 4814, packet dispatched to next module

...

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
```

从输出中看到，踪迹显示Action：allow，即使无法到达下一跳，并且防火墙以静默方式丢弃数据包！在这种情况下，还必须检查Packet Tracer工具，因为它能提供更准确的输出：

```
<#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
```

```
...
```

```
Phase: 14
```

```
Type: FLOW-CREATION
```

```
Subtype:
```

```
Result: ALLOW
```

```
Config:
```

```
Additional Information:
```

```
New flow created with id 4816, packet dispatched to next module
```

```
...
```

```
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),
```

在最新的ASA/Firepower版本中，以前的消息已优化为：

```
<#root>
```

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

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

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

### 可能的原因和建议的操作摘要

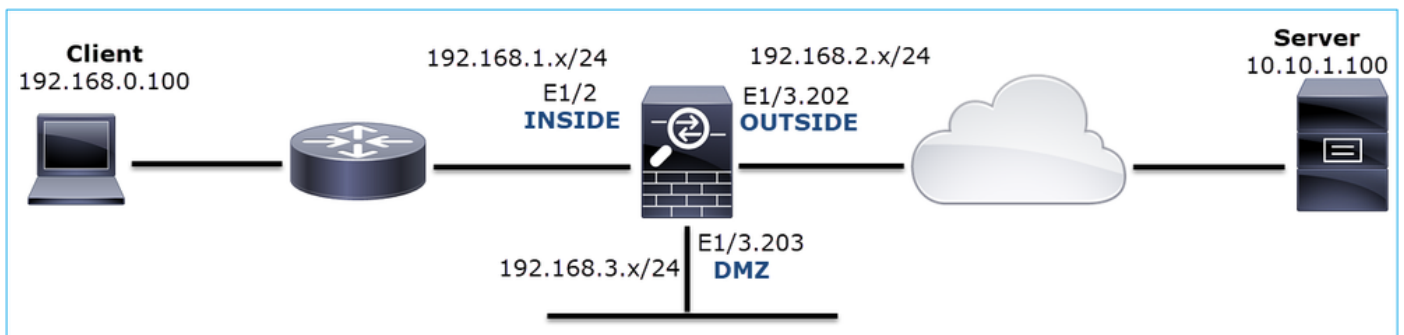
如果仅在入口接口上看到TCP SYN数据包，但未从预期出口接口发送任何TCP SYN数据包，则可能的原因包括：

可能的原因	推荐的操作
防火墙访问策略会丢弃数据包。	<ul style="list-style-type: none"><li>• 使用packet-tracer或capture w/trace查看防火墙如何处理数据包。</li><li>• 检查防火墙日志。</li><li>• 检查防火墙ASP丢弃(show asp drop或捕获类型asp-drop)。</li><li>• 检查FMC连接事件。这假设规则已启用日志记录。</li></ul>
捕获过滤器错误。	<ul style="list-style-type: none"><li>• 使用packet-tracer或capture w/trace查看是否存在可修改源IP或目标IP的NAT转换。在这种情况下，请调整捕获过滤器。</li><li>• show conn long 命令输出显示NATed IP。</li></ul>
将数据包发送到其他出口接口。	<ul style="list-style-type: none"><li>• 使用packet-tracer或capture w/trace查看防火墙如何处理数据包。请记住有关出口接口确定、当前连接、UN-NAT、PBR和路由表查找的操作顺序。</li><li>• 检查防火墙日志。</li><li>• 检查防火墙连接表(show conn)。</li></ul> <p>如果由于数据包与当前连接匹配而发送到错误的接口</p>

	，则使用clear conn address 命令并指定要清除的连接的5元组。
没有通往目的地的路由。	<ul style="list-style-type: none"> <li>• 使用packet-tracer或capture w/trace查看防火墙如何处理数据包。</li> <li>• 检查防火墙ASP丢弃(show asp drop)以了解无路由丢弃的原因。</li> </ul>
出口接口上没有ARP条目。	<ul style="list-style-type: none"> <li>• 检查防火墙ARP缓存(show arp)。</li> <li>• 使用Packet-tracer查看是否存在有效的邻接关系。</li> </ul>
出口接口关闭。	检查防火墙上show interface ip brief命令的输出，并验证接口状态。

## 案例 2.来自客户端的TCP SYN、来自服务器的TCP RST

下图显示拓扑：



问题说明：HTTP不起作用

受影响的流：

源IP：192.168.0.100

DST IP：10.10.1.100

协议：TCP 80

捕获分析

在FTD LINA引擎上启用捕获。



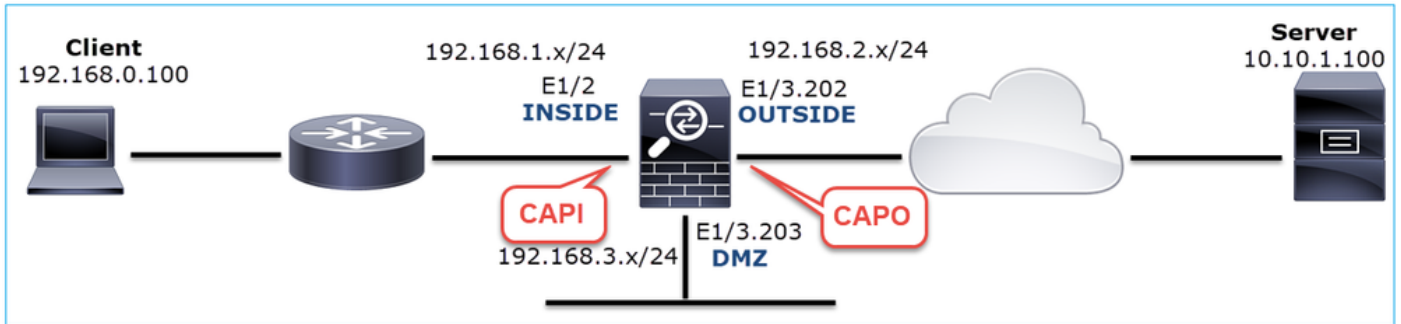
```
<#root>
```

```
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
```



捕获-非功能场景：

以下是设备CLI中的捕获结果：

```
<#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内容：

```
<#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内容 :

```
<#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>
```

下图显示了Wireshark中的CAPI捕获。

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	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	0.250094	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
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	66	[TCP Retransmission] 22196 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
5	0.000671	10.10.1.100	192.168.0.100	TCP	54	80 → 22196 [RST, ACK] Seq=2476911971 Ack=1 Win=0 Len=0
6	0.499380	192.168.0.100	10.10.1.100	TCP	62	[TCP Retransmission] 22196 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 SACK_PERM=1
7	0.000625	10.10.1.100	192.168.0.100	TCP	54	80 → 22196 [RST, ACK] Seq=2853655305 Ack=1 Win=0 Len=0
8	1.739729	192.168.0.100	10.10.1.100	TCP	66	[TCP Retransmission] 22195 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
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=8192 Len=0 MSS=1460 SACK_PERM=1
11	0.000671	10.10.1.100	192.168.0.100	TCP	54	80 → 22195 [RST, ACK] Seq=151733665 Ack=1 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: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

要点:

1. 源设备发送TCP SYN数据包。
2. 向源设备发送TCP RST。
3. 源设备重新传输TCP SYN数据包。
4. MAC地址正确 (在入口数据包上, 源MAC地址属于下游路由器, 目标MAC地址属于防火墙内部接口)。

下图显示Wireshark中的CAPO捕获 :

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=1380 WS=4 SACK_PERM=1
2	2019-10-11 07:20:36.994478	192.168.0.100	10.10.1.100	TCP	70	22196 → 80 [SYN] Seq=0 Win=8192 Len=0 MSS=1380 WS=4 SACK_PERM=1
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] Seq=0 Win=8192 Len=0 MSS=1380 WS=4 SACK_PERM=1
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] Seq=0 Win=8192 Len=0 MSS=1380 SACK_PERM=1
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 Win=8192 Len=0 MSS=1380 WS=4 SACK_PERM=1
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] Seq=0 Win=8192 Len=0 MSS=1380 SACK_PERM=1
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: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

要点:

1. 源设备发送TCP SYN数据包。
2. TCP RST到达外部接口。
3. 源设备重新传输TCP SYN数据包。
4. MAC地址正确 (在出口数据包上, 防火墙OUTSIDE是源MAC, 上游路由器是目标MAC)。

根据这2条捕获信息, 可以得出以下结论 :

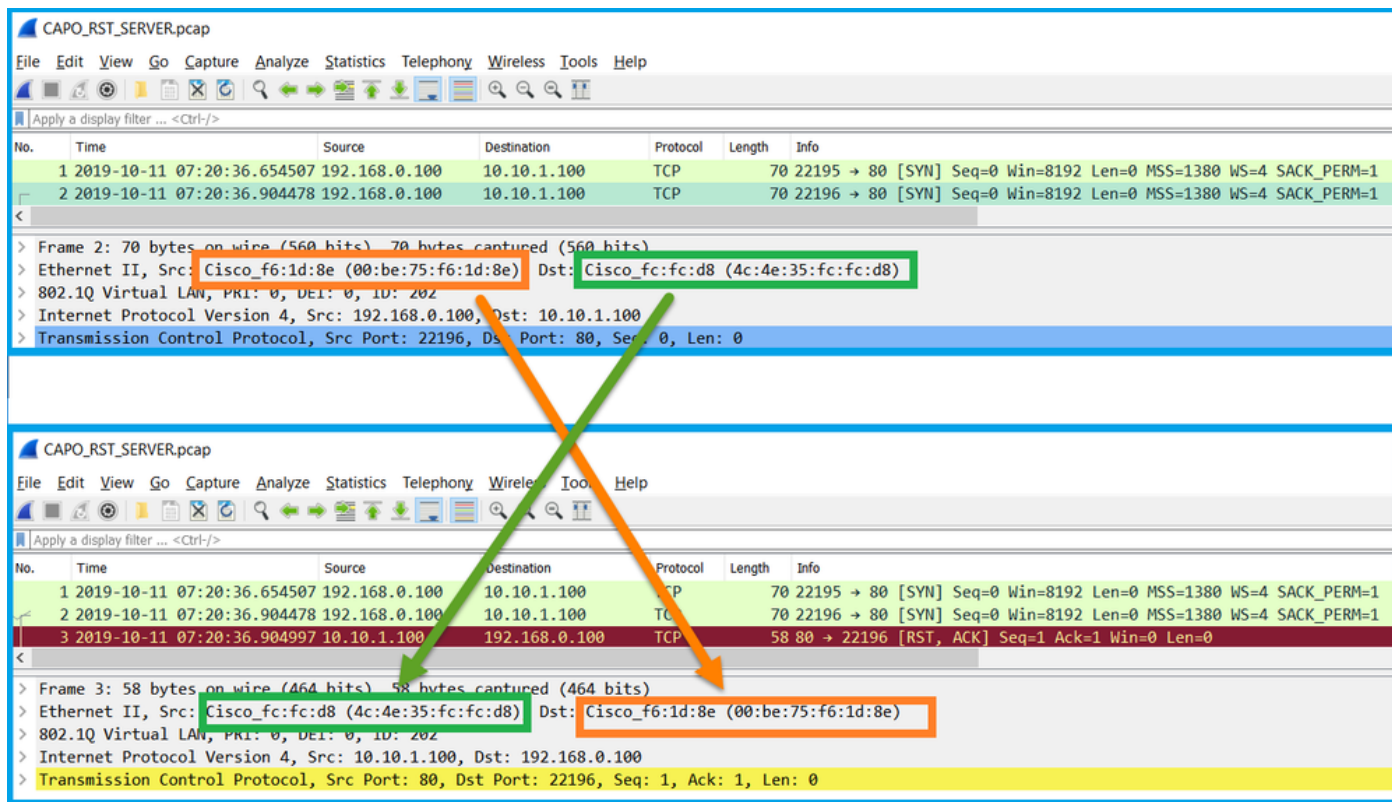
- 客户端和服务端之间的TCP三次握手没有完成
- 有一个到达防火墙出口接口的TCP RST
- 防火墙与适当的上游和下游设备“通信” (基于MAC地址)

推荐的操作

本部分列出的操作旨在进一步缩小问题范围。

行动1.检查发送TCP RST的源MAC地址。

验证TCP SYN数据包中的目标MAC与TCP RST数据包中的源MAC是否相同。

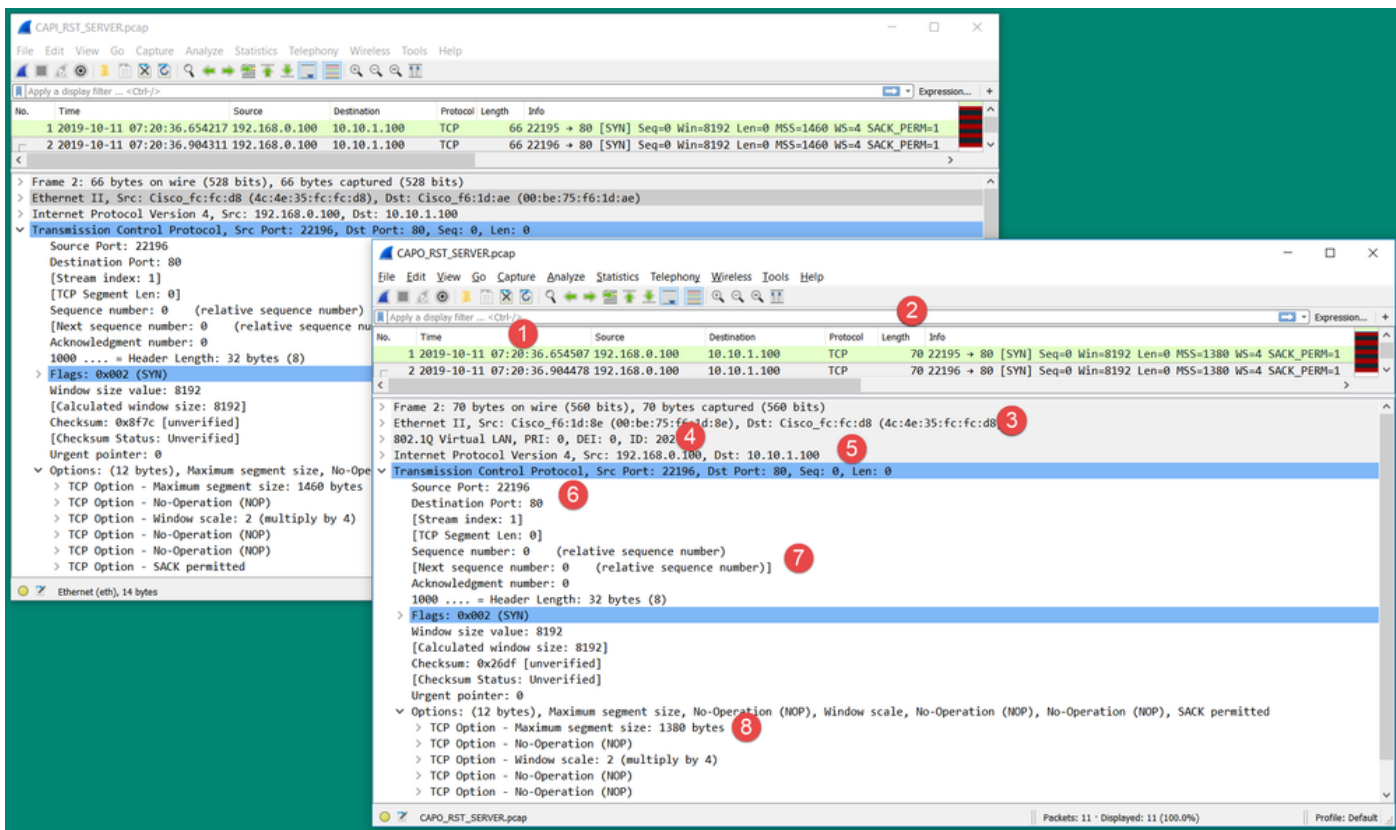


此检查旨在确认以下两点：

- 验证不存在非对称流。
- 验证MAC是否属于预期的上游设备。

行动2.比较入口和出口数据包。

目视比较Wireshark上的两个数据包，验证防火墙没有修改/损坏这些数据包。突出显示了一些预期差异。



## 要点:

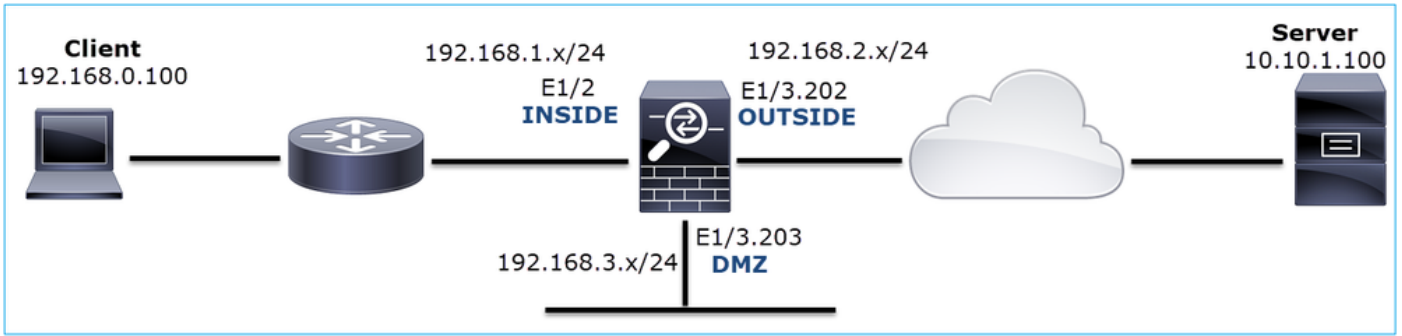
1. 时间戳不同。另一方面，这种差异必须小而合理。这取决于应用于数据包的功能和策略检查以及设备上的负载。
2. 如果防火墙仅在一侧添加/删除dot1Q报头，数据包的长度会特别不同。
3. MAC地址不同。
4. 如果捕获是在子接口上执行的，则可能会存在dot1Q报头。
5. 如果NAT或端口地址转换(PAT)应用于数据包，则IP地址会有所不同。
6. 如果对数据包应用NAT或PAT，则源端口或目标端口是不同的。
7. 如果您禁用Wireshark Relative Sequence Number选项，则会看到防火墙由于初始序列号(ISN)随机化而修改TCP序列号/确认号。
8. 某些TCP选项可能会被覆盖。例如，防火墙默认将TCP最大分段大小(MSS)更改为1380，以避免传输路径中的数据包分段。

行动3.在目的地捕获数据。

如果可能，在目的地本身捕获数据。如果无法实现，则尽可能靠近目的地捕获数据。此处的目标是验证谁发送了TCP RST (是目标服务器还是路径中的其他设备?)。

## 案例 3.来自一个终端的TCP三次握手+ RST

下图显示拓扑：



问题说明：HTTP不起作用

受影响的流：

源IP：192.168.0.100

DST IP：10.10.1.100

协议：TCP 80

捕获分析

在FTD LINA引擎上启用捕获。

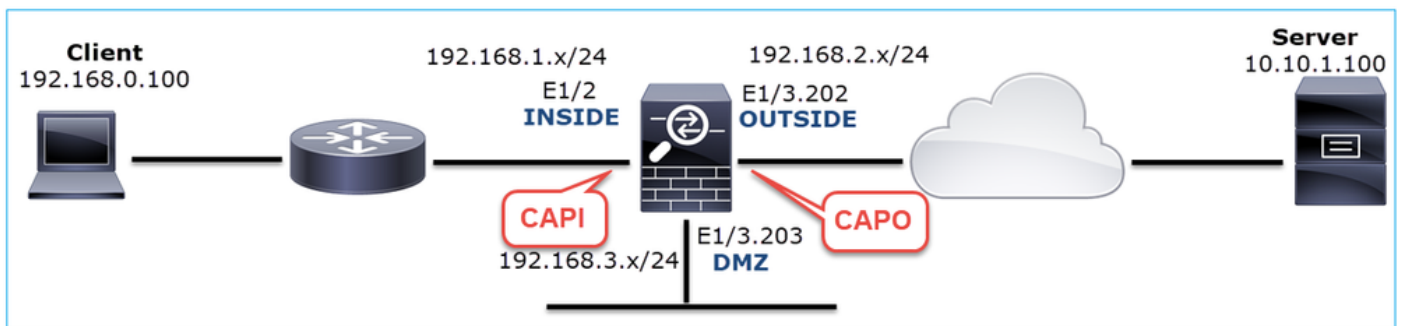
<#root>

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
```



捕获-非功能场景：

此问题在捕获中可能表现为几种不同的方式。

3.1 -来自客户端的TCP三次握手+延迟RST

如图所示，防火墙捕获CAPI和CAPO包含相同的数据包。



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 WS=256 SACK_PERM=1
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=8192 Len=0 MSS=1380 WS=256 SACK_PERM=1
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=3838911938 Win=66240 Len=0 SLE=3838911937 SRE=3838911938
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=65535 Len=0 MSS=1380 SACK_PERM=1
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 Len=0 SLE=3838911937 SRE=3838911938
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

## 要点:

1. TCP三次握手会通过防火墙。
2. 服务器重新传输SYN/ACK。
3. 客户端重新传输ACK。
4. 大约20秒后，客户端放弃并发送TCP RST。

## 推荐的操作

本部分列出的操作旨在进一步缩小问题范围。

行动1.尽可能靠近两个终端进行捕获。

防火墙捕获表明服务器未处理客户端ACK。这是基于以下事实：

- 服务器重新传输SYN/ACK。
- 客户端重新传输ACK。
- 客户端在任何数据之前发送TCP RST或FIN/ACK。

在服务器上执行捕获操作可显示问题。来自TCP三次握手的客户端ACK从未到达：

26	7.636612	192.168.0.100	10.10.1.100	TCP	66	55324→80 [SYN] Seq=433201323 Win=8192 Len=0 MSS=1380 WS=4 SAC...
29	7.637571	10.10.1.100	192.168.0.100	TCP	66	80→55324 [SYN, ACK] Seq=4063222169 Ack=433201324 Win=8192 Len...
30	7.930152	192.168.0.100	10.10.1.100	TCP	66	55325→80 [SYN] Seq=366197499 Win=8192 Len=0 MSS=1380 WS=4 SAC...
31	7.930221	10.10.1.100	192.168.0.100	TCP	66	80→55325 [SYN, ACK] Seq=2154790336 Ack=366197500 Win=8192 Len...
41	10.629868	192.168.0.100	10.10.1.100	TCP	66	[TCP Spurious Retransmission] 55324→80 [SYN] Seq=433201323 Wi...
42	10.633208	10.10.1.100	192.168.0.100	TCP	66	[TCP Retransmission] 80→55324 [SYN, ACK] Seq=4063222169 Ack=4...
44	10.945178	10.10.1.100	192.168.0.100	TCP	66	[TCP Retransmission] 80→55325 [SYN, ACK] Seq=2154790336 Ack=3...
60	16.636255	192.168.0.100	10.10.1.100	TCP	62	[TCP Spurious Retransmission] 55324→80 [SYN] Seq=433201323 Wi...
61	16.639145	10.10.1.100	192.168.0.100	TCP	62	[TCP Retransmission] 80→55324 [SYN, ACK] Seq=4063222169 Ack=4...
62	16.951195	10.10.1.100	192.168.0.100	TCP	62	[TCP Retransmission] 80→55325 [SYN, ACK] Seq=2154790336 Ack=3...

## 3.2 - TCP三次握手+来自客户端的延迟FIN/ACK +来自服务器的延迟RST

如图所示，防火墙捕获CAPI和CAPO包含相同的数据包。

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 SACK_PERM=1
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 Len=0 MSS=1460 WS=4 SACK_PERM=1
30	2019-10-13 17:07:09.854844	10.10.1.100	192.168.0.100	TCP	66	80 → 48299 [SYN, ACK] Seq=808763519 Ack=3239914003 Win=8192 Len=0 MSS=1380 WS=256 SACK_PERM=1
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	54	48299 → 80 [FIN, ACK] Seq=3239914003 Ack=808763520 Win=66240 Len=0
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=3239914003 Win=65535 Len=0 MSS=1380 SACK_PERM=1
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=808763520 Win=66240 Len=0 SLE=808763519 SRE=808763520
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=808763520 Win=66240 Len=0
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=808763520 Win=66240 Len=0
46	2019-10-13 17:07:27.858949	10.10.1.100	192.168.0.100	TCP	54	80 → 48299 [RST] Seq=808763520 Win=0 Len=0

## 要点:

1. TCP三次握手会通过防火墙。
2. 约5秒后，客户端发送FIN/ACK。
3. 大约20秒后，服务器会放弃并发送TCP RST。

根据捕获结果，可以断定，虽然存在通过防火墙的TCP三次握手，但似乎在一个终端上从未真正完成握手（重新传输表明此情况）。

### 推荐的操作

与例3.1相同

### 3.3 -来自客户端的TCP三次握手+延迟RST

如图所示，防火墙捕获CAPI和CAPO包含相同的数据包。

No.	Time	Source	Destination	Protocol	Length	Info
129	2019-10-13 17:09:20.513355	192.168.0.100	10.10.1.100	TCP	66	48355 → 80 [SYN] Seq=2581697538 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
130	2019-10-13 17:09:20.514011	10.10.1.100	192.168.0.100	TCP	66	80 → 48355 [SYN, ACK] Seq=1633018698 Ack=2581697539 Win=8192 Len=0 MSS=1460
131	2019-10-13 17:09:20.514438	192.168.0.100	10.10.1.100	TCP	54	48355 → 80 [ACK] Seq=2581697539 Ack=1633018699 Win=66240 Len=0
132	2019-10-13 17:09:39.473089	192.168.0.100	10.10.1.100	TCP	54	48355 → 80 [RST, ACK] Seq=2581697939 Ack=1633018699 Win=0 Len=0

### 要点:

1. TCP三次握手会通过防火墙。
2. 大约20秒后，客户端放弃并发送TCP RST。

根据这些捕获信息，可以得出以下结论：

- 5-20秒后，一个终端放弃并决定终止连接。

### 推荐的操作

与例3.1相同

### 3.4 -来自服务器的TCP三次握手+即时RST

如图所示，防火墙捕获CAPI和CAPO都包含这些数据包。

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=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
27	2019-10-13 17:07:07.105112	10.10.1.100	192.168.0.100	TCP	66	80 → 48300 [SYN, ACK] Seq=3757137497 Ack=2563435280 Win=8192 Len=0 MSS=1380
28	2019-10-13 17:07:07.105554	192.168.0.100	10.10.1.100	TCP	54	48300 → 80 [ACK] Seq=2563435280 Ack=3757137498 Win=66240 Len=0
41	2019-10-13 17:07:07.106325	10.10.1.100	192.168.0.100	TCP	54	80 → 48300 [RST] Seq=2563435280 Win=0 Len=0

### 要点:

1. TCP三次握手会通过防火墙。
2. 在ACK数据包过几毫秒后，服务器会发出一个TCP RST。

### 推荐的操作

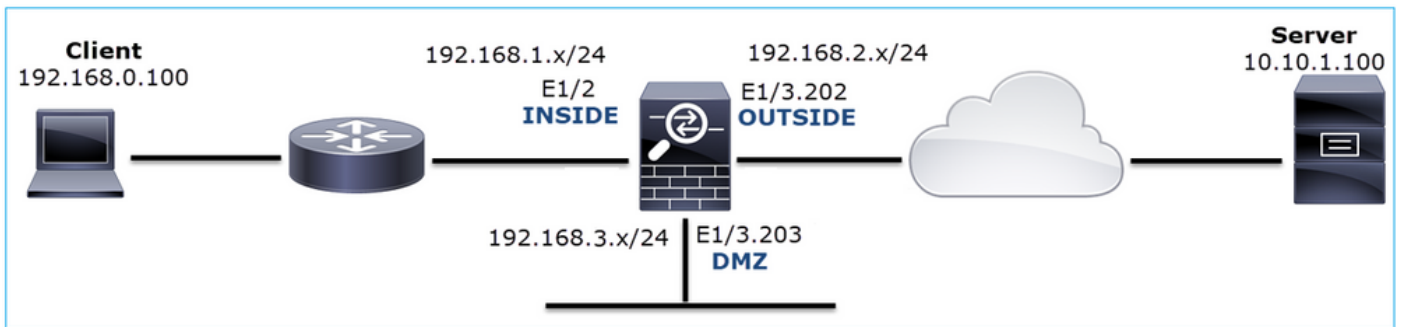
操作：尽可能靠近服务器进行捕获。

来自服务器的即时TCP RST可能表示发送TCP RST的路径中存在故障服务器或设备。捕获服务器本身并确定TCP RST的来源。



## 案例 4.来自客户端的TCP RST

下图显示拓扑：



问题说明：HTTP不起作用。

受影响的流：

源IP：192.168.0.100

DST IP：10.10.1.100

协议：TCP 80

捕获分析

在FTD LINA引擎上启用捕获。

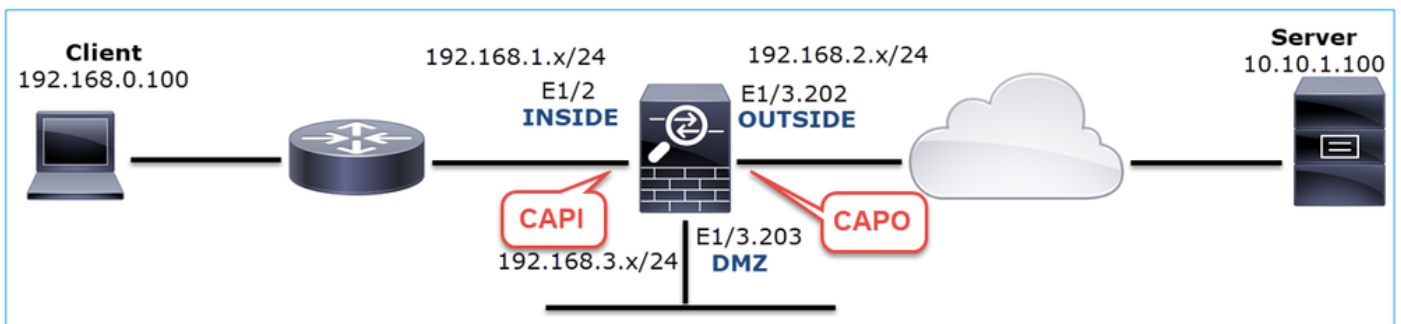
```
<#root>
```

```
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
```



捕获-非功能场景：

以下是CAPI内容。

<#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
2: 12:32:23.111307 192.168.0.100.47079 > 10.10.1.100.80: S 2486945841:2486945841(0) win 8192 <mss
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
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
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
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
14: 12:32:32.160956 192.168.0.100.47079 > 10.10.1.100.80: R 3000518858:3000518858(0) win 0
```

14 packets shown

以下是CAPO内容：

<#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 1386249852:138624985
2: 12:32:23.111429 802.1Q vlan#202 P0 192.168.0.100.47079 > 10.10.1.100.80: S 3000518857:300051885
3: 12:32:23.112405 802.1Q vlan#202 P0 192.168.0.100.47079 > 10.10.1.100.80: R 3514091874:351409187
4: 12:32:25.858125 802.1Q vlan#202 P0 192.168.0.100.47078 > 10.10.1.100.80: S 1386249852:138624985
5: 12:32:25.868729 802.1Q vlan#202 P0 192.168.0.100.47078 > 10.10.1.100.80: R 2968892337:296889233
6: 12:32:26.108240 802.1Q vlan#202 P0 192.168.0.100.47079 > 10.10.1.100.80: S 3822259745:382225974
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 4294058752:429405875
9: 12:32:31.860917 802.1Q vlan#202 P0 192.168.0.100.47078 > 10.10.1.100.80: R 1581733941:158173394
10: 12:32:32.160102 802.1Q vlan#202 P0 192.168.0.100.47079 > 10.10.1.100.80: S 4284301197:428430119
11: 12:32:32.160971 802.1Q vlan#202 P0 192.168.0.100.47079 > 10.10.1.100.80: R 502906918:502906918(
```

11 packets shown

防火墙日志显示：

<#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 (
Oct 13 2019 13:57:36: %FTD-6-302014: Teardown TCP connection 4869 for INSIDE:192.168.0.100/47741 to OUT
```

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 (
Oct 13 2019 13:57:39: %FTD-6-302014: Teardown TCP connection 4870 for INSIDE:192.168.0.100/47741 to OUT
```

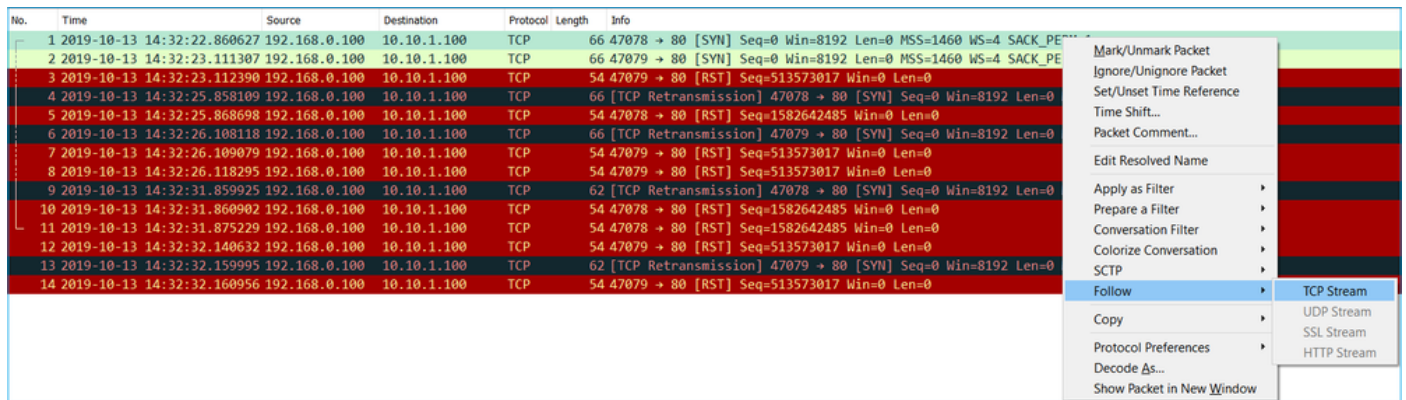
TCP Reset-O from INSIDE

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

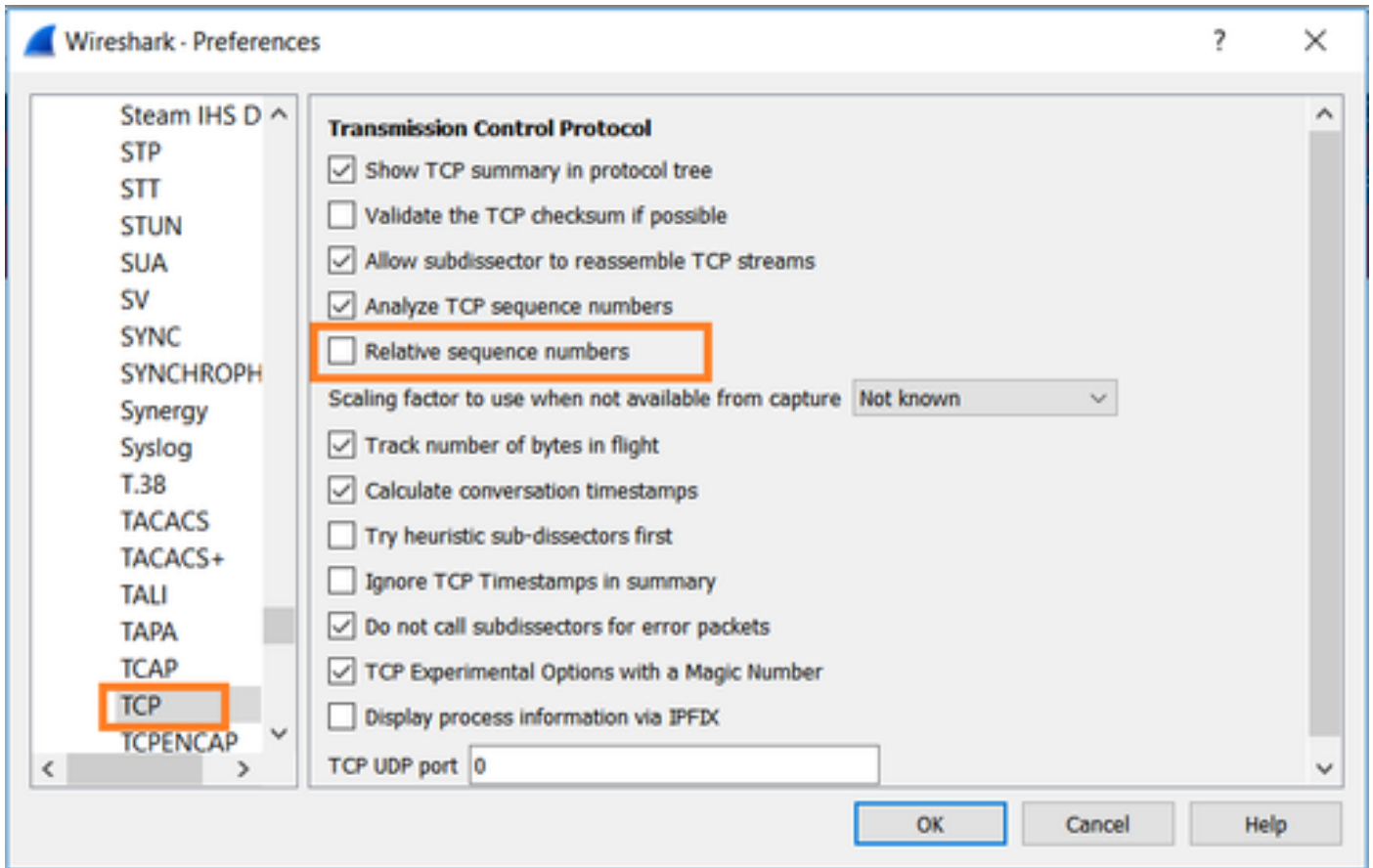
这些日志表明有一个到达防火墙INSIDE接口的TCP RST

Wireshark中的CAPI捕获：

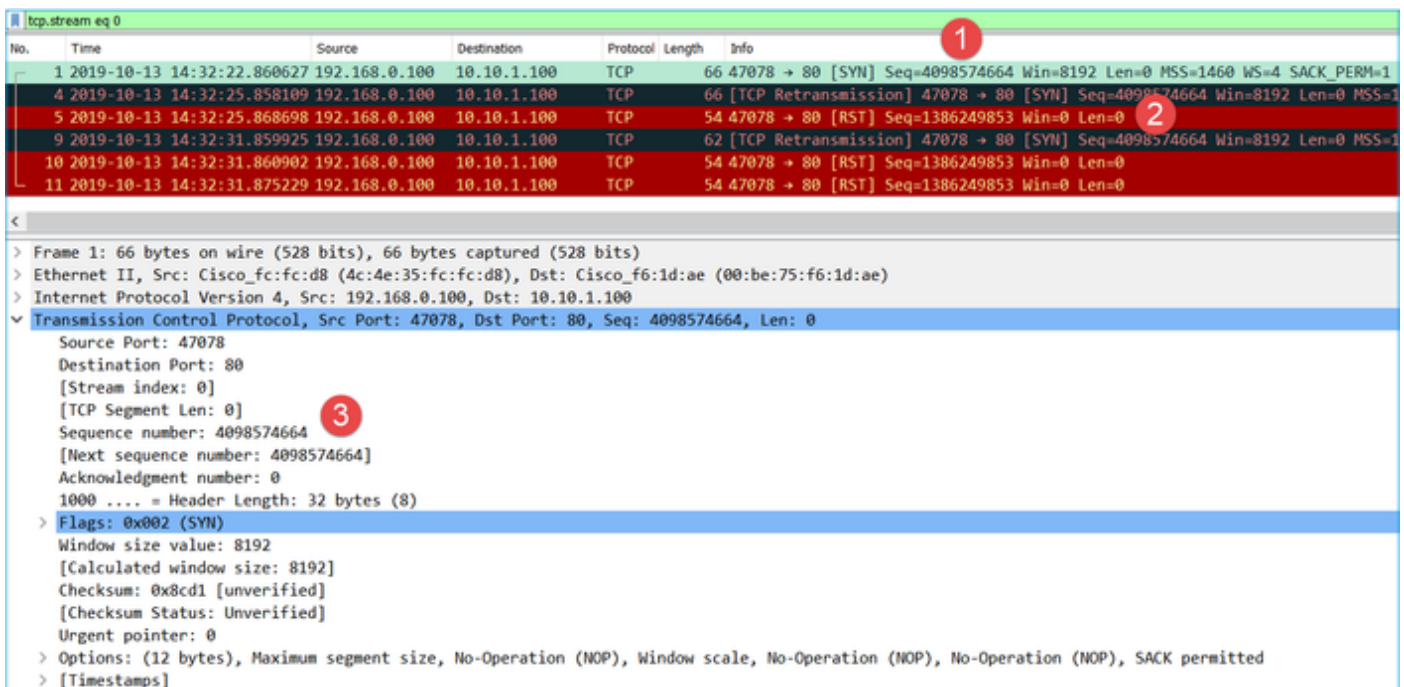
第一个TCP数据流之后，如图所示。



在Wireshark下，导航到编辑>首选项>协议>TCP，取消选择相对序列号选项（如图所示）。



下图显示CAPI捕获中的第一个流的内容：



要点：

1. 客户端发送TCP SYN数据包。
2. 客户端发送TCP RST数据包。
3. TCP SYN数据包的序列号值等于4098574664。

## CAPO捕获中的同一流包含：

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 Win=8192 Len=0 MSS=1380 WS=4 SACK_PERM=1
4	2019-10-13 14:32:25.858125	192.168.0.100	10.10.1.100	TCP	70	[TCP Retransmission] 47078 → 80 [SYN] Seq=1386249852 Win=8192 Len=0 MSS=1380
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 Len=0

<

> Frame 1: 70 bytes on wire (560 bits), 70 bytes captured (560 bits)  
> Ethernet II, Src: Cisco\_fc: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: 47078, Dst Port: 80, Seq: 1386249852, Len: 0

### 要点:

1. 客户端发送TCP SYN数据包。防火墙随机分配ISN。
2. 客户端发送TCP RST数据包。

根据这两个捕获结果，可以得出以下结论：

- 客户端和服务端之间没有TCP三次握手。
- 有一个来自客户端的TCP RST。CAPI捕获中的TCP RST序列号值为1386249853。

### 推荐的操作

本部分列出的操作旨在进一步缩小问题范围。

#### 行动1.捕获客户端。

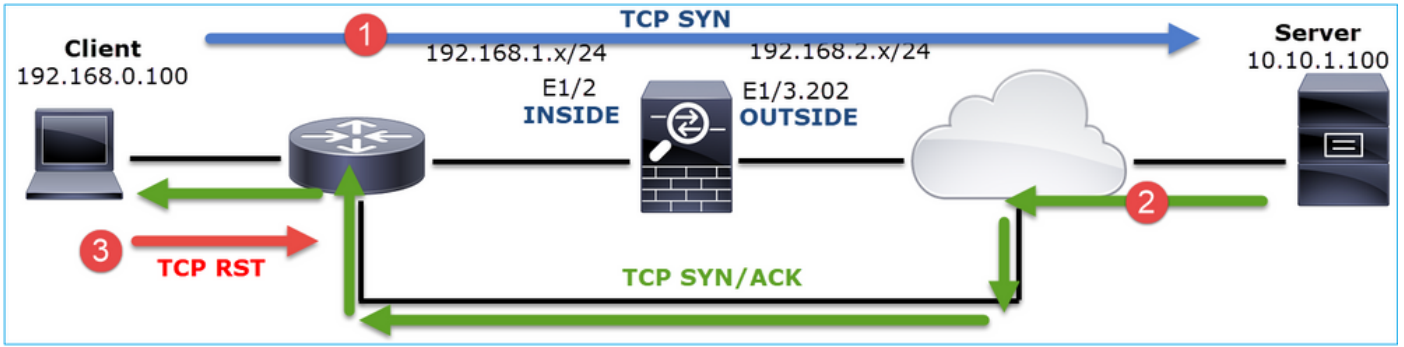
根据在防火墙上收集的捕获，有强烈的不对称流量指示。这是基于客户端发送值为1386249853 ( 随机ISN ) 的TCP RST这一事实：

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 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
29	9.037499	192.168.0.100	10.10.1.100	TCP	66	[TCP Retransmission] 47078→80 [SYN] Seq=4098574664 Win=8192 Len=0 MSS=1460 WS=4
30	9.048155	10.10.1.100	192.168.0.100	TCP	66	[TCP ACKed unseen segment] 80→47078 [SYN, ACK] Seq=1924342422 Ack=1386249853 Win=8192 Len=0 MSS=1460 WS=4 SACK_PERM=1
31	9.048184	192.168.0.100	10.10.1.100	TCP	54	47078→80 [RST] Seq=1386249853 Win=0 Len=0

### 要点:

1. 客户端发送TCP SYN数据包。序列号为4098574664，与在防火墙INSIDE接口(CAPI)上看到的序列号相同
2. 有一个ACK号为1386249853的TCP SYN/ACK ( 由于ISN随机化，预计会出现这种情况 )。在防火墙捕获中看不到此数据包
3. 客户端发送TCP RST，因为它预期会收到ACK号值为4098574665的SYN/ACK，但收到的值为1386249853

上述内容可以图形表示为：

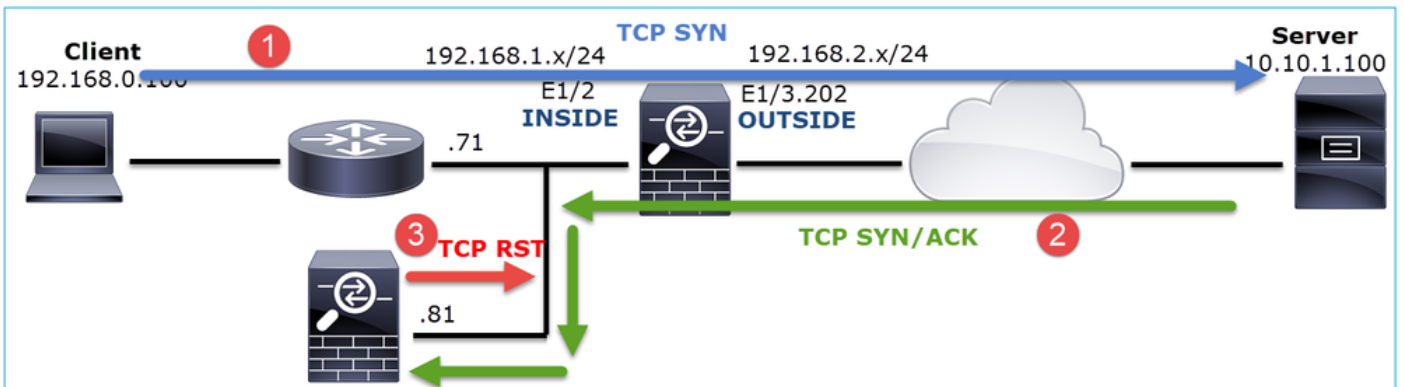


行动2.检查客户端和防火墙之间的路由。

确认：

- 捕获中看到的MAC地址是预期的MAC地址。
- 确保防火墙和客户端之间的路由对称。

在某些情况下，RST来自位于防火墙和客户端之间的设备，而内部网络中存在不对称路由。图中显示了一个典型案例：



在这种情况下，捕获包含此内容。注意TCP SYN数据包的源MAC地址与TCP RST的源MAC地址以及TCP SYN/ACK数据包的目的MAC地址之间的区别：

```
<#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,
```

```
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,
```

```
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
```

```
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)  
...

## 案例 5.缓慢TCP传输 ( 场景1 )

问题说明:

主机10.11.4.171和主机10.77.19.11之间的SFTP传输缓慢。虽然两台主机之间的最小带宽(BW)为100 Mbps，但传输速度不超过5 Mbps。

同时，主机10.11.2.124和172.25.18.134之间的传输速度要快得多。

背景理论:

单个TCP流的最大传输速度由带宽延迟积(BDP)决定。图中显示所使用的公式：

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

有关BDP的更多详细信息，请点击此处查看资源：

- [为什么即使链路为1Gbps，您的应用程序也只使用10Mbps？](#)
- [BRKSEC-3021 -高级-最大化防火墙性能](#)

场景 1.缓慢传输

下图显示拓扑：



受影响的流：

源IP：10.11.4.171



DST IP : 10.77.19.11

协议 : SFTP(FTP over SSH)

## 捕获分析

在FTD LINA引擎上启用捕获 :

```
<#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
```

---

 **警告** : FP1xxx和FP21xx捕获上的LINA影响通过FTD的数据流的传输速率。排除性能 ( 通过FTD传输缓慢 ) 故障时 , 请勿在FP1xxx和FP21xxx平台上启用LINA捕获。除了在源主机和目的主机上捕获数据外 , 还应使用SPAN或HW分路器设备。思科漏洞ID [CSCvo30697](#)中记录了此问题。

---

```
<#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.
```

## 推荐的操作

本部分列出的操作旨在进一步缩小问题范围。

## 往返时间(RTT)计算

首先 , 确定传输流并遵循它 :



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

View	Info
Follow	TCP Stream
Copy	UDP Stream
Protocol Preferences	SSL Stream
	HTTP Stream

更改Wireshark视图，显示自上次显示数据包以来的秒数。这简化了RTT的计算：

File	Edit	View	Go	Capture	Analyze	Statistics	Telephony	Wireless	Tools	Help
<ul style="list-style-type: none"> <li>Main Toolbar</li> <li>Filter Toolbar</li> <li>Status Bar</li> <li>Full Screen (F11)</li> <li>Packet List</li> <li>Packet Details</li> <li>Packet Bytes</li> <li>Time Display Format <ul style="list-style-type: none"> <li>Date and Time of Day (1970-01-01 01:02:03.123456) Ctrl+Alt+1</li> <li>Year, Day of Year, and Time of Day (1970/001 01:02:03.123456)</li> <li>Time of Day (01:02:03.123456) Ctrl+Alt+2</li> <li>Seconds Since 1970-01-01 Ctrl+Alt+3</li> <li>Seconds Since Beginning of Capture Ctrl+Alt+4</li> <li>Seconds Since Previous Captured Packet Ctrl+Alt+5</li> <li>Seconds Since Previous Displayed Packet Ctrl+Alt+6</li> </ul> </li> <li>Name Resolution</li> <li>Zoom</li> <li>Expand Subtrees (Shift+Right)</li> <li>Collapse Subtrees (Shift+Left)</li> <li>Expand All (Ctrl+Right)</li> </ul>										

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 Len=0 MSS=1460 WS=1 SACK_PERM=1
2	0.072521	10.77.19.11	10.11.4.171	TCP	70	49680	22 → 39744 [SYN, ACK] Seq=835172681 Ack=1737026094 Win=49680 Len=0 MSS=1380 WS=1 SACK_PERM=1
3	0.000168	10.11.4.171	10.77.19.11	TCP	58	49680	39744 → 22 [ACK] Seq=1737026094 Ack=835172682 Win=49680 Len=0
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 Win=49680 Len=0
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 Win=49680 Len=0
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 Win=49680 Len=0
11	0.030165	10.77.19.11	10.11.4.171	TCP	58	49680	22 → 39744 [ACK] Seq=835173384 Ack=1737026596 Win=49680 Len=0
12	0.000168	10.11.4.171	10.77.19.11	SSHv2	82	49680	Client: Diffie-Hellman Group Exchange Request

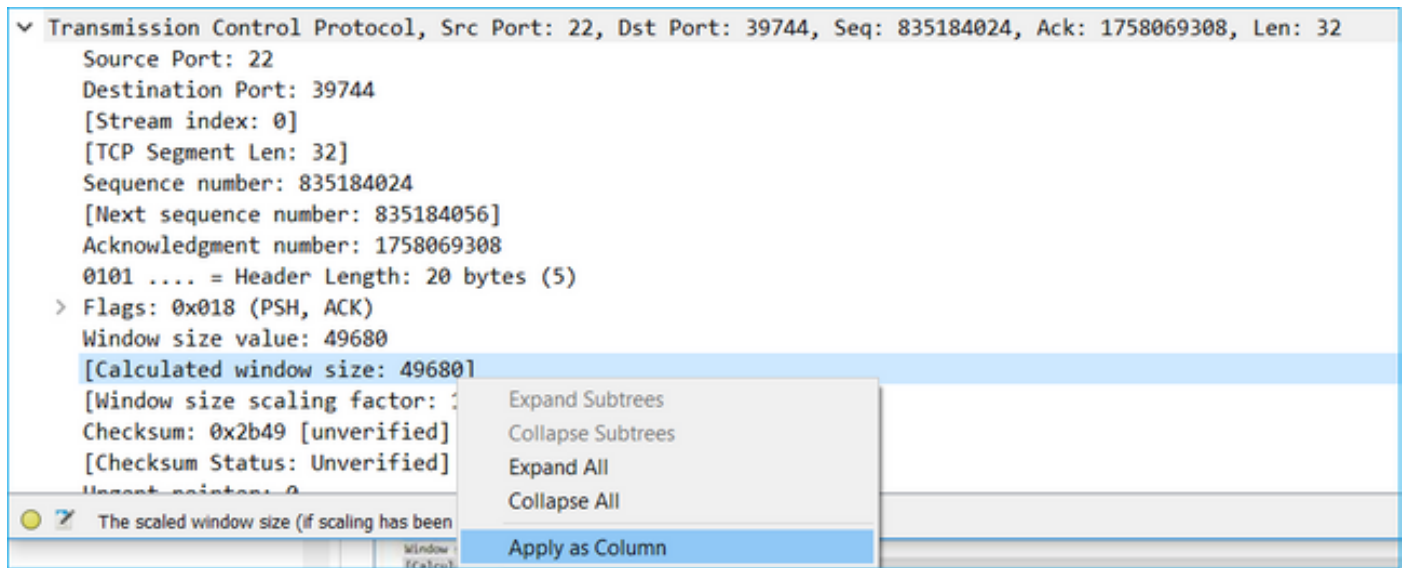
RTT可以通过在2个数据包交换（一个指向源，一个指向目标）之间加上时间值来计算。在这种情况下，数据包#2显示防火墙与发送SYN/ACK数据包的设备（服务器）之间的RTT。Packet #3显示防火墙与发送ACK数据包的设备（客户端）之间的RTT。增加两个数字可以很好地估计端到端RTT：

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 Len=0 MSS=1460 WS=1 SACK_PERM=1
2	0.072521	10.77.19.11	10.11.4.171	TCP	70	49680	22 → 39744 [SYN, ACK] Seq=835172681 Ack=1737026094 Win=49680 Len=0 MSS=1380 WS=1 SACK_PERM=1
3	0.000168	10.11.4.171	10.77.19.11	TCP	58	49680	39744 → 22 [ACK] Seq=1737026094 Ack=835172682 Win=49680 Len=0
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 Win=49680 Len=0
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 Win=49680 Len=0
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 Win=49680 Len=0
11	0.030165	10.77.19.11	10.11.4.171	TCP	58	49680	22 → 39744 [ACK] Seq=835173384 Ack=1737026596 Win=49680 Len=0
12	0.000168	10.11.4.171	10.77.19.11	SSHv2	82	49680	Client: Diffie-Hellman Group Exchange Request

RTT ≈ 80毫秒

## TCP窗口大小计算

展开TCP数据包，然后展开TCP报头，再选择Calculated window size，然后选择Apply as Column：



检查Calculated window size value列，查看在TCP会话期间的最大窗口大小值。您还可以选择列名称并对值排序。

如果测试文件下载(server > client)，则必须检查服务器通告的值。服务器通告的最大窗口大小值决定了获得的最大传输速度。

在这种情况下，TCP窗口大小为≈ 50000字节

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 Ack=83
24...	0.000077	10.77.19.11	10.11.4.171	TCP	58	49680	22 → 39744 [FIN, ACK] Seq=835184152 Ac
24...	0.071605	10.77.19.11	10.11.4.171	TCP	58	49680	22 → 39744 [ACK] Seq=835184152 Ack=175
24...	0.000153	10.11.4.171	10.77.19.11	TCP	58	49680	39744 → 22 [FIN, ACK] Seq=1758069340 A
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 Ack=83
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)

根据这些值，并使用“带宽延迟乘积”公式，您可以获得在这些条件下可以达到的最大理论带宽： $50000 * 8 / 0.08 = 5 \text{ Mbps}$ 的最大理论带宽。

这与客户端在此案例中所体验的情景相匹配。

仔细检查TCP三次握手。两端（更重要的是服务器）都通告窗口缩放值0，这意味着 $2^0 = 1$ （无窗口缩放）。这会对传输速率产生负面影响：

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 Len=0 MSS=1460 WS=1 SACK_PERM=1
2	0.072521	10.77.19.11	10.11.4.171	TCP	70	49680 22 → 39744	[SYN, ACK] Seq=835172681 Ack=1737026094 Win=49680 Len=0 MSS=1384 WS=1 SACK_PERM=1

```

> Frame 2: 70 bytes on wire (560 bits), 70 bytes captured (560 bits) on interface 0
> 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
> 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
  > Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Operation (NOP), SACK permitted
    > 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)

```

此时，需要在服务器上捕获数据，确认是通告window scale = 0的服务器，然后重新配置它（检查服务器文档以了解如何执行此操作）。

## 场景 2：快速传输

现在，让我们来看看好方案（通过同一网络快速传输）：

拓扑：



利益流向：

源IP：10.11.2.124

DST IP：172.25.18.134

协议：SFTP(FTP over SSH)

在FTD LINA引擎上启用捕获

```
<#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
```

往返时间(RTT)计算：在这种情况下，RTT≈300毫秒。

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窗口大小计算：服务器通告TCP窗口缩放系数7。

```
> Internet Protocol Version 4, Src: 172.25.18.134, Dst: 10.11.2.124
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
  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]
```

服务器的TCP窗口大小为≈ 1600000字节：

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 pack
23...	0.000076	172.25.18.134	10.11.2.124	SSHv2	118	12854	1645312	Server: Encrypted pack
23...	0.000351	172.25.18.134	10.11.2.124	SSHv2	118	12854	1645312	Server: Encrypted pack
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=

根据这些值，带宽延迟乘积公式可得出：

$$1600000 * 8 / 0.3 = 43 \text{ Mbps最大理论传输速度}$$



## 案例 6.缓慢TCP传输 ( 场景2 )

问题描述：通过防火墙的FTP文件传输（下载）速度缓慢。

下图显示拓扑：



受影响的流：

源IP：192.168.2.220

DST IP：192.168.1.220

协议：FTP

捕获分析

在FTD LINA引擎上启用捕获。

<#root>

firepower#

```
capture CAPI type raw-data buffer 33554432 interface INSIDE match tcp host 192.168.2.220 host 192.168.1.220
```

firepower#

```
cap CAPO type raw-data buffer 33554432 interface OUTSIDE match tcp host 192.168.2.220 host 192.168.1.220
```

选择FTP-DATA数据包，然后按照FTD内部捕获(CAPI)的FTP数据信道操作：

Seq	Time	Source IP	Destination IP	Protocol	Details
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	88 → 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	(RETR file15mb)

## FTP-DATA流内容：

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 TSecr=0 WS=128
28	1.026534	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=1 TSval=3577289526 TSecr=0 WS=128
29	1.001564	192.168.1.220	192.168.2.220	TCP	74 2388 → 54494 [SYN, ACK] Seq=2669998978 Ack=1884231612 Win=8192 Len=0 MSS=1260 WS=256 SACK_PERM=1 TSval=4264384 TSecr=3577288500
30	0.000488	192.168.2.220	192.168.1.220	TCP	66 54494 → 2388 [ACK] Seq=1884231612 Ack=2669998979 Win=29312 Len=0 TSval=3577291508 TSecr=4264384
34	0.001617	192.168.1.220	192.168.2.220	FTP-DATA	1314 FTP Data: 1248 bytes (PASV) (RETR file5mb)
35	0.000351	192.168.2.220	192.168.1.220	TCP	66 54494 → 2388 [ACK] Seq=1884231612 Ack=2669999027 Win=32128 Len=0 TSval=3577291510 TSecr=4264384
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 file5mb)
37	0.000961	192.168.1.220	192.168.2.220	FTP-DATA	1314 FTP Data: 1248 bytes (PASV) (RETR file5mb)
38	0.000198	192.168.2.220	192.168.1.220	TCP	78 [TCP Window Update] 54494 → 2388 [ACK] Seq=1884231612 Ack=2669998927 Win=35072 Len=0 TSval=3577291511 TSecr=4264384 SLE=2669992175 SRE=2669993423
39	0.000077	192.168.2.220	192.168.1.220	TCP	78 [TCP Window Update] 54494 → 2388 [ACK] Seq=1884231612 Ack=2669998927 Win=37888 Len=0 TSval=3577291511 TSecr=4264384 SLE=2669992175 SRE=2669994671
40	0.309096	192.168.1.220	192.168.2.220	TCP	1314 [TCP Out-Of-Order] 2388 → 54494 [ACK] Seq=2669999027 Ack=1884231612 Win=66048 Len=1248 TSval=4264415 TSecr=3577291511
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=4264415
42	0.000489	192.168.1.220	192.168.2.220	FTP-DATA	1314 FTP Data: 1248 bytes (PASV) (RETR file5mb)
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 file5mb)
44	0.000077	192.168.1.220	192.168.2.220	FTP-DATA	1314 FTP Data: 1248 bytes (PASV) (RETR file5mb)
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=4264415
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=3577291821 TSecr=4264415 SLE=2669997167 SRE=2669999663
47	0.000504	192.168.1.220	192.168.2.220	FTP-DATA	1314 FTP Data: 1248 bytes (PASV) (RETR file5mb)
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=3577291822 TSecr=4264415 SLE=2669997167 SRE=2670000911
49	0.018176	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 TSval=4264507 TSecr=3577291822
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=4264507
51	0.000519	192.168.1.220	192.168.2.220	FTP-DATA	1314 FTP Data: 1248 bytes (PASV) (RETR file5mb)
52	0.000061	192.168.2.220	192.168.1.220	FTP-DATA	1314 FTP Data: 1248 bytes (PASV) (RETR file5mb)
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 file5mb)
54	0.000015	192.168.1.220	192.168.2.220	FTP-DATA	1314 FTP Data: 1248 bytes (PASV) (RETR file5mb)
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=4264507
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=4264507
57	0.000183	192.168.1.220	192.168.2.220	FTP-DATA	1314 FTP Data: 1248 bytes (PASV) (RETR file5mb)
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=3577292742 TSecr=4264507 SLE=2670004655 SRE=2670007151
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=3577292743 TSecr=4264507 SLE=2670004655 SRE=2670008399
60	0.000000	192.168.1.220	192.168.2.220	FTP-DATA	1314 FTP Data: 1248 bytes (PASV) (RETR file5mb)

## CAPO捕获内容：

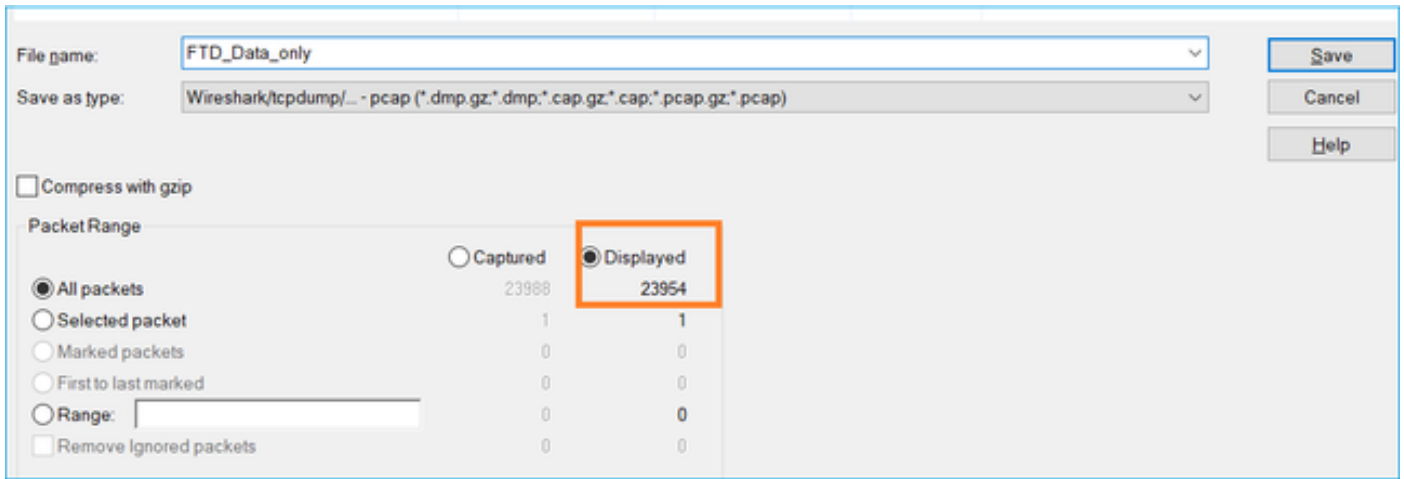
31	0.000000	192.168.2.220	192.168.1.220	TCP	74 54494 → 2388 [SYN] Seq=2157030681 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=3577288500 TSecr=0 WS=128
33	1.026534	192.168.2.220	192.168.1.220	TCP	74 [TCP Retransmission] 54494 → 2388 [SYN] Seq=2157030681 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=3577289526 TSecr=0 WS=128
34	1.981490	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_PERM=1 TSval=4264384 TSecr=3577288500
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=4264384
38	0.001328	192.168.1.220	192.168.2.220	FTP-DATA	1314 FTP Data: 1248 bytes (PASV) (RETR file5mb)
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 TSecr=4264384
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 file5mb)
42	0.000046	192.168.1.220	192.168.2.220	FTP-DATA	1314 FTP Data: 1248 bytes (PASV) (RETR file5mb)
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 TSval=3577291511 TSecr=4264384 SLE=2224319408 SRE=2224320656
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 TSval=3577291511 TSecr=4264384 SLE=2224319408 SRE=2224321904
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 TSval=4264415 TSecr=3577291511
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=4264415
47	0.000412	192.168.1.220	192.168.2.220	FTP-DATA	1314 FTP Data: 1248 bytes (PASV) (RETR file5mb)
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 file5mb)
49	0.000076	192.168.1.220	192.168.2.220	FTP-DATA	1314 FTP Data: 1248 bytes (PASV) (RETR file5mb)
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=4264415
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 TSval=3577291821 TSecr=4264415 SLE=2224324400 SRE=2224326896
52	0.000412	192.168.1.220	192.168.2.220	FTP-DATA	1314 FTP Data: 1248 bytes (PASV) (RETR file5mb)
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 TSval=3577291822 TSecr=4264415 SLE=2224324400 SRE=2224328144
54	0.018019	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 TSval=4264507 TSecr=3577291822
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=4264507
56	0.000457	192.168.1.220	192.168.2.220	FTP-DATA	1314 FTP Data: 1248 bytes (PASV) (RETR file5mb)
57	0.000061	192.168.2.220	192.168.1.220	FTP-DATA	1314 FTP Data: 1248 bytes (PASV) (RETR file5mb)
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 file5mb)
59	0.000000	192.168.1.220	192.168.2.220	FTP-DATA	1314 FTP Data: 1248 bytes (PASV) (RETR file5mb)
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=4264507
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=4264507
62	0.000122	192.168.1.220	192.168.2.220	FTP-DATA	1314 FTP Data: 1248 bytes (PASV) (RETR file5mb)
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 TSval=3577292742 TSecr=4264507 SLE=2224331888 SRE=2224334384
64	0.000107	192.168.1.220	192.168.2.220	FTP-DATA	1314 FTP Data: 1248 bytes (PASV) (RETR file5mb)

## 要点：

1. 存在TCP乱序(OOO)数据包。
2. 存在TCP重新传输。
3. 存在数据包丢失 ( 丢弃的数据包 ) 的指示。



提示：导航到File > Export Specified Packets时保存捕获。然后仅保存显示的数据包范围



## 推荐的操作

本部分列出的操作旨在进一步缩小问题范围。

行动1.确定数据包丢失位置。

在这种情况下，您必须同时执行捕获，并使用分治法来识别导致数据包丢失的网段。从防火墙的角度来看，主要有3种场景：

1. 数据包丢失是由防火墙本身导致的。
2. 数据包丢失导致在防火墙设备的下游（从服务器到客户端的方向）。
3. 数据包丢失导致上游到防火墙设备（从客户端到服务器的方向）。

防火墙导致的数据包丢失：为了确定数据包丢失是否由防火墙引起，需要将入口捕获与出口捕获进行比较。有很多方法可以比较两种不同的捕获。本部分演示了一种执行此任务的方法。

比较2次捕获以确定数据包丢失的过程

步骤1:确保2个捕获包含来自同一时间窗口的数据包。这意味着一个捕获中一定没有数据包是在另一个捕获之前或之后捕获的。有几种方法可以做到这一点：

- 检查第一个和最后一个数据包IP标识(ID)值。
- 检查第一个和最后一个数据包的时间戳值。

在本例中，您可以看到每个捕获的第一个数据包具有相同的IP ID值：



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	74	0xb0a3d (2612)	54494 → 2388 [SYN] Seq=1884231611 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=3577289526 TSecr=0 WS=128
2	2019-10-16 16:13:45.195958	192.168.2.220	192.168.1.220	TCP	74	0xb0a35 (2613)	[TCP Retransmission] 54494 → 2388 [SYN] Seq=1884231611 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=3577289526 TSecr=0 WS=128
3	2019-10-16 16:13:47.177542	192.168.1.220	192.168.2.220	TCP	66	0xb0a3f (2615)	2388 → 54494 [SYN, ACK] Seq=2669989678 Ack=1884231612 Win=8192 Len=0 MSS=256 SACK_PERM=1 TSval=4264384 TSecr=3577288500
4	2019-10-16 16:13:47.178030	192.168.2.220	192.168.1.220	TCP	66	0xb0a36 (2614)	54494 → 2388 [ACK] Seq=1884231612 Ack=2669989678 Win=29312 Len=0 TSval=3572291508 TSecr=4264384
5	2019-10-16 16:13:47.179647	192.168.1.220	192.168.2.220	TCP	1314	0x1521 (5409)	2388 → 54494 [ACK] Seq=2669989678 Ack=1884231612 Win=8192 Len=0 MSS=256 SACK_PERM=1 TSval=4264384 TSecr=3577288500
6	2019-10-16 16:13:47.179998	192.168.2.220	192.168.1.220	TCP	66	0xb0a37 (2615)	54494 → 2388 [ACK] Seq=2669989678 Ack=1884231612 Win=8192 Len=0 MSS=256 SACK_PERM=1 TSval=4264384 TSecr=3577288500
7	2019-10-16 16:13:47.180456	192.168.1.220	192.168.2.220	TCP	1314	0x1522 (5411)	2388 → 54494 [ACK] Seq=2669989678 Ack=1884231612 Win=8192 Len=0 MSS=256 SACK_PERM=1 TSval=4264384 TSecr=3577288500
8	2019-10-16 16:13:47.180951	192.168.1.220	192.168.2.220	TCP	1314	0x1524 (5412)	2388 → 54494 [ACK] Seq=2669989678 Ack=1884231612 Win=8192 Len=0 MSS=256 SACK_PERM=1 TSval=4264384 TSecr=3577288500
9	2019-10-16 16:13:47.180715	192.168.2.220	192.168.1.220	TCP	78	0xb0a38 (2616)	54494 → 2388 [ACK] Seq=2669989678 Ack=1884231612 Win=8192 Len=0 MSS=256 SACK_PERM=1 TSval=4264384 TSecr=3577288500
10	2019-10-16 16:13:47.180792	192.168.2.220	192.168.1.220	TCP	78	0xb0a39 (2617)	54494 → 2388 [ACK] Seq=2669989678 Ack=1884231612 Win=8192 Len=0 MSS=256 SACK_PERM=1 TSval=4264384 TSecr=3577288500
11	2019-10-16 16:13:47.489888	192.168.1.220	192.168.2.220	TCP	1314	0x1525 (5413)	2388 → 54494 [ACK] Seq=2669989678 Ack=1884231612 Win=8192 Len=0 MSS=256 SACK_PERM=1 TSval=4264384 TSecr=3577288500
12	2019-10-16 16:13:47.490376	192.168.2.220	192.168.1.220	TCP	66	0xb0a3a (2618)	54494 → 2388 [ACK] Seq=2669989678 Ack=1884231612 Win=8192 Len=0 MSS=256 SACK_PERM=1 TSval=4264384 TSecr=3577288500
13	2019-10-16 16:13:47.490865	192.168.1.220	192.168.2.220	TCP	1314	0x1526 (5414)	2388 → 54494 [ACK] Seq=2669989678 Ack=1884231612 Win=8192 Len=0 MSS=256 SACK_PERM=1 TSval=4264384 TSecr=3577288500
14	2019-10-16 16:13:47.490917	192.168.2.220	192.168.1.220	TCP	1314	0x1529 (5415)	54494 → 2388 [ACK] Seq=2669989678 Ack=1884231612 Win=8192 Len=0 MSS=256 SACK_PERM=1 TSval=4264384 TSecr=3577288500
15	2019-10-16 16:13:47.490987	192.168.1.220	192.168.2.220	TCP	1314	0x1529 (5415)	2388 → 54494 [ACK] Seq=2669989678 Ack=1884231612 Win=8192 Len=0 MSS=256 SACK_PERM=1 TSval=4264384 TSecr=3577288500
16	2019-10-16 16:13:47.491231	192.168.2.220	192.168.1.220	TCP	66	0xb0a3b (2619)	54494 → 2388 [ACK] Seq=2669989678 Ack=1884231612 Win=8192 Len=0 MSS=256 SACK_PERM=1 TSval=4264384 TSecr=3577288500
17	2019-10-16 16:13:47.491261	192.168.2.220	192.168.1.220	TCP	78	0xb0a3c (2620)	54494 → 2388 [ACK] Seq=2669989678 Ack=1884231612 Win=8192 Len=0 MSS=256 SACK_PERM=1 TSval=4264384 TSecr=3577288500
18	2019-10-16 16:13:47.491765	192.168.1.220	192.168.2.220	TCP	1314	0x152a (5418)	2388 → 54494 [ACK] Seq=2669989678 Ack=1884231612 Win=8192 Len=0 MSS=256 SACK_PERM=1 TSval=4264384 TSecr=3577288500
19	2019-10-16 16:13:47.492024	192.168.2.220	192.168.1.220	TCP	78	0xb0a3d (2621)	54494 → 2388 [ACK] Seq=2669989678 Ack=1884231612 Win=8192 Len=0 MSS=256 SACK_PERM=1 TSval=4264384 TSecr=3577288500
20	2019-10-16 16:13:48.410150	192.168.1.220	192.168.2.220	TCP	1314	0x152e (5422)	2388 → 54494 [ACK] Seq=2669989678 Ack=1884231612 Win=8192 Len=0 MSS=256 SACK_PERM=1 TSval=4264384 TSecr=3577288500
21	2019-10-16 16:13:48.411050	192.168.2.220	192.168.1.220	TCP	66	0xb0a3e (2622)	54494 → 2388 [ACK] Seq=2669989678 Ack=1884231612 Win=8192 Len=0 MSS=256 SACK_PERM=1 TSval=4264384 TSecr=3577288500
22	2019-10-16 16:13:48.411569	192.168.1.220	192.168.2.220	TCP	1314	0x152f (5423)	2388 → 54494 [ACK] Seq=2669989678 Ack=1884231612 Win=8192 Len=0 MSS=256 SACK_PERM=1 TSval=4264384 TSecr=3577288500
23	2019-10-16 16:13:48.411630	192.168.2.220	192.168.1.220	TCP	1314	0x1530 (5424)	54494 → 2388 [ACK] Seq=2669989678 Ack=1884231612 Win=8192 Len=0 MSS=256 SACK_PERM=1 TSval=4264384 TSecr=3577288500
24	2019-10-16 16:13:48.411660	192.168.1.220	192.168.2.220	TCP	1314	0x1533 (5427)	2388 → 54494 [ACK] Seq=2669989678 Ack=1884231612 Win=8192 Len=0 MSS=256 SACK_PERM=1 TSval=4264384 TSecr=3577288500
25	2019-10-16 16:13:48.411660	192.168.1.220	192.168.2.220	TCP	1314	0x1533 (5427)	2388 → 54494 [ACK] Seq=2669989678 Ack=1884231612 Win=8192 Len=0 MSS=256 SACK_PERM=1 TSval=4264384 TSecr=3577288500
26	2019-10-16 16:13:48.411859	192.168.2.220	192.168.1.220	TCP	66	0xb0a3f (2623)	54494 → 2388 [ACK] Seq=2669989678 Ack=1884231612 Win=8192 Len=0 MSS=256 SACK_PERM=1 TSval=4264384 TSecr=3577288500
27	2019-10-16 16:13:48.412088	192.168.2.220	192.168.1.220	TCP	66	0xb0a40 (2624)	54494 → 2388 [ACK] Seq=2669989678 Ack=1884231612 Win=8192 Len=0 MSS=256 SACK_PERM=1 TSval=4264384 TSecr=3577288500

如果它们不同，那么：

1. 比较每个捕获的第一个数据包的时间戳。
2. 从具有最新Timestamp的捕获中获取过滤器，将Timestamp过滤器从==更改为>= ( 第一个数据包 ) 和<= ( 最后一个数据包 ) 更改，例如：

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 16:13:43.245638000 Central European Daylight Time

[Time shift for this packet: 0.000000000 seconds]

Epoch Time: 1571235223.245638000 seconds

[Time delta from previous captured frame: 0.000000000 seconds]

[Time delta from previous displayed frame: 0.000000000 seconds]

[Time since reference or first frame: 0.000000000 seconds]

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")

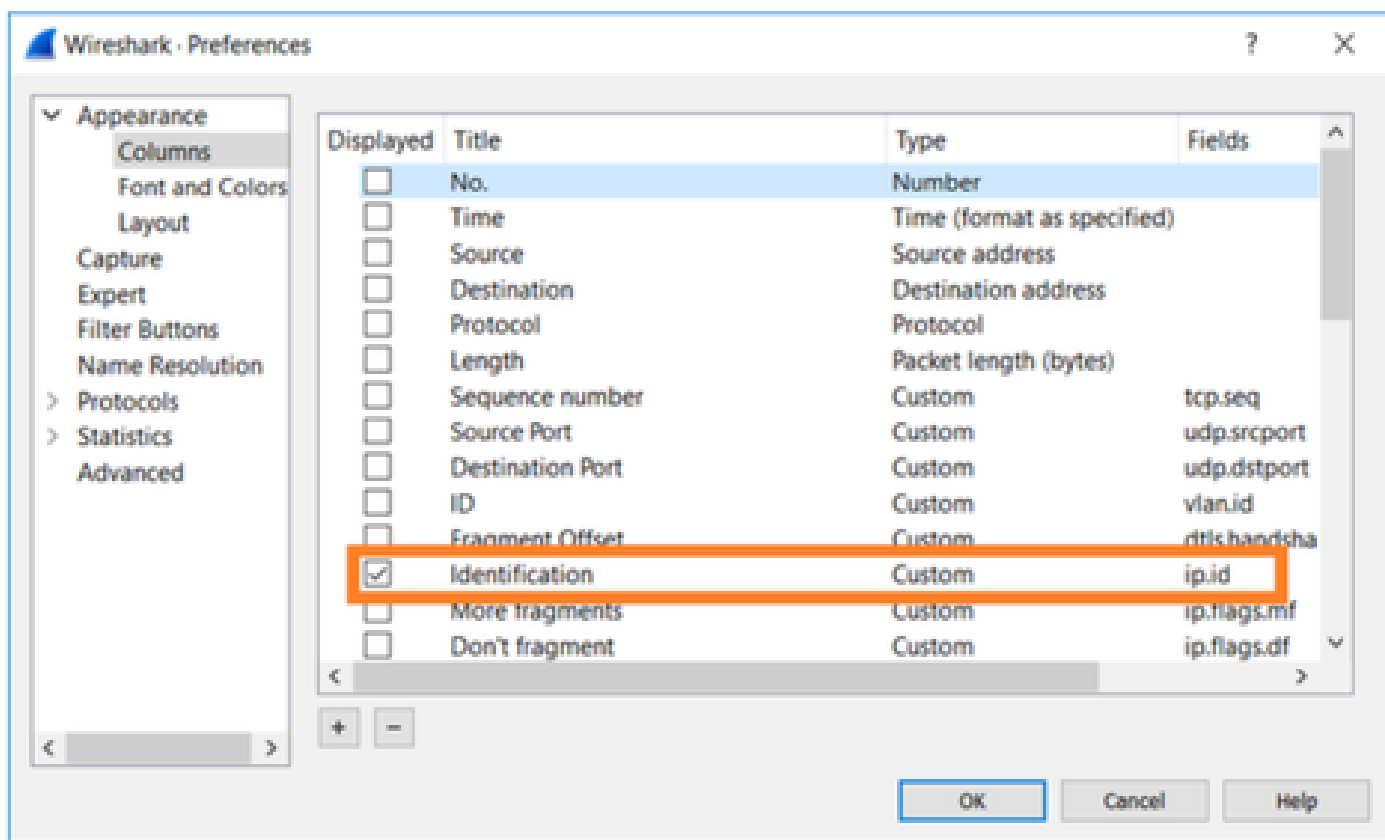
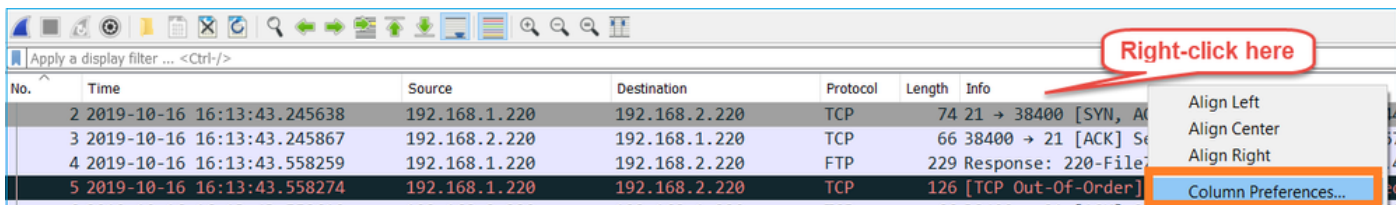
3. 将指定数据包导出到新捕获，选择文件>导出指定数据包，然后保存显示的数据包。此时，两个捕获都必须包含覆盖同一时间窗口的数据包。现在您可以开始比较2个捕获。

第二步：指定用于比较2个捕获的数据包字段。可以使用的字段示例：

- IP标识
- RTP序列号
- ICMP序列号

创建每个捕获的文本版本，其中包含您在第1步中指定的每个数据包的字段。为此，请仅保留感兴趣的列，例如，如果要根据IP标识比较数据包，请修改捕获，如图所示。

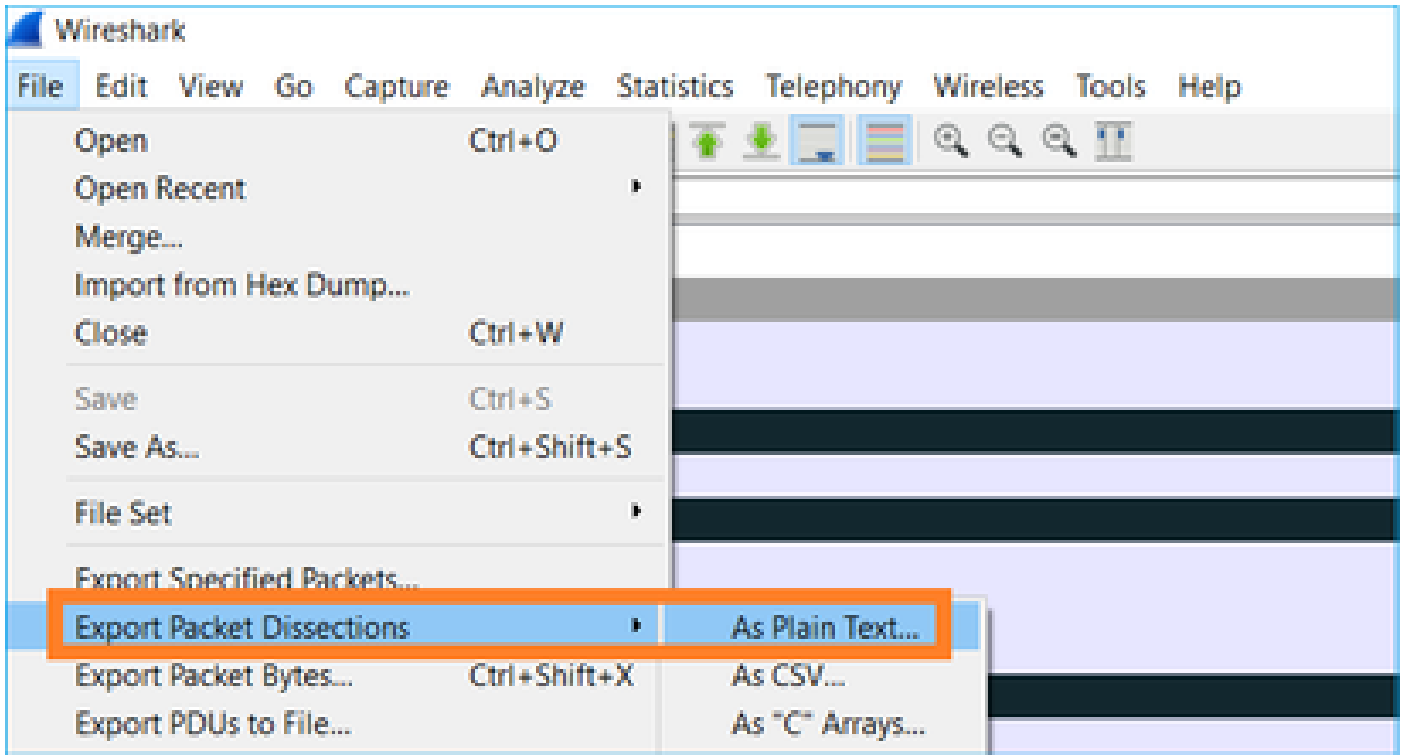




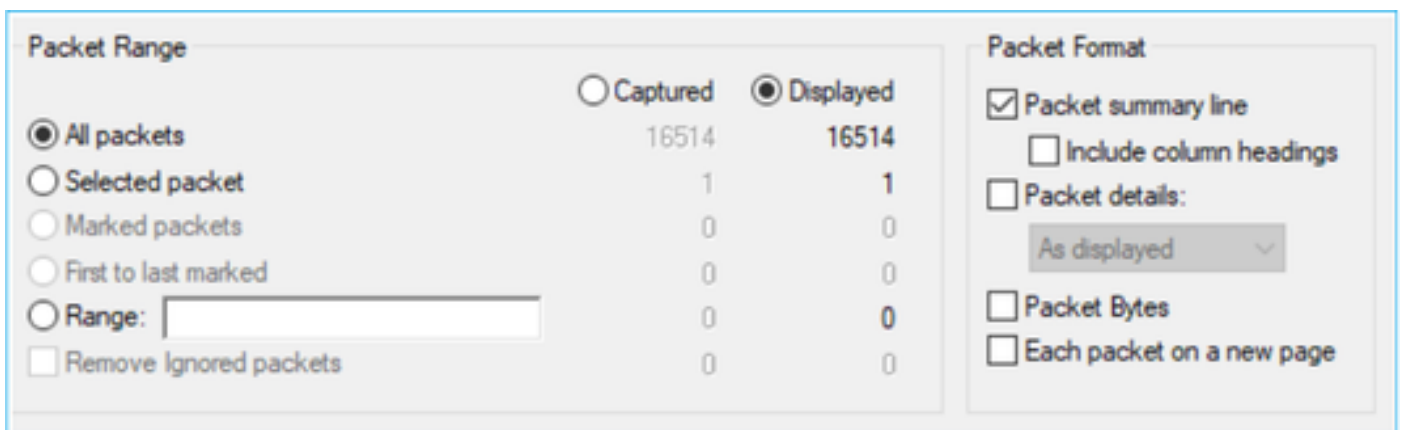
结果：

Identification
0x150e (5398)
0xfdb0 (64944)
0x1512 (5394)
<b>0x1510 (5392)</b>
0xfdb1 (64945)
<b>0xfdb2 (64946)</b>
0xfdb3 (64947)
0x1513 (5395)
0xfdb4 (64948)
<b>0xfdb5 (64949)</b>
0x1516 (5398)
<b>0x1515 (5397)</b>
0xfdb6 (64950)
0x1517 (5399)
0xfdb7 (64951)
0x1518 (5400)
0xfdb8 (64952)
<b>0xfdb9 (64953)</b>
0x151b (5403)
<b>0x151a (5402)</b>
0xfdba (64954)
0x151c (5404)
0xfdbb (64955)
0x151d (5405)
0x0a34 (2612)
0xfdbc (64956)
<b>0x0a35 (2613)</b>
0x151f (5407)
0x0a36 (2614)
<ul style="list-style-type: none"> <li>▼ Frame 23988: 66 bytes on wire (528 bits), 66 bytes captured (528 bits)</li> <li style="padding-left: 20px;">Encapsulation type: Ethernet (1)</li> <li style="padding-left: 20px;">Arrival Time: Oct 16, 2019 16:20:21.785130000 Central European Daylight Time</li> </ul>

第三步：创建捕获的文本版本(File > Export Packet Dissections > As Plain Text...)，如图所示：



取消选中Include column headings和Packet details选项以仅导出所显示字段的值，如图所示：



第四步：对文件中的数据包进行排序。可以使用Linux sort 命令来完成此操作：

```
<#root>
```

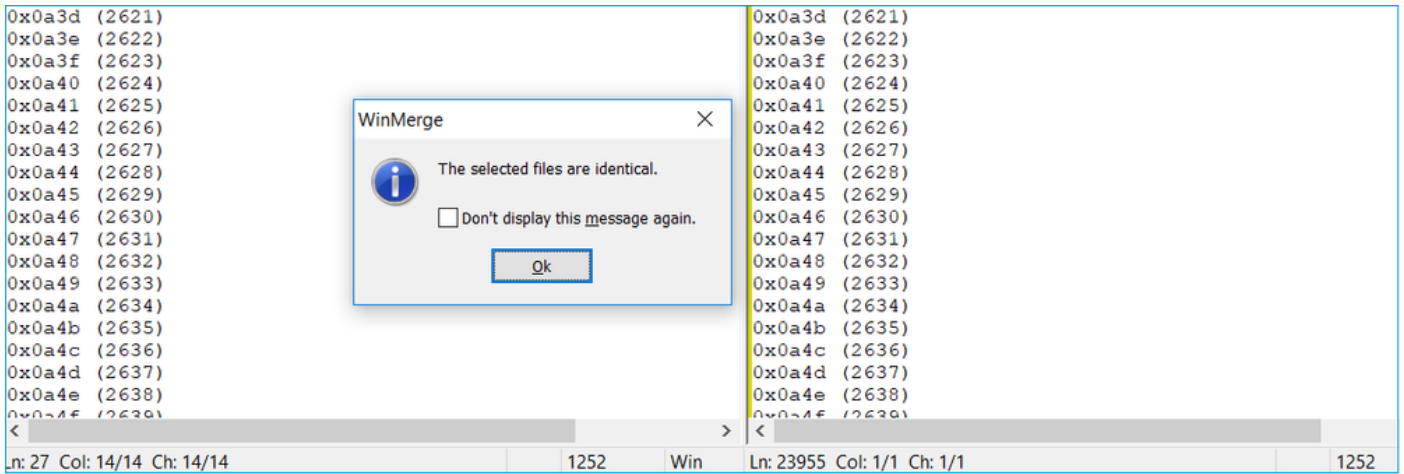
```
#
```

```
sort CAPI_IDs > file1.sorted
```

```
#
```

```
sort CAPO_IDs > file2.sorted
```

第五步：使用文本比较工具（例如，WinMerge）或Linux diff 命令查找两次捕获之间的差异。

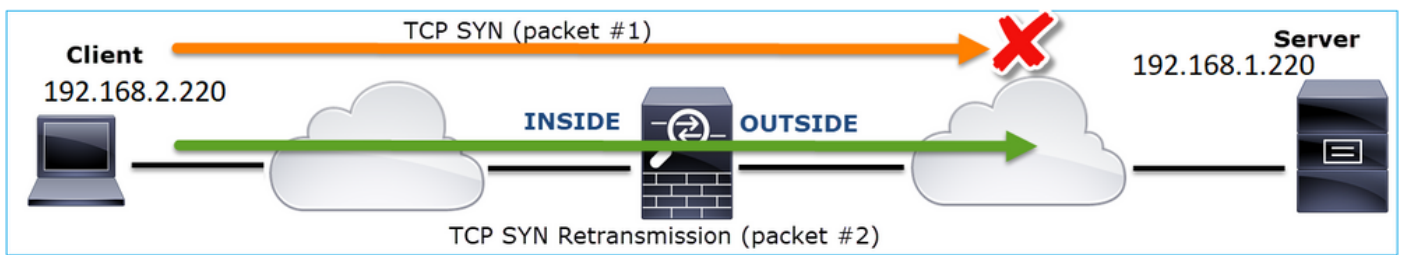


在这种情况下，FTP数据流量的CAPI和CAPO捕获相同。这证明数据包丢失不是防火墙导致的。确定上行/下行数据包丢失。

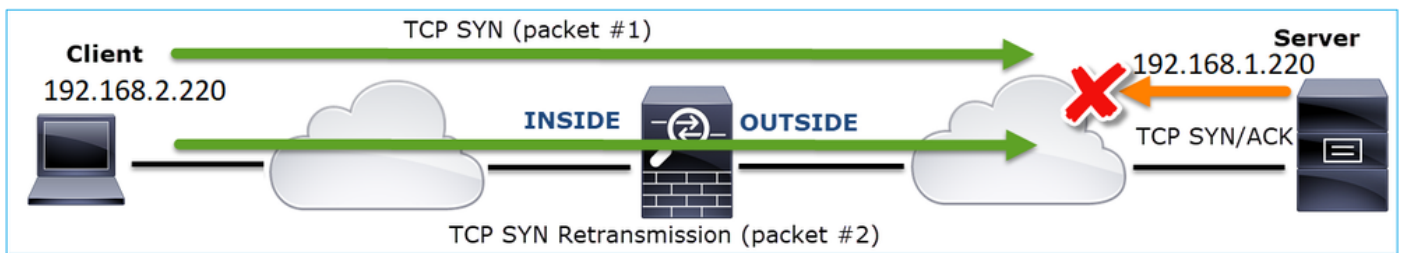
No.	Time	Source	Destination	Protocol	Length	Info
1	2019-10-16 16:13:44.169516	192.168.2.220	192.168.1.220	TCP	74	54494 → 2388 [SYN] Seq=2157030681 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=3577288500 TSecr=0 WS=1
2	2019-10-16 16:13:45.196050	192.168.2.220	192.168.1.220	TCP	74	[TCP Retransmission] 54494 → 2388 [SYN] Seq=2157030681 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=3577288500 TSecr=0 WS=1
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=2157030682 Win=8192 Len=0 MSS=1260 WS=256 SACK_PERM=1 TSv=3577291508 TSecr=4264384
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 Win=29312 Len=0 TSval=3577291508 TSecr=4264384
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 Win=66048 Len=1248 TSval=4264384 TSecr=3577291508
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 Win=32128 Len=0 TSval=3577291510 TSecr=4264384
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 [ACK] Seq=2224319408 Ack=2157030682 Win=66048 Len=1248 TSval=4264384 TSecr=3577291510
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 Win=66048 Len=1248 TSval=4264384 TSecr=3577291510
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=2157030682 Ack=2224318160 Win=35072 Len=0 TSval=3577291510 TSecr=4264384
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=2157030682 Ack=2224318160 Win=37888 Len=0 TSval=3577291510 TSecr=4264384
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=2224318160 Ack=2157030682 Win=66048 Len=1248 TSval=4264415 TSecr=3577291820
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 Win=40832 Len=0 TSval=3577291820 TSecr=4264415
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 Win=66048 Len=1248 TSval=4264415 TSecr=3577291820
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 [ACK] Seq=2224324400 Ack=2157030682 Win=66048 Len=1248 TSval=4264415 TSecr=3577291820
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 Win=66048 Len=1248 TSval=4264415 TSecr=3577291820
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 Win=43776 Len=0 TSval=3577291821 TSecr=4264415

要点:

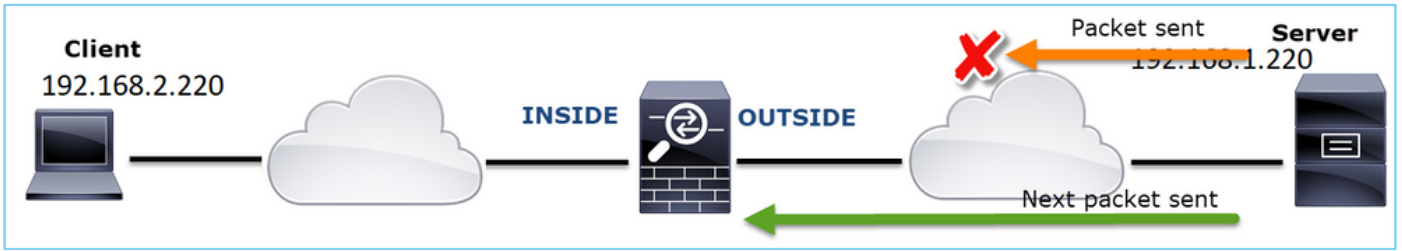
1. 此数据包是TCP重传。具体而言，它是从客户端发送到服务器的TCP SYN数据包，用于被动模式下的FTP数据。由于客户端重新发送数据包，您可以看到初始SYN(数据包#1)数据包在防火墙的上游丢失。



在这种情况下，有可能是SYN数据包发送到服务器，但SYN/ACK数据包在返回途中丢失：



2. 服务器发出一个数据包，Wireshark确定未看到/捕获上一个数据段。由于未捕获的数据包从服务器发送到客户端，并且在防火墙捕获中看不到，这意味着数据包在服务器和防火墙之间丢失。



这表示FTP服务器和防火墙之间存在数据包丢失。

行动2.进行其他捕获。

在终端处执行额外捕获和捕获。尝试应用分治法，进一步隔离导致数据包丢失的问题数据段。

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) (RETR file15mb)
156	2019-10-16 16:13:51.749860	192.168.1.220	192.168.2.220	FTP-DA..	1314	FTP Data: 1248 bytes (PASV) (RETR file15mb)
157	2019-10-16 16:13:51.749872	192.168.1.220	192.168.2.220	FTP-DA..	1314	FTP Data: 1248 bytes (PASV) (RETR file15mb)
158	2019-10-16 16:13:51.750722	192.168.2.220	192.168.1.220	TCP	66	54494 → 2388 [ACK] Seq=2157030682 Ack=2224385552 Win=180480 Len=0 TSv
159	2019-10-16 16:13:51.750744	192.168.1.220	192.168.2.220	FTP-DA..	1314	FTP Data: 1248 bytes (PASV) (RETR file15mb)
160	2019-10-16 16:13:51.750768	192.168.2.220	192.168.1.220	TCP	66	54494 → 2388 [ACK] Seq=2157030682 Ack=2224386800 Win=183424 Len=0 TSv
161	2019-10-16 16:13:51.750782	192.168.1.220	192.168.2.220	FTP-DA..	1314	FTP Data: 1248 bytes (PASV) (RETR file15mb)
162	2019-10-16 16:13:51.751001	192.168.2.220	192.168.1.220	TCP	70	[TCP Dup ACK 160#1] 54494 → 2388 [ACK] Seq=2157030682 Ack=2224386800
163	2019-10-16 16:13:51.751024	192.168.1.220	192.168.2.220	FTP-DA..	1314	FTP Data: 1248 bytes (PASV) (RETR file15mb)
164	2019-10-16 16:13:51.751378	192.168.2.220	192.168.1.220	TCP	70	[TCP Dup ACK 160#2] 54494 → 2388 [ACK] Seq=2157030682 Ack=2224386800
165	2019-10-16 16:13:51.751402	192.168.1.220	192.168.2.220	FTP-DA..	1314	FTP Data: 1248 bytes (PASV) (RETR file15mb)
166	2019-10-16 16:13:51.751622	192.168.2.220	192.168.1.220	TCP	70	[TCP Dup ACK 160#3] 54494 → 2388 [ACK] Seq=2157030682 Ack=2224386800
167	2019-10-16 16:13:51.751648	192.168.1.220	192.168.2.220	FTP-DA..	1314	[TCP Fast Retransmission] FTP Data: 1248 bytes (PASV) (RETR file15mb)

```

> 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)
  
```

要点:

1. 接收方（本例中为FTP客户端）跟踪传入的TCP序列号。如果它检测到数据包丢失（已跳过预期序列号），则生成ACK数据包，其ACK为“已跳过预期序列号”。在本示例中，Ack=2224386800。
2. Dup ACK触发TCP快速重新传输（收到重复的ACK后，在20毫秒内重新传输）。

重复ACK是什么意思？

- 有几个ACK重复，但实际没有重新传输，这表明到达的数据包更有可能顺序混乱。
- 重复ACK和实际重新传输表明存在一定程度的数据包丢失。

行动3.计算传输数据包的防火墙处理时间。

在两个不同的接口上应用相同的捕获：

<#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
```

导出捕获检查入口数据包与出口数据包之间的时间差异

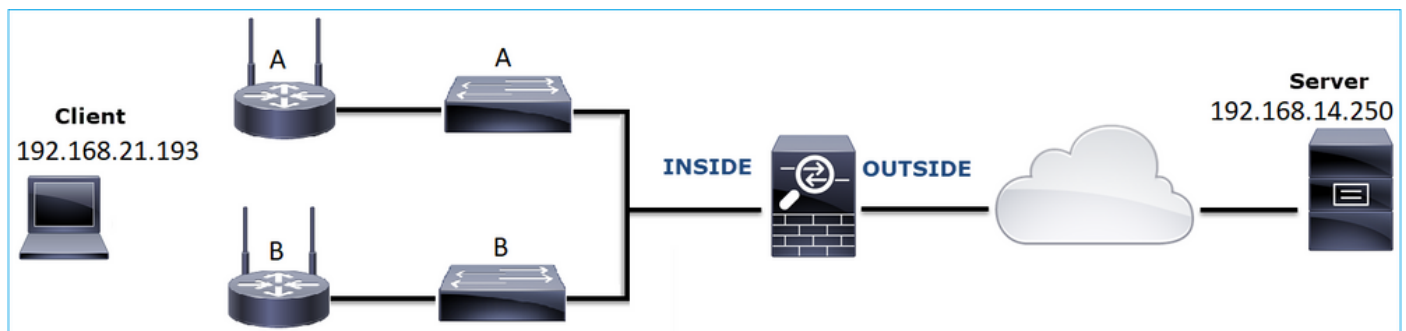
## 案例 7.TCP连接问题 ( 数据包损坏 )

问题说明:

无线客户端(192.168.21.193)尝试连接到目标服务器(192.168.14.250 - HTTP)，但有2种不同的场景：

- 当客户端连接到接入点(AP)“A”时，HTTP连接不起作用。
- 当客户端连接到接入点(AP)“B”时，HTTP连接会正常工作。

下图显示拓扑：



受影响的流：

源IP：192.168.21.193

DST IP：192.168.14.250

协议：TCP 80

捕获分析

在FTD LINA引擎上启用捕获：

```
<#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

捕获-功能场景：

作为基准，使用已知良好场景中的捕获信息总是非常有用。

下图显示了在NGFW内部接口上捕获的流量

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=1460 SACK_PERM=1
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 Win=64240 Len=0 MSS=1380 SACK_PERM=1
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=65535 Len=0
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=63928 Len=1380 [TCP segment of a reassembled PDU]
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=65535 Len=0
8	2013-08-08 17:03:34.391749	192.168.21.193	192.168.14.250	HTTP	369	GET /ttest.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=65007 Len=0
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

下图显示了在NGFW外部接口上捕获的流量。

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=1380 SACK_PERM=1
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 Win=64240 Len=0 MSS=1460 SACK_PERM=1
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=65535 Len=0
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=63928 Len=1380 [TCP segment of a reassembled PDU]
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=65535 Len=0
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=65007 Len=0
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

要点：

1. 2个捕获几乎相同（考虑ISN随机化）。
2. 没有数据包丢失的迹象。
3. 无乱序(OOO)数据包
4. 有3个HTTP GET请求。第一个收到404“Not Found”，第二个收到200“OK”，第三个收到304“Not Modified”重定向消息。

捕获-已知故障场景：

入口捕获(CAPI)内容。

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=1460 SACK_PERM=1
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 Win=64240 Len=0 MSS=1380 SACK_PERM=1
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=65535 Len=2[Malformed Packet]
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=4231766829 Ack=867575960 Win=65535 Len=311
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=867576125 Ack=4231767140 Win=63929 Len=0
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=867575960 Ack=4231767140 Win=63929 Len=164
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=65371 Len=2
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=4231767140 Ack=867576125 Win=65371 Len=2
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=867576125 Ack=4231767143 Win=63927 Len=0
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=1460 SACK_PERM=1
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 Win=64240 Len=0 MSS=1380 SACK_PERM=1
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=65535 Len=2[Malformed Packet]
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=2130836821 Ack=2991287217 Win=65535 Len=313
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=65371 Len=2
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=2991287382 Ack=2130837136 Win=63925 Len=0
18	2013-08-08 15:33:35.126846	192.168.21.193	192.168.14.250	TCP	60	[TCP Spurious Retransmission] 3073 → 80 [FIN, ACK] Seq=2130837134 Ack=2991287382 Win=65371 Len=2
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=2991287382 Ack=2130837137 Win=63925 Len=0

要点：

1. 存在TCP三次握手。
2. 有TCP重新传输和数据包丢失指示。
3. Wireshark发现一个数据包(TCP ACK)存在格式错误。

下图显示了出口捕获(CAPO)内容。

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=1380 SACK_PERM=1
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 Win=64240 Len=0 MSS=1460 SACK_PERM=1
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=65535 Len=2 [Malformed Packet]
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 15:33:31.988357	192.168.21.193	192.168.14.250	TCP	369	[TCP Retransmission] 3072 → 80 [PSH, ACK] Seq=230342489 Ack=268013987 Win=65535 Len=311
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=268014152 Ack=230342800 Win=63929 Len=0
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] Seq=268013987 Ack=230342800 Win=63929 Len=164
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=65371 Len=2
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=230342800 Ack=268014152 Win=65371 Len=2
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=268014152 Ack=230342803 Win=63927 Len=0
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 MSS=1380 SACK_PERM=1
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=2731219423 Win=64240 Len=0 MSS=1460 SACK_PERM=1
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=65535 Len=2 [Malformed Packet]
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=2731219423 Ack=2453407926 Win=65535 Len=313
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=65371 Len=2
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=2453408091 Ack=2731219738 Win=63925 Len=0
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] Seq=2731219736 Ack=2453408091 Win=65371 Len=2
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=2453408091 Ack=2731219739 Win=63925 Len=0

要点:

2个捕获几乎相同 (考虑ISN随机化) :

1. 存在TCP三次握手。
2. 有TCP重新传输和数据包丢失指示。
3. Wireshark发现一个数据包(TCP ACK)存在格式错误。

检查格式错误的数据包 :

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=1460 SACK_PERM=1
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 Win=64240 Len=0 MSS=1380 SACK_PERM=1
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=65535 Len=2 [Malformed Packet]

```

> Frame 3: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0
> 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
> 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)
  [Malformed Packet: Tunnel Socket]
  [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 a7 7d 33 b6  ....P:;..3.
0030 28 98 50 10 ff ff 01 bf 00 00 00 00 00 00 00  (-P.....)

```

要点:

1. 数据包被识别为Wireshark的格式错误。
2. 长度为2个字节。

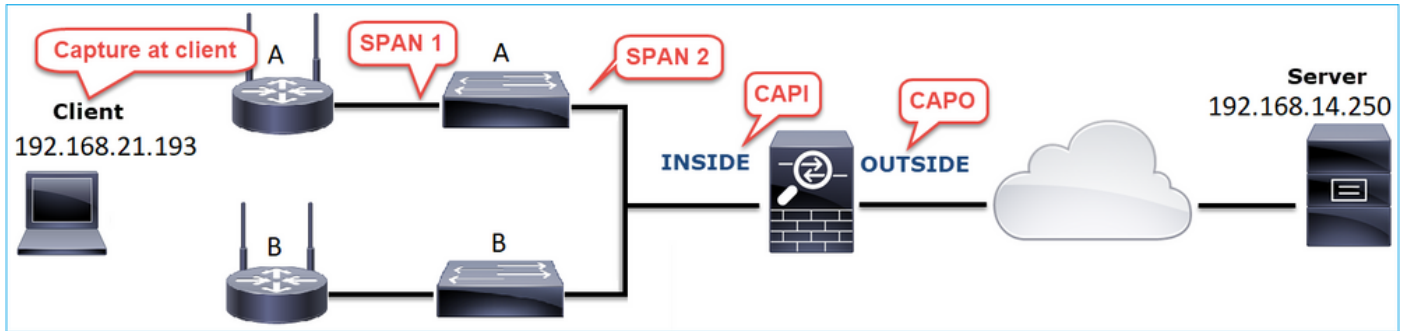


- 3. 有2个字节的TCP负载。
- 4. 负载是4个额外的零(00 00)。

### 推荐的操作

本部分列出的操作旨在进一步缩小问题范围。

行动1.获取其他捕获。在终端包括捕获，如果可能，请尝试应用分治法隔离数据包损坏的来源，例如：

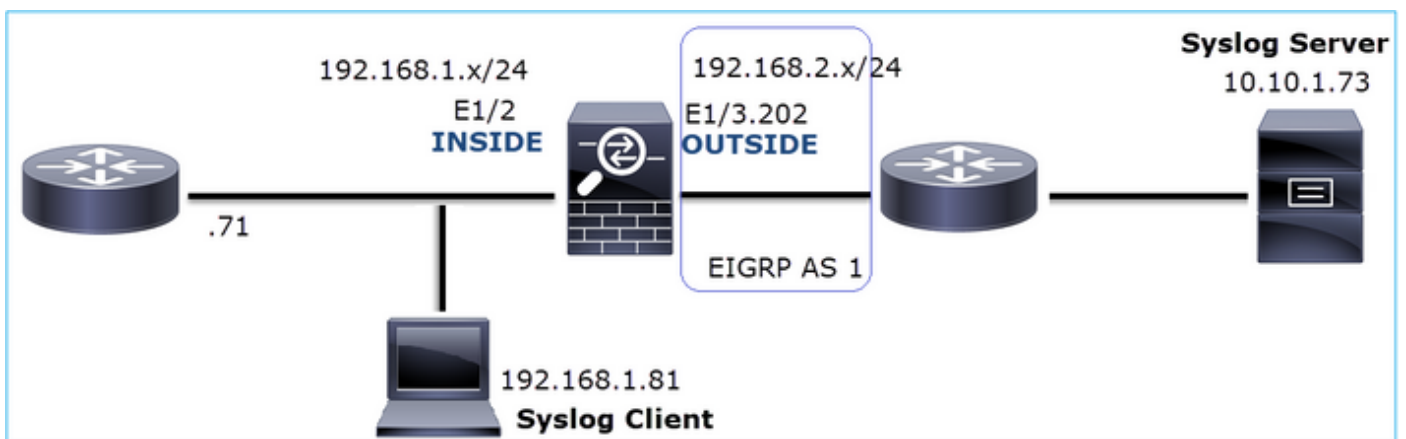


在这种情况下，交换机“A”接口驱动程序添加了2个额外字节，解决方案是更换导致损坏的交换机。

### 案例 8.UDP连接问题 ( 缺少数据包 )

问题说明：在目标Syslog服务器上看不到Syslog (UDP 514)消息。

下图显示拓扑：



受影响的流：

源IP：192.168.1.81

DST IP：10.10.1.73

协议：UDP 514

## 捕获分析

在FTD LINA引擎上启用捕获：

```
<#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捕获显示无数据包：

```
<#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
```

## 推荐的操作

本部分列出的操作旨在进一步缩小问题范围。

行动1.检查FTD连接表。

要检查特定连接，可以使用此语法：

```
<#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
```

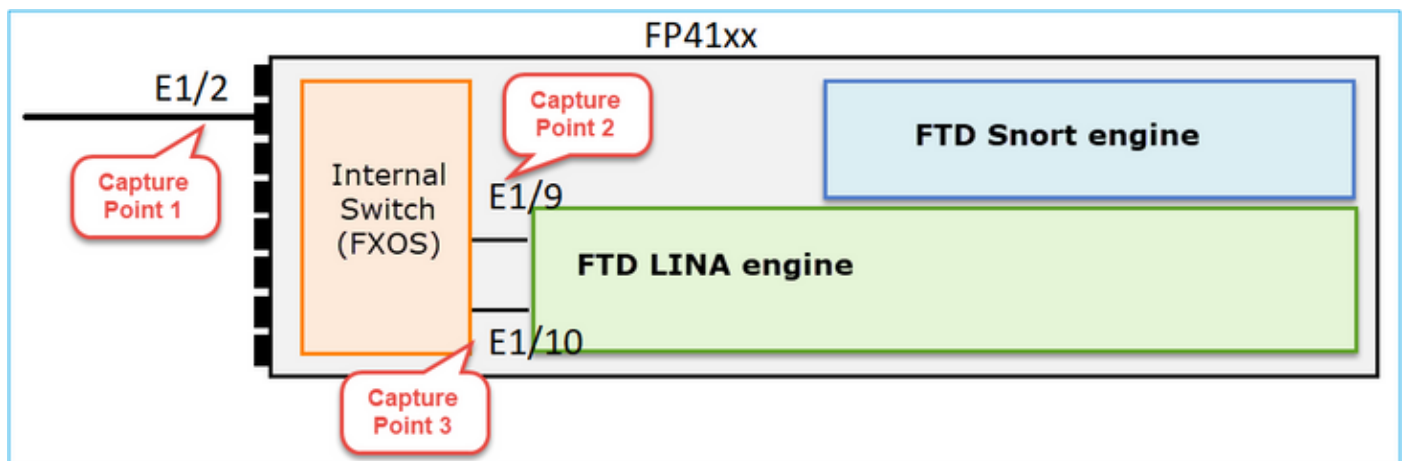
```
192.168.1.81:514, idle 0:00:00, bytes
480379697
, flags -
o
N1
```

### 要点:

1. 入口和出口接口相同(U-turn)。
2. 字节数具有非常大的值 ( 约5 GB ) 。
3. 标志“o”表示流量分流 ( HW加速流量 ) 。这就是为什么FTD捕获不显示任何数据包。仅在41xx和93xx平台上支持流量分流。在本例中，设备是41xx。

### 行动2.获取机箱级别捕获。

连接到Firepower机箱管理器，在入口接口（本例中为E1/2）和背板接口（E1/9和E1/10）上启用捕获，如图所示：



Overview Interfaces Logical Devices Security Engine Platform Settings System Tools Help admin

Select an instance: mzafeiro\_FTD

**mzafeiro\_FTD**

Ethernet1/2

Ethernet1/3

Ethernet1/1

**FTD**  
Ethernet1/9, Ethernet1/10

Session Name\* CAPI

Selected Interfaces Ethernet1/2

Buffer Size 256 MB

Snap length: 1518 Bytes

Store Packets

Capture On All Backplane Ports

Capture Filter

几秒钟后：

Capture Session Filter List

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

提示：在Wireshark中排除VN标记的数据包，以消除物理接口级别的数据包重复

攻击前：

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

在:

CAPI-ethernet-1-2-0.pcap

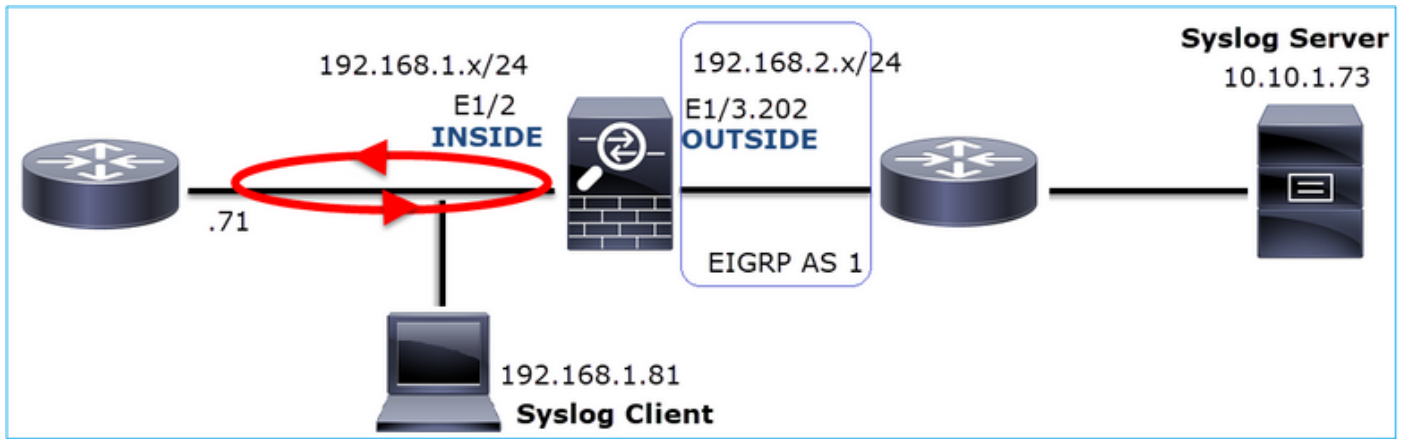
File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

syslog && !mtag

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: Teardown local-host identity:192.168.1.81 dur
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: Teardown local-host identity:192.168.1.81 dur
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: Teardown local-host identity:192.168.1.81 dur
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: Built local-host NET_FIREWALL:192.168.1.71\n
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: Teardown local-host identity:192.168.1.81 dur
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: Built local-host NET_FIREWALL:192.168.1.71\n
1346	0.000002758	192.168.1.81	10.10.1.73	Syslog	147		251 LOCAL4.DEBUG: Oct 15 2019 07:47:17: %ASA-7-609002: Teardown local-host identity:192.168.1.81 dur
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: Built local-host NET_FIREWALL:192.168.1.71\n
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: Teardown local-host identity:192.168.1.81 dur
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: Built inbound ICMP connection for faddr 192.16
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: Built local-host NET_FIREWALL:192.168.1.71\n
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: Teardown local-host identity:192.168.1.81 dur
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: Teardown ICMP connection for faddr 192.168.1.7
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: Teardown local-host NET_FIREWALL:192.168.1.71
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: Built inbound ICMP connection for faddr 192.16
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: Built local-host NET_FIREWALL:192.168.1.71\n
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: Teardown local-host identity:192.168.1.81 dur
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: Teardown ICMP connection for faddr 192.168.1.7
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: Teardown local-host NET_FIREWALL:192.168.1.71
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: Built inbound ICMP connection for faddr 192.16
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: Built local-host NET_FIREWALL:192.168.1.71\n
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: Teardown local-host identity:192.168.1.81 dur
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: Teardown ICMP connection for faddr 192.168.1.7
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: Teardown local-host NET_FIREWALL:192.168.1.71
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: Built inbound ICMP connection for faddr 192.16
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: Built local-host NET_FIREWALL:192.168.1.71\n

要点:

1. 系统会应用显示过滤器来删除重复的数据包并仅显示syslog。
2. 数据包之间的差异处于微秒级。这表示数据包速率非常高。
3. 生存时间(TTL)值持续减小。这表示存在数据包环路。



行动3.使用packet-tracer。

由于数据包不通过防火墙LINA引擎，因此您无法执行实时跟踪（通过跟踪捕获），但可以使用packet-tracer跟踪模拟数据包：

```
<#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

行动4. 确认FTD路由。

检查防火墙路由表以查看是否存在任何路由问题：

<#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

要点:

1. 路由指向正确的出口接口。
2. 路由几分钟前获知(0:02:37)。

行动5.确认连接正常运行时间。

检查连接正常运行时间以查看此连接建立的时间：

<#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

要点：

1. 该连接是在约4分钟前（即路由表中安装EIGRP路由之前）建立的

行动6.清除已建立的连接。



在这种情况下，数据包与已建立的连接匹配，并被路由到错误的出口接口；这将导致环路。这是因为防火墙的操作顺序：

1. 已建立的连接查找（其优先级高于全局路由表查找）。
2. 网络地址转换(NAT)查找- UN-NAT（目标NAT）阶段的优先级高于PBR和路由查找。
3. 基于策略的路由 (PBR)
4. 全局路由表查找

由于连接永不超时（当UDP连接空闲超时为2分钟时，系统日志客户端持续发送数据包），因此需要手动清除连接：

```
<#root>
firepower#
clear conn address 10.10.1.73 address 192.168.1.81 protocol udp port 514
1 connection(s) deleted.
```

验证是否已建立新连接：

```
<#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
```

行动7.配置浮动连接超时。

这是解决此问题并避免次优路由的正确解决方案，对于UDP数据流尤其如此。导航到设备>平台设置>超时，然后设置值：

SMTP Server	H.323	Default	0:05:00	(0:0:0 or 0:0:0 - 1193:0:0)
SNMP	SIP	Default	0:30:00	(0:0:0 or 0:5:0 - 1193:0:0)
SSL	SIP Media	Default	0:02:00	(0:0:0 or 0:1:0 - 1193:0:0)
Syslog	SIP Disconnect:	Default	0:02:00	(0:02:0 or 0:0:1 - 0:10:0)
Timeouts	SIP Invite	Default	0:03:00	(0:1:0 or 0:1:0 - 0:30:0)
Time Synchronization	SIP Provisional Media	Default	0:02:00	(0:2:0 or 0:1:0 - 0:30:0)
UCAPL/CC Compliance	Floating Connection	Custom	0:00:30	(0:0:0 or 0:0:30 - 1193:0:0)
	Xlate-PAT	Default	0:00:30	(0:0:30 or 0:0:30 - 0:5:0)

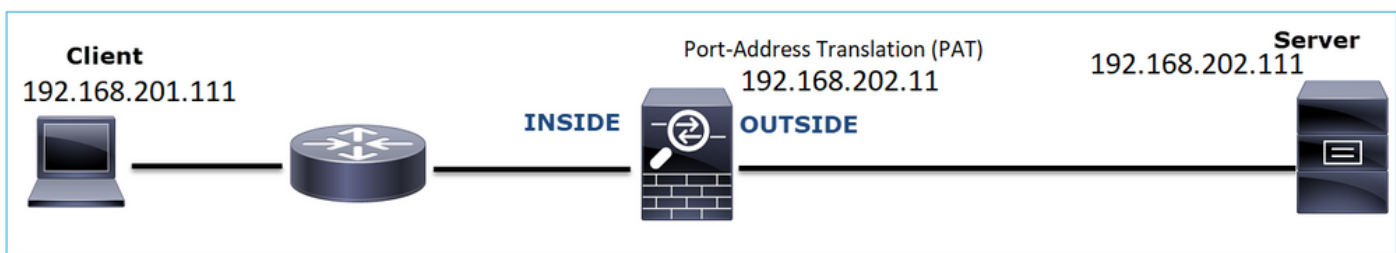
有关浮动连接超时的更多详细信息，请参阅《命令参考》：

<https://www.cisco.com/c/en/us/td/docs/security/asa/asa-cli-reference/T-Z/asa-command-ref-T-Z.html#pgfId-1649892>

### 案例 9.HTTPS连接问题 ( 场景1 )

问题描述：无法建立客户端192.168.201.105和服务器192.168.202.101之间的HTTPS通信

下图显示拓扑：



受影响的流：

源IP：192.168.201.111

目的IP：192.168.202.111

协议：TCP 443 (HTTPS)

### 捕获分析

在FTD LINA引擎上启用捕获：

由于端口地址转换配置，OUTSIDE捕获中使用的IP不同。

<#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

下图显示了在NGFW内部接口上捕获的流量：

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 TSval=192658158 TSecr=0 WS=128
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=1380 SACK_PERM=1 TSval=311915816
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=192658158 TSecr=3119615816
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	0xfcb4 (61364)	443 → 6666 [ACK] Seq=4086514532 Ack=2034865888 Win=8192 Len=0 TSval=3119615816 TSecr=192658174
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

要点:

1. 存在TCP三次握手。
2. SSL协商开始。客户端发送Client Hello消息。
3. 向客户端发送了TCP ACK。
4. 有一个发送到客户端的TCP RST。

下图显示了在NGFW外部接口上捕获的流量。

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=2486930707 Win=29200 Len=0 MSS=1380 SACK_PERM=1 TSval=192658158 TSecr=0 WS=128
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=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=3119615816 TSecr=192658174
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 Len=0 TSval=192658158 TSecr=3119615816
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	0x0905 (47305)	[TCP Retransmission] 15880 → 443 [PSH, ACK] Seq=2486930708 Ack=3674405383 Win=8192 Len=187 TSval=192660190 TSecr=0
38	2018-02-01 10:39:41.297332	192.168.202.11	192.168.202.111	TCP	257	0x884f (34991)	[TCP Retransmission] 15880 → 443 [PSH, ACK] Seq=2486930708 Ack=3674405383 Win=8192 Len=187 TSval=192664224 TSecr=0
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=2486930708 Ack=3674405383 Win=8192 Len=187 TSval=192672244 TSecr=0
40	2018-02-01 10:40:05.317305	192.168.202.11	192.168.202.111	TCP	70	0xd621 (54817)	15880 → 443 [RST] Seq=2486930895 Win=8192 Len=0 TSval=192688266 TSecr=0
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=3674405382 Ack=2486930708 Win=28960 Len=0 MSS=1460 SACK_PERM=1 TSval=3119645908 TSecr=0

要点:

1. 存在TCP三次握手。
2. SSL协商开始。客户端发送Client Hello消息。
3. 防火墙会向服务器发送TCP重新传输。
4. 有一个发送到服务器的TCP RST。

### 推荐的操作

本部分列出的操作旨在进一步缩小问题范围。

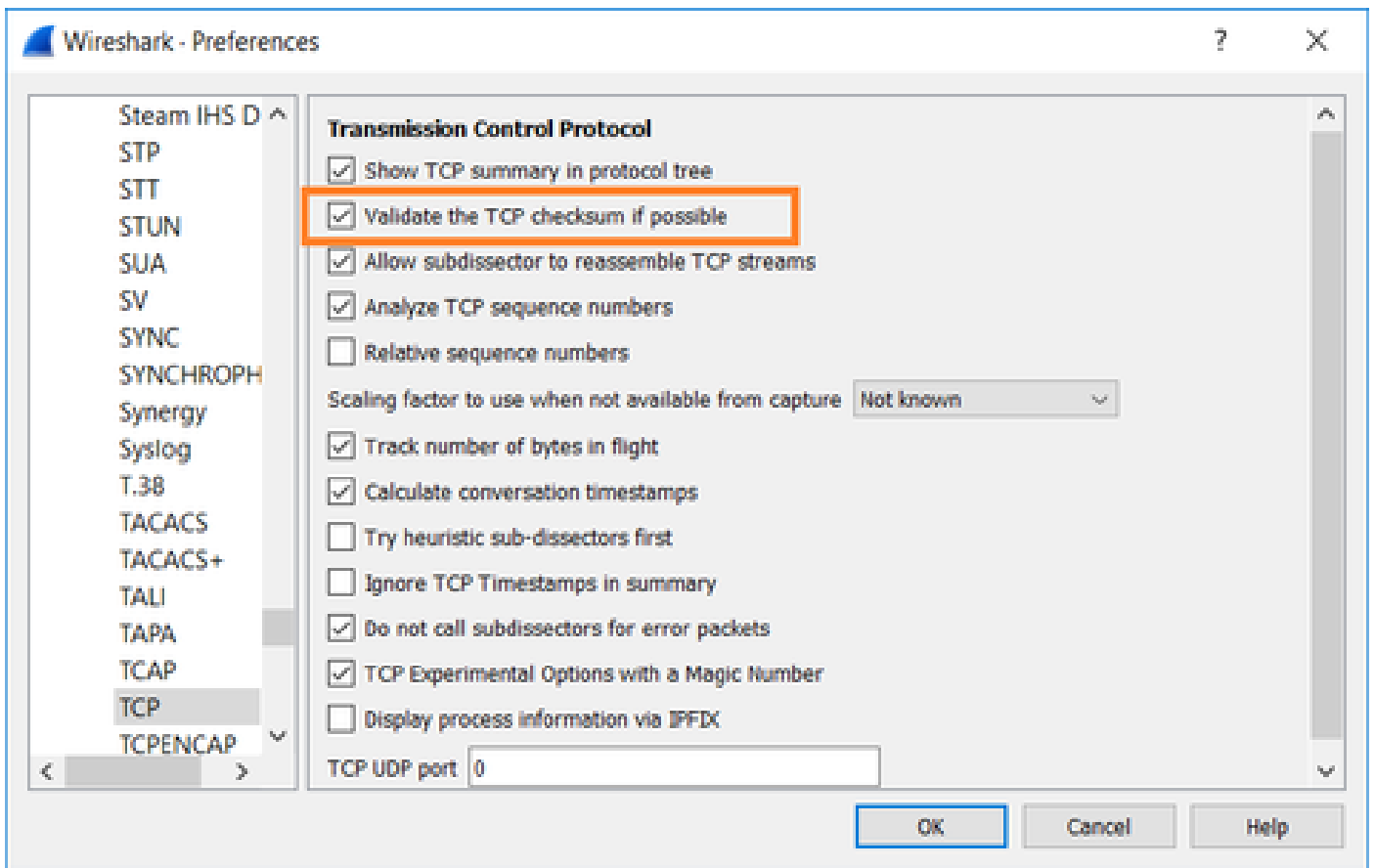
行动1.获取其他捕获。

在服务器上捕获的信息表明，服务器收到包含损坏的TCP校验和的TLS客户端Hello数据包，然后将其静默丢弃（没有指向客户端的TCP RST或任何其他应答数据包）：

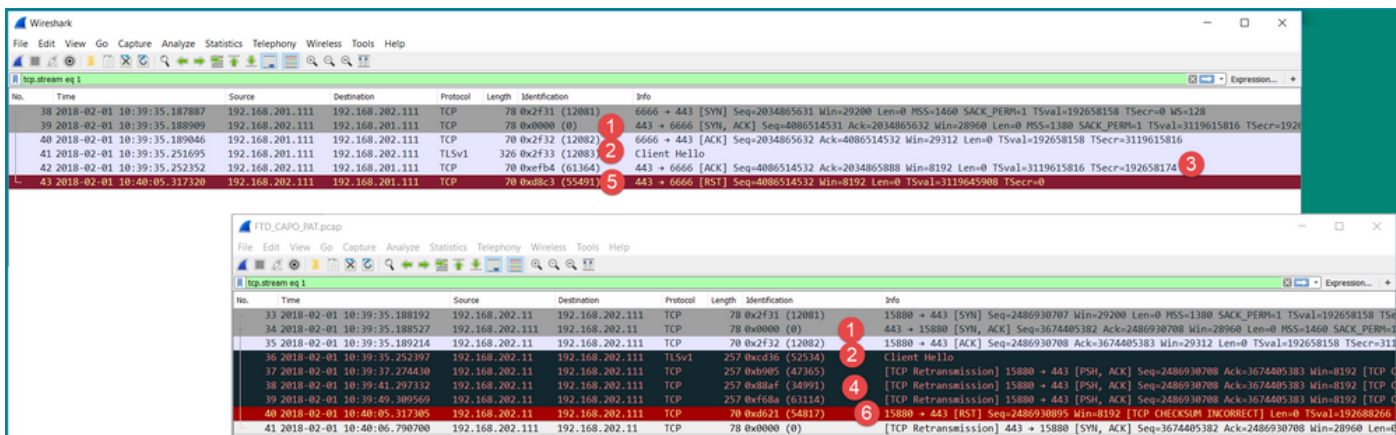
```
21:26:27.133677 IP (tos 0x0, ttl 64, id 52534, offset 0, flags [DF], proto TCP (6), length 239)
  192.168.202.11.15880 > 192.168.202.111.443: Flags [P.], cksum 0x0c65 (incorrect -> 0x3063), seq 1:188, ack 1, win 64, options [nop,nop,T
S val 192658174 ecr 3119615816], length 187
21:26:29.155652 IP (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, ack 1, win 64, options [nop,nop,T
S val 192660198 ecr 0], length 187
21:26:33.178142 IP (tos 0x0, ttl 64, id 34991, offset 0, flags [DF], proto TCP (6), length 239)
  192.168.202.11.15880 > 192.168.202.111.443: Flags [P.], cksum 0x3dd (incorrect -> 0x61fb), seq 1:188, ack 1, win 64, options [nop,nop,T
S val 192664224 ecr 0], length 187
21:26:41.189640 IP (tos 0x0, ttl 64, id 63114, offset 0, flags [DF], proto TCP (6), length 239)
  192.168.202.11.15880 > 192.168.202.111.443: Flags [P.], cksum 0x1e9 (incorrect -> 0x42a7), seq 1:188, ack 1, win 64, options [nop,nop,T
S val 192672244 ecr 0], length 187
21:26:57.195947 IP (tos 0x0, ttl 64, id 54817, offset 0, flags [DF], proto TCP (6), length 52)
  192.168.202.11.15880 > 192.168.202.111.443: Flags [R], cksum 0x9ee (incorrect -> 0xc2e8), seq 2486930895, win 64, options [nop,nop,TS v
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 3674405382, ack 2486930708, win 28960, o
ptions [mss 1460,sackOK,TS val 3119647415 ecr 192658158,nop,wscale 7], length 0
^C
154 packets captured
154 packets received by filter
```

当你把所有东西都放在一起时：

在这种情况下，为了便于理解，需要在Wireshark上启用Validate the TCP checksum if possible选项。导航到Edit > Preferences > Protocols > TCP，如图所示。

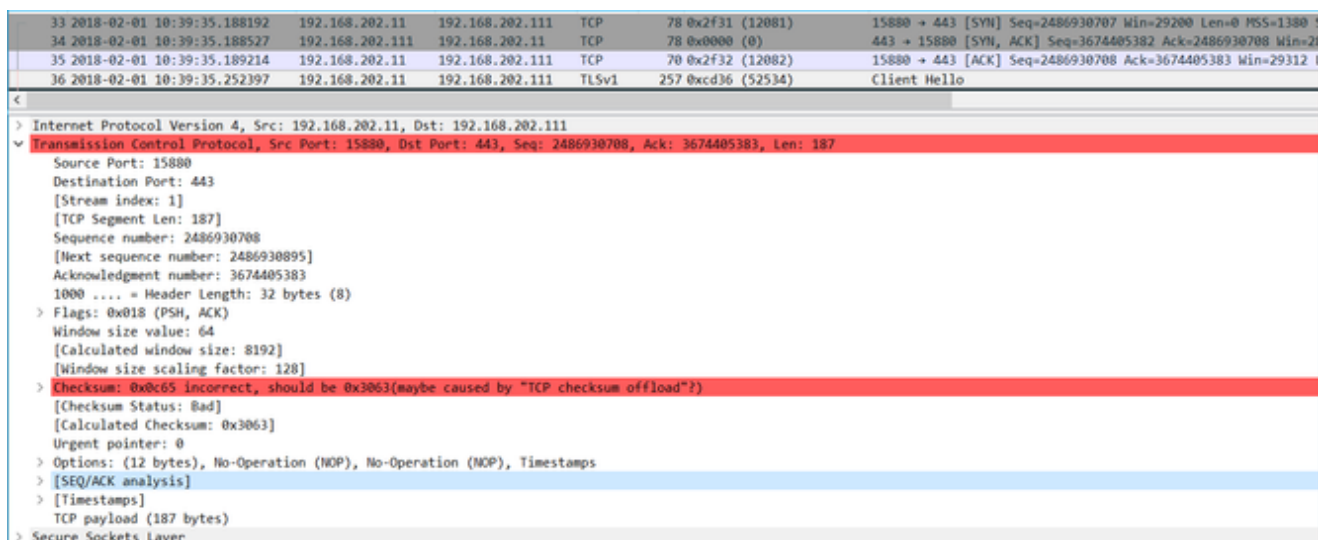


在这种情况下，将捕获并排使用以获得完整信息会很有帮助：



## 要点:

1. 存在TCP三次握手。IP ID相同。这意味着该流未被防火墙代理。
2. TLS客户端Hello来自具有IP ID 12083的客户端。数据包由防火墙代理（在本例中，防火墙配置了TLS解密策略），并且IP ID更改为52534。此外，数据包TCP校验和已损坏（由于稍后修复的软件缺陷）。
3. 防火墙处于TCP代理模式并向客户端（欺骗服务器）发送ACK。



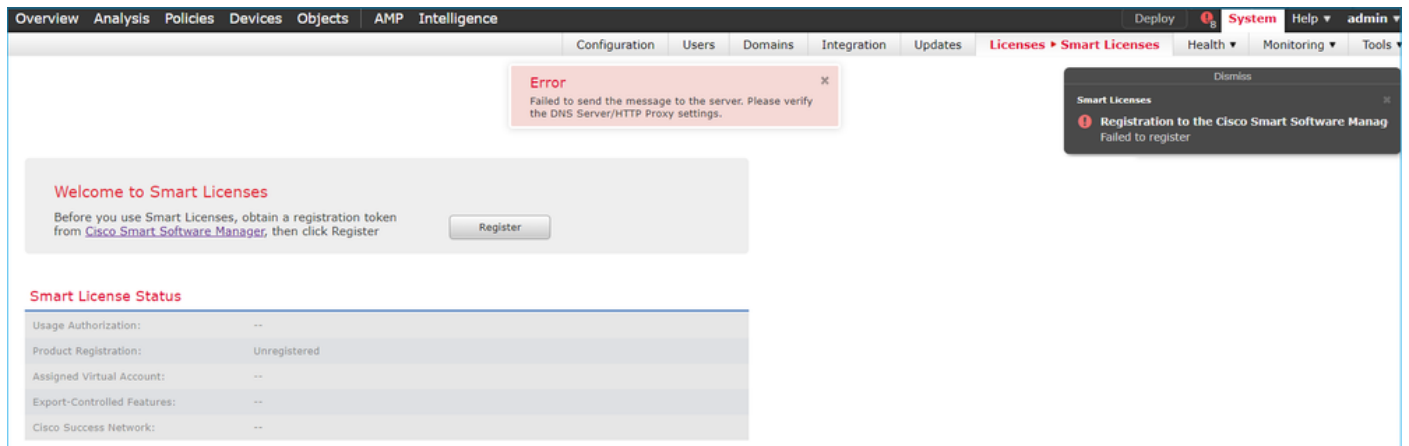
4. 防火墙没有从服务器收到任何TCP ACK数据包，而是重新传输TLS客户端Hello消息。这再次归因于防火墙激活的TCP代理模式。
5. 大约30秒后，防火墙会放弃并向客户端发送TCP RST。
6. 防火墙向服务器发送TCP RST。

## 供参考：

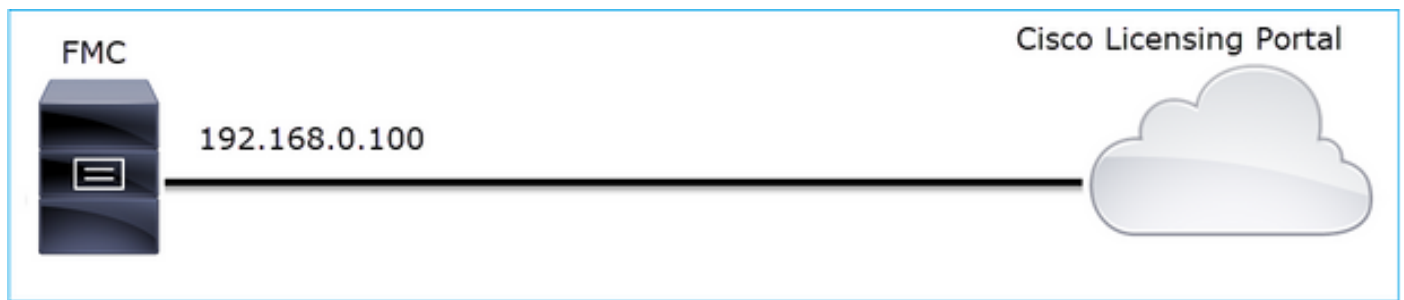
[Firepower TLS/SSL握手处理](#)

## 案例 10.HTTPS连接问题 ( 场景2 )

问题说明：FMC智能许可证注册失败。



下图显示拓扑：



受影响的流：

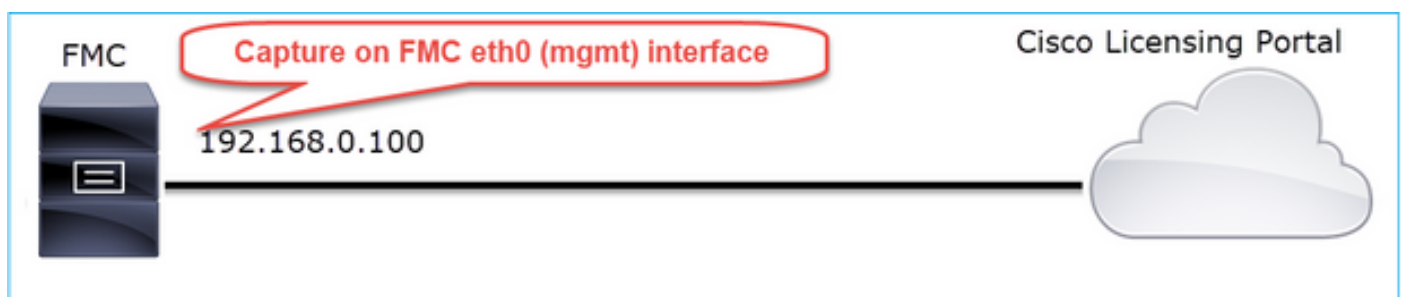
源IP：192.168.0.100

Dst：tools.cisco.com

协议：TCP 443 (HTTPS)

捕获分析

在FMC管理接口上启用捕获：



尝试重新注册。出现错误消息后，按CTRL-C停止捕获：



<#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

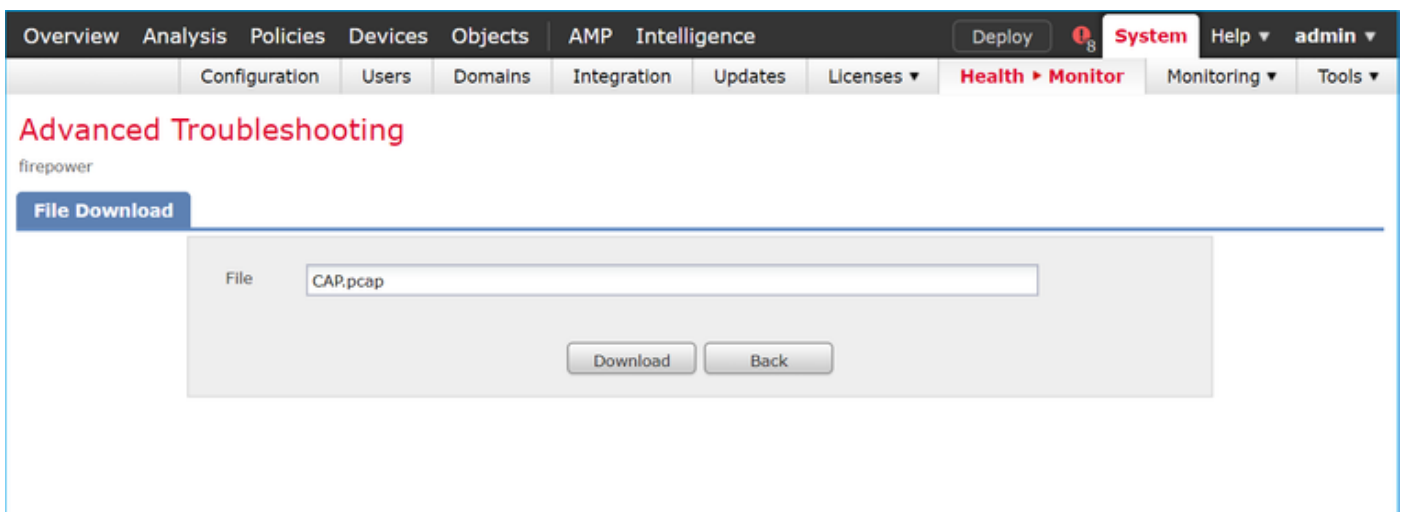
<- CTRL-C

264 packets received by filter

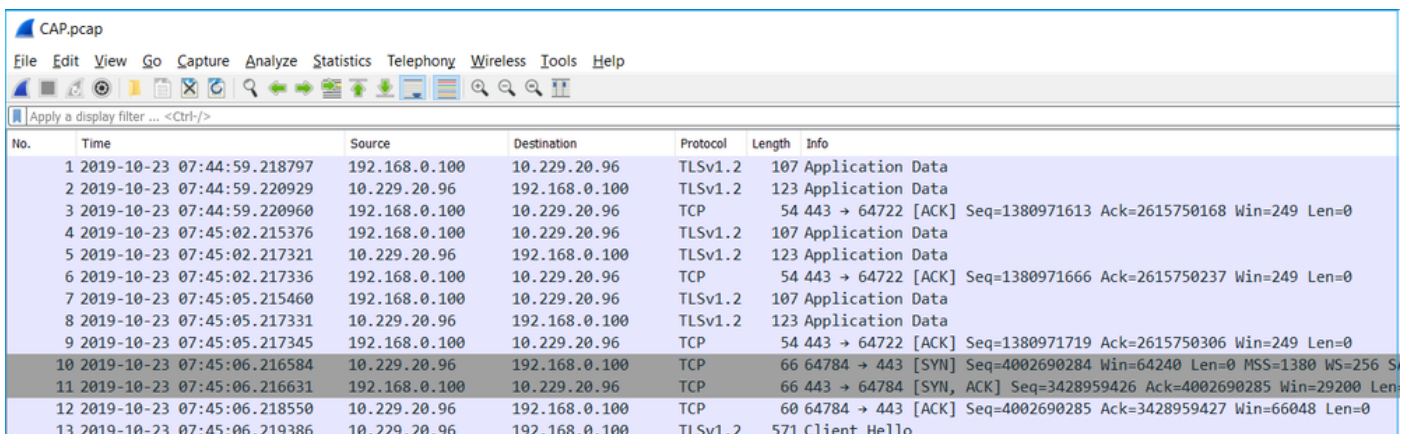
0 packets dropped by kernel

root@firepower:/Volume/home/admin#

从FMC收集捕获(System > Health > Monitor , 选择设备并选择Advanced Troubleshooting) , 如图所示 :



下图显示了Wireshark上的FMC捕获 :



 提示 : 要检查捕获的所有新TCP会话 , 请在Wireshark上使用tcp.flags==0x2显示过滤器。这将



## 过滤捕获的所有TCP SYN数据包。

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=1380 WS=256 SACK_PERM=1
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=1380 WS=256 SACK_PERM=1
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=1380 WS=256 SACK_PERM=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=1380 WS=256 SACK_PERM=1
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=1460 SACK_PERM=1 TSval=16127801 TSecr=0 WS=128
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=1460 SACK_PERM=1 TSval=16138303 TSecr=0 WS=128
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=1460 SACK_PERM=1 TSval=2040670996 TSecr=0 WS=128
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=1460 SACK_PERM=1 TSval=2040681496 TSecr=0 WS=128
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=1460 SACK_PERM=1 TSval=16169802 TSecr=0 WS=128
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=1380 WS=256 SACK_PERM=1
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=1380 WS=256 SACK_PERM=1

## 提示：将SSL客户端Hello中的Server Name字段应用为列。

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

Secure Sockets Layer

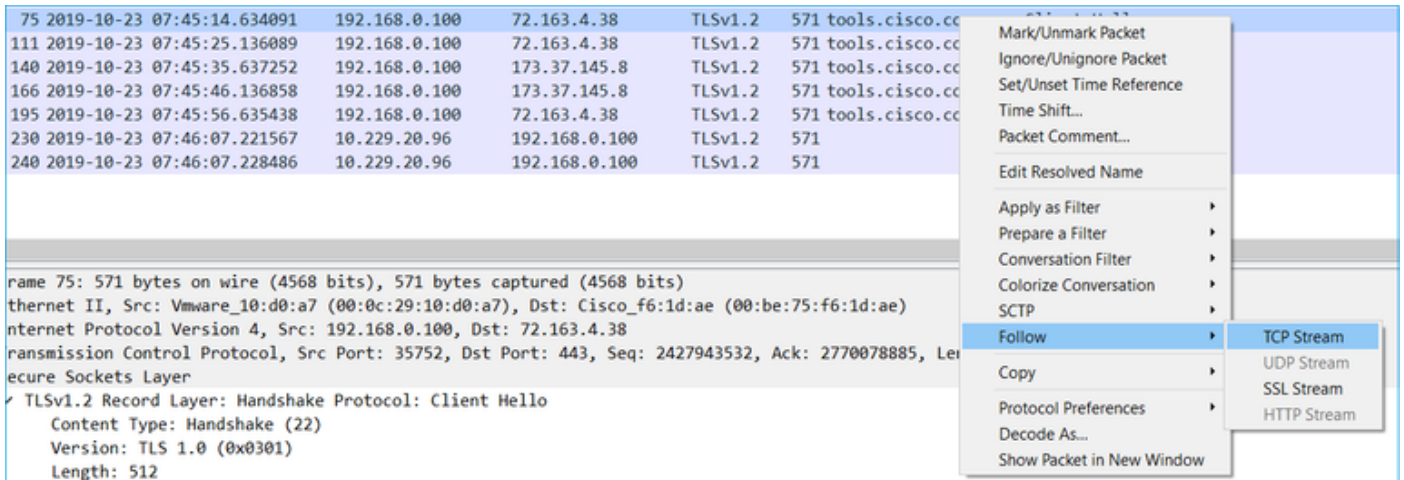
- TLsv1.2 Record Layer: Handshake Protocol  
Content Type: Handshake (22)  
Version: TLS 1.0 (0x0301)  
Length: 512
  - Handshake Protocol: Client Hello  
Handshake Type: Client Hello (1)  
Length: 508  
Version: TLS 1.2 (0x0303)
    - Random: 234490a107438c73b595646532
    - Session ID Length: 0
    - Cipher Suites Length: 100
    - Cipher Suites (50 suites)
    - Compression Methods Length: 1
    - Compression Methods (1 method)
    - Extensions Length: 367
      - Extension: server\_name (len=20)  
Type: server\_name (0)  
Length: 20
        - Server Name Indication extension  
Server Name list length: 18  
Server Name Type: host\_name (0)  
Server Name length: 15  
Server Name: tools.cisco.com

## 提示：应用此显示过滤器以仅查看客户端Hello消息ssl.handshake.type == 1

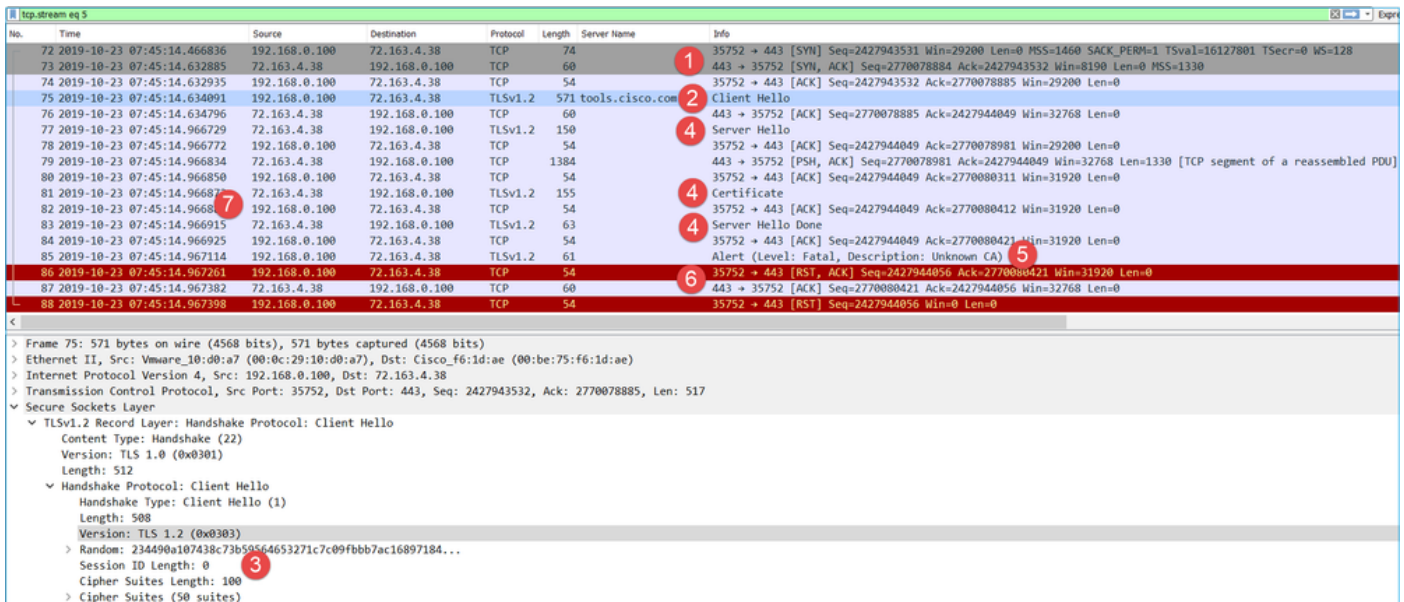
No.	Time	Source	Destination	Protocol	Length	Server Name	Info
13	2019-10-23 07:45:06.219386	10.229.20.96	192.168.0.100	TLSv1.2	571		Client Hello
23	2019-10-23 07:45:06.227250	10.229.20.96	192.168.0.100	TLSv1.2	571		Client Hello
48	2019-10-23 07:45:12.406366	10.229.20.96	192.168.0.100	TLSv1.2	571		Client 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

 注意：在撰写本文时，智能许可门户(tools.cisco.com)使用以下IP：72.163.4.38、173.37.145.8

按照其中一个TCP流操作(Follow > TCP Stream)，如图所示。



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=1460 SACK_PERM=1 TSval=16127801 TSecr=0 WS=128
73	2019-10-23 07:45:14.632855	72.163.4.38	192.168.0.100	TCP	60		443 → 35752 [SYN, ACK] Seq=2770078884 Ack=2427943532 Win=8190 Len=0 MSS=1330
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=29200 Len=0
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=32768 Len=0
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=29200 Len=0
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=2427944049 Win=32768 Len=1330 [TCP segment of a reassembled PDU]
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=31920 Len=0
81	2019-10-23 07:45:14.966877	72.163.4.38	192.168.0.100	TLSv1.2	155		Certificate
82	2019-10-23 07:45:14.966887	192.168.0.100	72.163.4.38	TCP	54		35752 → 443 [ACK] Seq=2427944049 Ack=2770080421 Win=31920 Len=0
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=31920 Len=0
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=2770080421 Win=31920 Len=0
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=32768 Len=0
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
Secure Sockets Layer
  TLSv1.2 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 512
  Handshake Protocol: Client Hello
    Handshake Type: Client Hello (1)
    Length: 508
    Version: TLS 1.2 (0x0303)
    Random: 234490a107438c73b55564653271c7c09fbbb7ac16897184...
    Session ID Length: 0
    Cipher Suites Length: 100
    Cipher Suites (50 suites)
```

要点:

1. 存在TCP三次握手。
2. 客户端(FMC)向智能许可门户发送SSL客户端Hello消息。
3. SSL会话ID为0。这意味着它不是续会。
4. 目标服务器回复服务器Hello、证书和服务器Hello完成消息。
5. 客户端发送有关“Unknown CA”的SSL严重警报。
6. 客户端发送TCP RST以关闭会话。
7. 整个TCP会话持续时间（从建立到关闭）约为0.5秒。

选择Server Certificate，然后展开issuer字段以查看commonName。在本例中，公用名显示执行中间人(MITM)的设备。

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=1460 SACK_PERM=1 TSval=16127801
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=2427943532 Win=8190 Len=0 MSS=1330
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=29200 Len=0
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=32768 Len=0
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=29200 Len=0
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=2427944049 Win=32768 Len=1330 [TCP segment
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=31920 Len=0
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 Systems, Inc.,id-at-localityName=San Jose,id-at-sto
                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)
                        > RDNSquence item: 1 item (id-at-organizationName=FTD_O)
                        > RDNSquence item: 1 item (id-at-organizationalUnitName=FTD_OU)
                        > RDNSquence item: 1 item (id-at-commonName=FTD4100_MITM)
                  > validity
                  > subject: rdnSequence (0)
                  > subjectPublicKeyInfo
                > extensions: 6 items
  
```

如下图所示：

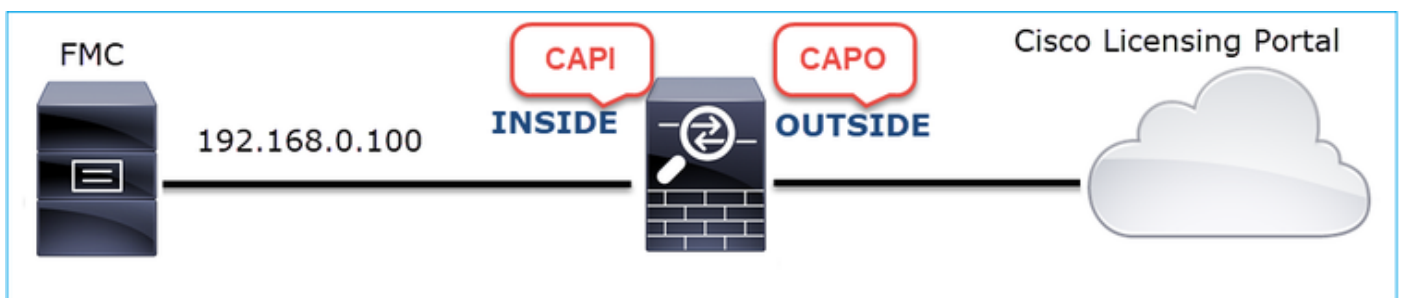


### 推荐的操作

本部分列出的操作旨在进一步缩小问题范围。

行动1.获取其他捕获。

捕获传输防火墙设备：



CAPI显示：



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=427175838 Win=29200 Len=0 MSS=1460 SACK_PERM=1
1222	2019-10-22 17:49:03.379023	173.37.145.8	192.168.0.100	TCP	58		443 → 39924 [SYN, ACK] Seq=236460465 Ack=427175839 Win=8190 Len=0 MSS=1336
1223	2019-10-22 17:49:03.379298	192.168.0.100	173.37.145.8	TCP	54		39924 → 443 [ACK] Seq=427175839 Ack=236460466 Win=29200 Len=0
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=236460466 Ack=427176356 Win=32768 Len=0
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] Seq=236460562 Ack=427176356 Win=32768 Len=1330
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=427176356 Ack=236460562 Win=29200 Len=0
1231	2019-10-22 17:49:03.710519	192.168.0.100	173.37.145.8	TCP	54		39924 → 443 [ACK] Seq=427176356 Ack=236461892 Win=31920 Len=0
1232	2019-10-22 17:49:03.710519	192.168.0.100	173.37.145.8	TCP	54		39924 → 443 [ACK] Seq=427176356 Ack=236461993 Win=31920 Len=0
1233	2019-10-22 17:49:03.710534	192.168.0.100	173.37.145.8	TCP	54		39924 → 443 [ACK] Seq=427176356 Ack=236462002 Win=31920 Len=0
1234	2019-10-22 17:49:03.710626	192.168.0.100	173.37.145.8	TLSv1.2	61		Alert (Level: Fatal, Description: Unknown CA)
1235	2019-10-22 17:49:03.710641	173.37.145.8	192.168.0.100	TCP	54		443 → 39924 [ACK] Seq=236462002 Ack=427176363 Win=32768 Len=0
1236	2019-10-22 17:49:03.710748	192.168.0.100	173.37.145.8	TCP	54		39924 → 443 [RST, ACK] Seq=427176363 Ack=236462002 Win=31920 Len=0
1237	2019-10-22 17:49:03.710870	192.168.0.100	173.37.145.8	TCP	54		39924 → 443 [RST] Seq=427176363 Win=0 Len=0

```

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 Systems, Inc.,id-at-localityName=San Jose)
    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)
          RDNSSequence item: 1 item (id-at-organizationName=FTD_O)
          RDNSSequence item: 1 item (id-at-organizationalUnitName=FTD_OU)
          RDNSSequence item: 1 item (id-at-commonName=FTD4100_MITM)
      validity
  
```

CAPO显示：

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=623942018 Win=29200 Len=0 MSS=1380 SACK_PERM=1 TSval=1169
1170	2019-10-22 17:49:03.378962	173.37.145.8	192.168.0.100	TCP	62		443 → 39924 [SYN, ACK] Seq=4179450724 Ack=623942019 Win=8190 Len=0 MSS=1336
1171	2019-10-22 17:49:03.379329	192.168.0.100	173.37.145.8	TCP	58		39924 → 443 [ACK] Seq=623942019 Ack=4179450725 Win=29200 Len=0
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=4179450725 Ack=623942473 Win=34780 Len=1330 [TCP
1174	2019-10-22 17:49:03.545809	173.37.145.8	192.168.0.100	TCP	1388		443 → 39924 [PSH, ACK] Seq=4179452055 Ack=623942473 Win=34780 Len=1330 [TCP
1175	2019-10-22 17:49:03.545824	192.168.0.100	173.37.145.8	TCP	58		39924 → 443 [ACK] Seq=623942473 Ack=4179453385 Win=65535 Len=0
1176	2019-10-22 17:49:03.545915	173.37.145.8	192.168.0.100	TCP	1388		443 → 39924 [PSH, ACK] Seq=4179453385 Ack=623942473 Win=34780 Len=1330 [TCP
1177	2019-10-22 17:49:03.545961	173.37.145.8	192.168.0.100	TCP	1388		443 → 39924 [PSH, ACK] Seq=4179454715 Ack=623942473 Win=34780 Len=1330 [TCP
1178	2019-10-22 17:49:03.545961	192.168.0.100	173.37.145.8	TCP	58		39924 → 443 [ACK] Seq=623942473 Ack=4179456045 Win=65535 Len=0
1179	2019-10-22 17:49:03.709420	173.37.145.8	192.168.0.100	TLSv1.2	82		Server Hello, Certificate, Server Hello Done
1180	2019-10-22 17:49:03.710687	192.168.0.100	173.37.145.8	TLSv1.2	65		Alert (Level: Fatal, Description: Unknown CA)
1181	2019-10-22 17:49:03.710885	192.168.0.100	173.37.145.8	TCP	58		39924 → 443 [FIN, PSH, ACK] Seq=623942480 Ack=4179456069 Win=65535 Len=0
1182	2019-10-22 17:49:03.874542	173.37.145.8	192.168.0.100	TCP	58		443 → 39924 [RST, ACK] Seq=4179456069 Ack=623942480 Win=9952 Len=0

```

Length: 5339
  Handshake Protocol: Server Hello
  Handshake Protocol: Certificate
    Handshake Type: Certificate (11)
    Length: 5240
    Certificates Length: 5237
  Certificates (5237 bytes)
    Certificate Length: 2025
  Certificate: 308207e5308205cda00302010202143000683b0f7504f7b2... (id-at-commonName=tools.cisco.com,id-at-organizationName=Cisco Systems, Inc.,id-at-localityName=San Jose)
    signedCertificate
      algorithmIdentifier (sha256WithRSAEncryption)
      Padding: 0
      encrypted: 6921d084f7a6f6167058f14e2aad8b98b4e6c971ea6ea3b4...
    Certificate Length: 1736
  Certificate: 308206c4308204ca00302010202147517167783d0437eb5... (id-at-commonName=HydrantID SSL ICA G2,id-at-organizationName=HydrantID (Avalanche Cloud Corporation),id-at-localityName=San Jose)
    signedCertificate
      version: v3 (2)
      serialNumber: 0x7517167783d0437eb556c357946e4563b8ebd3ac
      signature (sha256WithRSAEncryption)
      issuer: rdnSequence (0)
        rdnSequence: 3 items (id-at-commonName=QuoVadis Root CA 2,id-at-organizationName=QuoVadis Limited,id-at-countryName=BM)
      validity
  
```

这些捕获证明传输防火墙修改了服务器证书(MITM)

行动2.检查设备日志。

您可以按照本文档中的说明收集FMC TS捆绑包：

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

在本例中，/dir-archives/var-log/process\_stdout.log文件显示如下消息：

```
<#root>
```

```
SOUT: 10-23 05:45:14 2019-10-23 05:45:36 s1a[10068]: *Wed .967 UTC: CH-LIB-ERROR: ch_pf_curl_send_msg[4]
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 s1a[10068]: *Wed .967 UTC: CH-LIB-TRACE: ch_pf_curl_is_cert_is
cert issue checking, ret 60, url "https://tools.cisco.com/its/
```

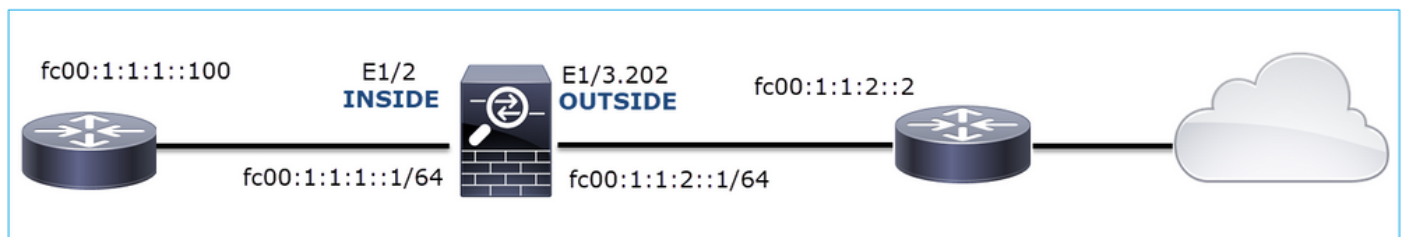
## 推荐方案

禁用特定流的MITM，以便FMC可以成功注册到智能许可云。

## 案例 11.IPv6连接问题

问题描述：内部主机（位于防火墙的INSIDE接口之后）无法与外部主机（位于防火墙的OUTSIDE接口之后的主机）通信。

下图显示拓扑：



受影响的流：

源IP：fc00:1:1:1::100

目标IP：fc00:1:1:2::2

协议：任意

## 捕获分析

在FTD LINA引擎上启用捕获。

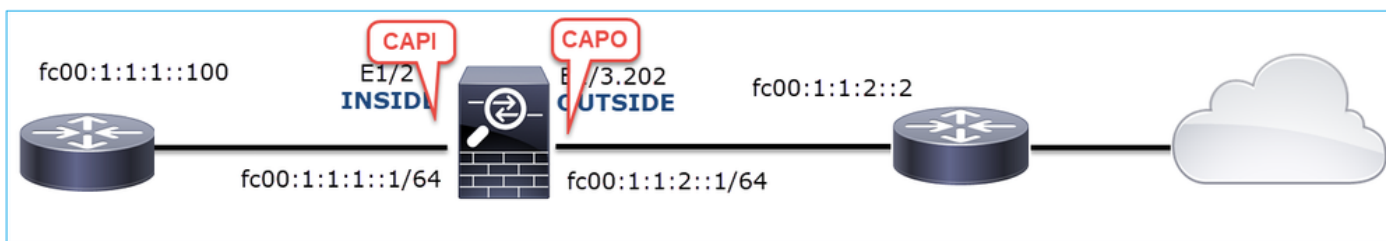
```
<#root>
```

```
firepower#
```

```
capture CAPI int INSIDE match ip any6 any6
```

```
firepower#
```

```
capture CAPO int OUTSIDE match ip any6 any6
```



## 捕获-非功能场景

这些捕获与从IP fc00:1:1:1:100 (内部路由器) 到IP fc00:1:1:2:2 (上游路由器) 的ICMP连接测试并行执行。

防火墙INSIDE接口上的捕获包含：

No.	Time	Source	Destination	Protocol	Length	Info
1	2019-10-24 13:02:07.001663	fc00:1:1:1:100	ff02::1:ff00:1	ICMPv6	86	Neighbor Solicitation for fc00:1:1:1:1 from 4c:4e:35:fc:fc:d8
2	2019-10-24 13:02:07.001876	fc00:1:1:1:1	fc00:1:1:1:100	ICMPv6	86	Neighbor Advertisement fc00:1:1:1:1 (rtr, sol, ovr) is at 00:be:75:f6:1d:ae
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, seq=0, hop limit=64 (no response found!)
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, seq=1, hop limit=64 (no response found!)
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, seq=2, hop limit=64 (no response found!)
6	2019-10-24 13:02:11.999917	fe80::2be:75ff:fe6:1dae	fc00:1:1:1:100	ICMPv6	86	Neighbor Solicitation for fc00:1:1:1:100 from 00:be:75:f6:1d:ae
7	2019-10-24 13:02:12.002075	fc00:1:1:1:100	fc00:1:1:1:100	ICMPv6	78	Neighbor Advertisement fc00:1:1:1:100 (rtr, sol)
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, seq=3, hop limit=64 (no response found!)
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, seq=4, hop limit=64 (no response found!)
10	2019-10-24 13:02:17.062725	fe80::4e4e:35ff:fe6:fc:d8	fe80::2be:75ff:fe6:1dae	ICMPv6	86	Neighbor Solicitation for fe80::2be:75ff:fe6:1dae from 4c:4e:35:fc:fc:d8
11	2019-10-24 13:02:17.062862	fe80::2be:75ff:fe6:1dae	fe80::4e4e:35ff:fe6:fc:d8	ICMPv6	78	Neighbor Advertisement fe80::2be:75ff:fe6:1dae (rtr, sol)
12	2019-10-24 13:02:22.059994	fe80::2be:75ff:fe6:1dae	fe80::4e4e:35ff:fe6:fc:d8	ICMPv6	86	Neighbor Solicitation for fe80::4e4e:35ff:fe6:fc:d8 from 00:be:75:f6:1d:ae
13	2019-10-24 13:02:22.063000	fe80::4e4e:35ff:fe6:fc:d8	fe80::2be:75ff:fe6:1dae	ICMPv6	78	Neighbor Advertisement fe80::4e4e:35ff:fe6:fc:d8 (rtr, sol)

## 要点:

1. 路由器发送IPv6邻居请求消息并请求上游设备的MAC地址(IP fc00:1:1:1:1)。
2. 防火墙以IPv6邻居通告作为回应。
3. 路由器发送ICMP回应请求。
4. 防火墙发送IPv6邻居请求消息并请求下游设备的MAC地址(fc00:1:1:1:100)。
5. 路由器会以IPv6邻居通告做出响应。
6. 路由器发送额外的IPv6 ICMP回应请求。

防火墙OUTSIDE接口上的捕获包含：

No.	Time	Source	Destination	Protocol	Length	Info
1	2019-10-24 13:02:07.002517	fe80::2be:75ff:fe6:1d8e	ff02::1:ff00:2	ICMPv6	90	Neighbor Solicitation for fc00:1:1:2:2 from 00:be:75:f6:1d:8e
2	2019-10-24 13:02:07.005569	fc00:1:1:2:2	fe80::2be:75ff:fe6:1d8e	ICMPv6	90	Neighbor Advertisement fc00:1:1:2:2 (rtr, sol, ovr) is at 4c:4e:35:fc:fc:d8
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, seq=1, hop limit=64 (no response found!)
4	2019-10-24 13:02:09.001815	fc00:1:1:2:2	ff02::1:ff00:100	ICMPv6	90	Neighbor Solicitation for fc00:1:1:1:100 from 4c:4e:35:fc:fc:d8
5	2019-10-24 13:02:10.025938	fc00:1:1:2:2	ff02::1:ff00:100	ICMPv6	90	Neighbor Solicitation for fc00:1:1:1:100 from 4c:4e:35:fc:fc:d8
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, seq=2, hop limit=64 (no response found!)
7	2019-10-24 13:02:11.050015	fc00:1:1:2:2	ff02::1:ff00:100	ICMPv6	90	Neighbor Solicitation for fc00:1:1:1:100 from 4c:4e:35:fc:fc:d8
8	2019-10-24 13:02:12.066082	fe80::4e4e:35ff:fe6:fc:d8	fe80::2be:75ff:fe6:1d8e	ICMPv6	90	Neighbor Solicitation for fe80::2be:75ff:fe6:1d8e from 4c:4e:35:fc:fc:d8
9	2019-10-24 13:02:12.066234	fe80::2be:75ff:fe6:1d8e	fe80::4e4e:35ff:fe6:fc:d8	ICMPv6	82	Neighbor Advertisement fe80::2be:75ff:fe6:1d8e (rtr, sol)
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, seq=3, hop limit=64 (no response found!)
11	2019-10-24 13:02:13.002105	fc00:1:1:2:2	ff02::1:ff00:100	ICMPv6	90	Neighbor Solicitation for fc00:1:1:1:100 from 4c:4e:35:fc:fc:d8
12	2019-10-24 13:02:14.090251	fc00:1:1:2:2	ff02::1:ff00:100	ICMPv6	90	Neighbor Solicitation for fc00:1:1:1:100 from 4c:4e:35:fc:fc:d8
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, seq=4, hop limit=64 (no response found!)
14	2019-10-24 13:02:15.178350	fc00:1:1:2:2	ff02::1:ff00:100	ICMPv6	90	Neighbor Solicitation for fc00:1:1:1:100 from 4c:4e:35:fc:fc:d8
15	2019-10-24 13:02:17.059963	fe80::2be:75ff:fe6:1d8e	fe80::4e4e:35ff:fe6:fc:d8	ICMPv6	90	Neighbor Solicitation for fe80::4e4e:35ff:fe6:fc:d8 from 00:be:75:f6:1d:8e
16	2019-10-24 13:02:17.062512	fe80::4e4e:35ff:fe6:fc:d8	fe80::2be:75ff:fe6:1d8e	ICMPv6	82	Neighbor Advertisement fe80::4e4e:35ff:fe6:fc:d8 (rtr, sol)

## 要点:

1. 防火墙发送IPv6邻居请求消息，请求上游设备的MAC地址(IP fc00:1:1:2:2)。
2. 路由器会以IPv6邻居通告做出响应。
3. 防火墙发送IPv6 ICMP回应请求。
4. 上游设备 (路由器fc00:1:1:2:2) 发送IPv6邻居请求消息，该消息请求IPv6地址 fc00:1:1:1:100的MAC地址。

5. 防火墙发送额外的IPv6 ICMP回应请求。

6. 上游路由器发送另一个IPv6邻居请求消息，要求获取IPv6地址fc00:1:1:1::100的MAC地址。

第四点很有意思。通常，上游路由器会要求防火墙OUTSIDE接口(fc00:1:1:2::2)的MAC地址，但实际上它要求的是fc00:1:1:1::100。这表示配置错误。

### 推荐的操作

本部分列出的操作旨在进一步缩小问题范围。

行动1.检查IPv6邻居表。

防火墙IPv6邻居表已正确填充。

<#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
```

行动2.检查IPv6配置。

这是防火墙配置。

<#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

```

上游设备配置显示了配置错误：

```

<#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

```

### 捕获-功能场景

子网掩码更改（从/48更改为/64）解决了问题。这是功能场景中的CAPI捕获。

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 fc00:1:1:1::1 from 4c:4e:35:fc:fc:d8
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::1 (rtr, sol, ovr) is at 00:be:75:f6:1d:ae
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=0, hop limit=64 (no response found!)
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=1, hop limit=64 (no response found!)
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=2, hop limit=64 (reply in 6)
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, hop limit=64 (request in 5)
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=3, hop limit=64 (reply in 8)
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, hop limit=64 (request in 7)
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=4, hop limit=64 (reply in 10)
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, hop limit=64 (request in 9)

要点：

1. 路由器发送IPv6邻居请求消息，请求上游设备的MAC地址(IP fc00:1 : 1:1 : : 1)。
2. 防火墙以IPv6邻居通告作为回应。
3. 路由器发送ICMP回应请求并获得应答。

CAPO内容：

No.	Time	Source	Destination	Protocol	Length	Info
1	2019-10-24 15:17:20.678645	fe80::2be:75ff:fe...	ff02::1:ff00:2	ICMPv6	90	Neighbor Solicitation for fc00:1:1:2::2 from 00:be:75:f6:1d:8e
2	2019-10-24 15:17:20.681818	fc00:1:1:2::2	fe80::2be:75ff:fe...	ICMPv6	90	Neighbor Advertisement fc00:1:1:2::2 (rtr, sol, ovr) is at 4c:4e:35:fc:fc:d8
3	2019-10-24 15:17:22.674342	fc00:1:1:1::100	fc00:1:1:2::2	ICMPv6	118	Echo (ping) request id=0x097e, seq=1, hop limit=64 (reply in 6)
4	2019-10-24 15:17:22.677943	fc00:1:1:2::2	ff02::1:ff00:1	ICMPv6	90	Neighbor Solicitation for fc00:1:1:2::1 from 4c:4e:35:fc:fc:d8
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::1 (rtr, sol, ovr) is at 00:be:75:f6:1d:8e
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, hop limit=64 (request in 3)
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=2, hop limit=64 (reply in 8)
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, hop limit=64 (request in 7)
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=3, hop limit=64 (reply in 10)
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, hop limit=64 (request in 9)
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=4, hop limit=64 (reply in 12)
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, hop limit=64 (request in 11)

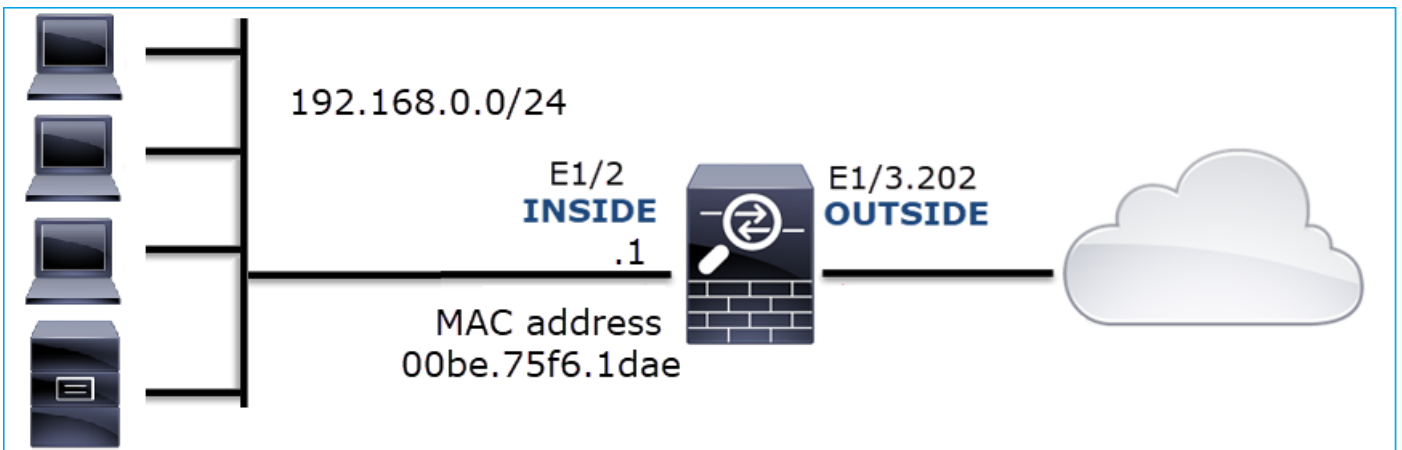
要点:

1. 防火墙发送IPv6邻居请求消息，请求上游设备的MAC地址(IP fc00:1 : 1:2 : : 2)。
2. 防火墙以IPv6邻居通告作为回应。
3. 防火墙发送ICMP回应请求。
4. 路由器发送IPv6邻居请求消息，请求下游设备的MAC地址(IP fc00:1 : 1:1 : : 1)。
5. 防火墙以IPv6邻居通告作为回应。
6. 防火墙发送ICMP回应请求并获得应答。

## 案例 12.间歇性连接问题 ( ARP毒化 )

问题描述：内部主机(192.168.0.x/24)与同一子网中的主机存在间歇性连接问题

下图显示拓扑：



受影响的流：

源IP：192.168.0.x/24

目标IP：192.168.0.x/24

协议：任意

内部主机的ARP缓存似乎已中毒：

```
C:\Windows\system32\cmd.exe
C:\Users\mzafeirol>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\mzafeirol>
```

### 捕获分析

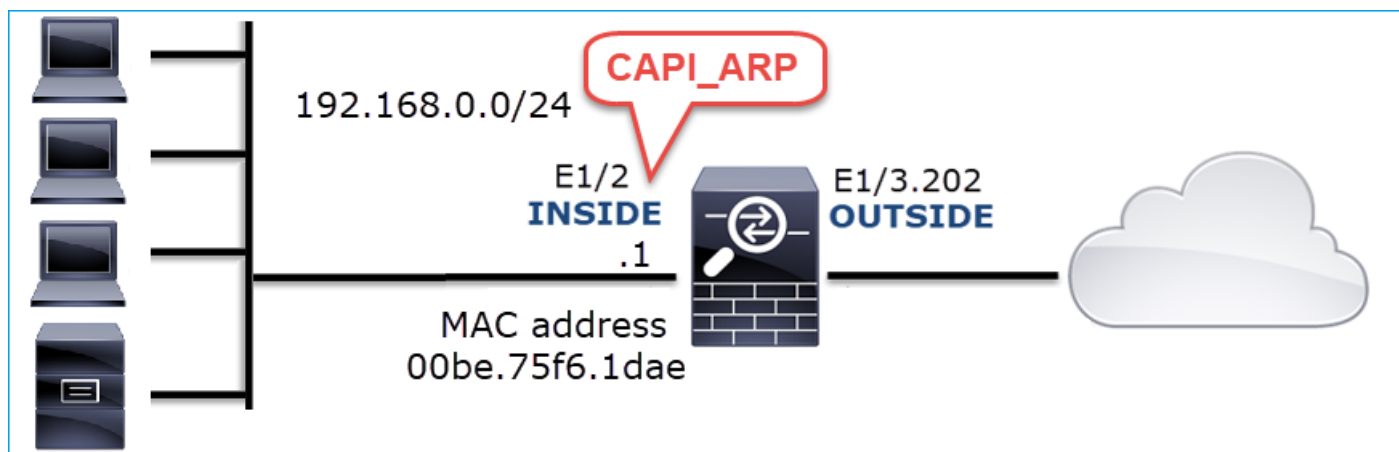
在FTD LINA引擎上启用捕获

此捕获仅捕获内部接口上的ARP数据包：

```
<#root>
```

```
firepower#
```

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



捕获-非功能场景：

防火墙INSIDE接口上的捕获包含。

No.	Time	Source	Destination	Protocol	Length	Info
4	2019-10-25 10:01:55.179571	Vmware_2c:9b:a7	Broadcast	ARP	60	Who has 192.168.0.23? Tell 192.168.0.55
5	2019-10-25 10:01:55.17969	Cisco_f6:1d:ae	Vmware_2c:9b:a7	ARP	42	192.168.0.23 is at 00:be:75:f6:1d:ae
35	2019-10-25 10:02:13.050397	Vmware_2c:9b:a7	Broadcast	ARP	60	Who has 192.168.0.24? Tell 192.168.0.55
36	2019-10-25 10:02:13.050488	Cisco_f6:1d:ae	Vmware_2c:9b:a7	ARP	42	192.168.0.24 is at 00:be:75:f6:1d:ae
47	2019-10-25 10:02:19.284683	Vmware_2c:9b:a7	Broadcast	ARP	60	Who has 192.168.0.25? Tell 192.168.0.55
48	2019-10-25 10:02:19.284775	Cisco_f6:1d:ae	Vmware_2c:9b:a7	ARP	42	192.168.0.25 is at 00:be:75:f6:1d:ae
61	2019-10-25 10:02:25.779821	Vmware_2c:9b:a7	Broadcast	ARP	60	Who has 192.168.0.26? Tell 192.168.0.55
62	2019-10-25 10:02:25.779912	Cisco_f6:1d:ae	Vmware_2c:9b:a7	ARP	42	192.168.0.26 is at 00:be:75:f6:1d:ae
76	2019-10-25 10:02:31.978175	Vmware_2c:9b:a7	Broadcast	ARP	60	Who has 192.168.0.27? Tell 192.168.0.55
77	2019-10-25 10:02:31.978251	Cisco_f6:1d:ae	Vmware_2c:9b:a7	ARP	42	192.168.0.27 is at 00:be:75:f6:1d:ae
97	2019-10-25 10:02:38.666515	Vmware_2c:9b:a7	Broadcast	ARP	60	Who has 192.168.0.28? Tell 192.168.0.55
98	2019-10-25 10:02:38.666606	Cisco_f6:1d:ae	Vmware_2c:9b:a7	ARP	42	192.168.0.28 is at 00:be:75:f6:1d:ae
121	2019-10-25 10:02:47.384074	Vmware_2c:9b:a7	Broadcast	ARP	60	Who has 192.168.0.29? Tell 192.168.0.55
122	2019-10-25 10:02:47.384150	Cisco_f6:1d:ae	Vmware_2c:9b:a7	ARP	42	192.168.0.29 is at 00:be:75:f6:1d:ae
137	2019-10-25 10:02:53.539995	Vmware_2c:9b:a7	Broadcast	ARP	60	Who has 192.168.0.30? Tell 192.168.0.55
138	2019-10-25 10:02:53.540087	Cisco_f6:1d:ae	Vmware_2c:9b:a7	ARP	42	192.168.0.30 is at 00:be:75:f6:1d:ae

要点:

1. 防火墙接收192.168.0.x/24网络内IP的各种ARP请求
2. 防火墙使用自己的MAC地址对所有设备 (代理ARP) 做出应答

推荐的操作

本部分列出的操作旨在进一步缩小问题范围。

行动1.检查NAT配置。

对于NAT配置，有些情况下no-proxy-arp关键字可防止早期行为：

```
<#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
```

行动2.在防火墙接口上禁用proxy-arp功能。

如果“no-proxy-arp”关键字不能解决问题，请尝试在接口上禁用代理ARP。如果是FTD，在撰写本文时，您必须使用FlexConfig并部署命令（指定适当的接口名称）。

```
sysopt noproxyarp INSIDE
```

案例 13.标识导致CPU占用的SNMP对象标识符(OID)

本例演示了如何根据SNMP版本3 (SNMPv3)数据包捕获的分析，将内存轮询的某些SNMP OID标识为CPU大量占用（性能问题）的根本原因。

问题描述：数据接口上的超限持续增加。进一步的研究表明，还有CPU大量占用（由SNMP进程引起）是接口超负荷运行的根本原因。

故障排除过程中的下一步是确定由SNMP进程导致的CPU大量占用的根本原因，特别是缩小问题的范围，以确定SNMP对象标识符(OID)，在轮询时，OID可能会导致CPU大量占用。

目前，FTD LINA引擎不为实时轮询的SNMP OID提供“show”命令。

用于轮询的SNMP OID列表可以从SNMP监控工具中检索，但是，在这种情况下，存在以下预防因素：

- FTD管理员没有访问SNMP监控工具的权限
- 在FTD上配置了具有身份验证和数据加密的SNMP第3版

## 捕获分析

由于FTD管理员拥有SNMP第3版身份验证和数据加密的凭证，因此建议以下行动计划：

1. 获取SNMP数据包捕获
2. 保存捕获并使用Wireshark SNMP协议首选项指定SNMP第3版凭证以解密SNMP第3版数据包。解密的捕获用于分析和检索SNMP OID

在用于snmp-server host配置的接口上配置SNMP数据包捕获：

```
<#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 capsntp interface management buffer 10000000 match udp host 192.168.10.10 host 192.168.5.254 eq
```

```
firepower#
```

```
show capture capsntp
```

```
capture capsnpmp 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...
```

## 关键点

1. SNMP源地址和目的地址/端口。
2. 无法解码SNMP协议PDU，因为Wireshark不知道privKey。
3. encryptedPDU基元的值。

## 推荐的操作

本部分列出的操作旨在进一步缩小问题范围。

行动1.解密SNMP捕获。

保存捕获并编辑Wireshark SNMP协议首选项，以指定用于解密数据包的SNMP第3版凭证。

```
<#root>
```

```
firepower#
```

```
copy /pcap capture: tftp:
```

```
Source capture name [capsnpmp]?
```

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

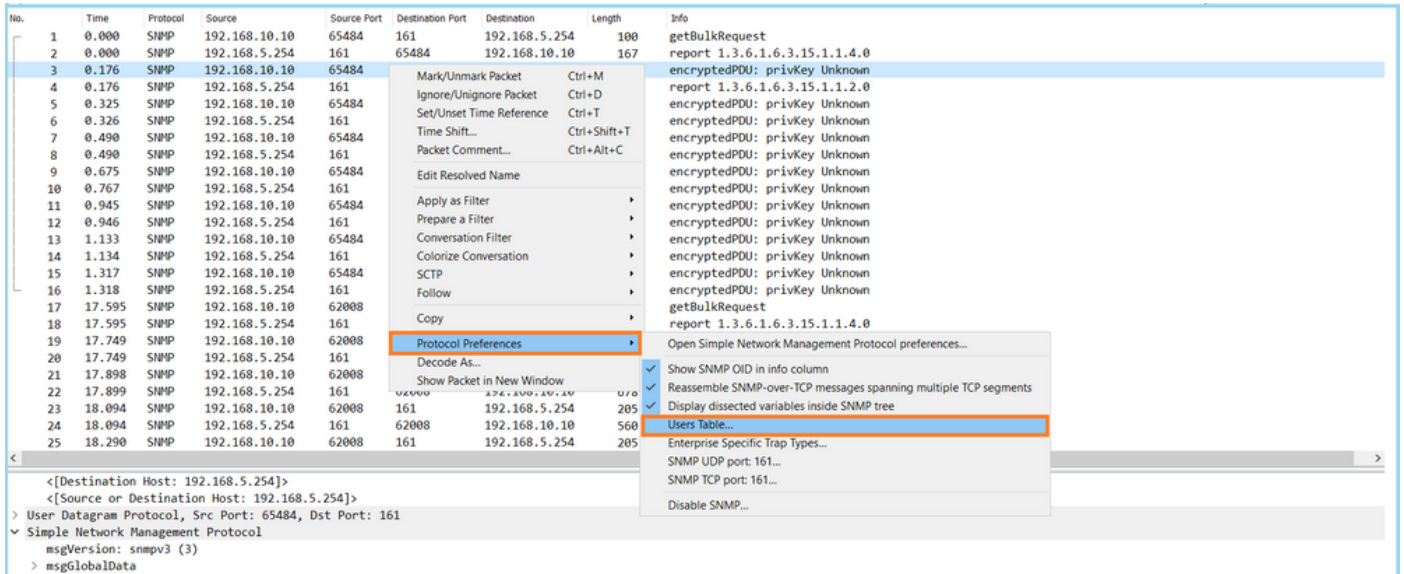


Destination filename [capsnmp]? capsnmp.pcap

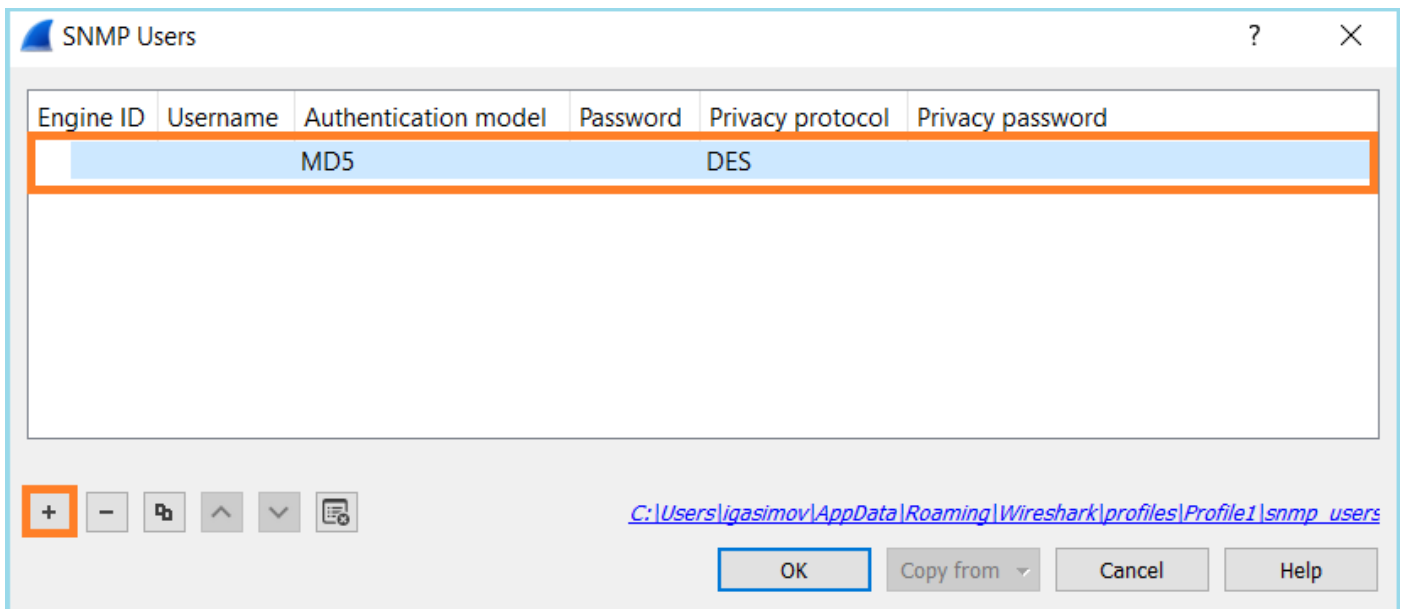
!!!!!!

64 packets copied in 0.40 secs

在Wireshark上打开捕获文件，选择SNMP数据包，然后导航到协议首选项>用户表，如图所示：



在“SNMP用户”(SNMP Users)表中，指定了SNMP第3版用户名、身份验证模型、身份验证密码、隐私协议和隐私密码（实际凭证未显示在下方）：



应用SNMP用户设置后，Wireshark显示解密的SNMP PDU：



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	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	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	get-response 1.3.6.1.4.1.9.9.221.1.1.1.1.2.1.1 1.3.6.1.4.1.9.9.221.1.1.1.1.2.1.2 1.3.6.1.4.1.9.9.221.1.1
7	0.490	SNMP	192.168.10.10	65484	161	192.168.5.254	205	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	get-response 1.3.6.1.4.1.9.9.221.1.1.1.1.5.1.1 1.3.6.1.4.1.9.9.221.1.1.1.1.5.1.2 1.3.6.1.4.1.9.9.221.1.1
9	0.675	SNMP	192.168.10.10	65484	161	192.168.5.254	205	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	get-response 1.3.6.1.4.1.9.9.221.1.1.1.1.7.1.1 1.3.6.1.4.1.9.9.221.1.1.1.1.7.1.2 1.3.6.1.4.1.9.9.221.1.1
11	0.945	SNMP	192.168.10.10	65484	161	192.168.5.254	205	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	get-response 1.3.6.1.4.1.9.9.221.1.1.1.1.17.1.1 1.3.6.1.4.1.9.9.221.1.1.1.1.17.1.2 1.3.6.1.4.1.9.9.221.1.1
13	1.133	SNMP	192.168.10.10	65484	161	192.168.5.254	205	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 1.3.6.1.4.1.9.9.221.1.1.1.1.19.1.2 1.3.6.1.4.1.9.9.221.1.1
15	1.317	SNMP	192.168.10.10	65484	161	192.168.5.254	205	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	get-response 1.3.6.1.4.1.9.9.392.1.1.1.0 1.3.6.1.4.1.9.9.392.1.1.2.0 1.3.6.1.4.1.9.9.392.1.1.3.0 1.3.6.1
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	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	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	get-response 1.3.6.1.4.1.9.9.221.1.1.1.1.2.1.1 1.3.6.1.4.1.9.9.221.1.1.1.1.2.1.2 1.3.6.1.4.1.9.9.221.1.1
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 1.3.6.1.4.1.9.9.221.1.1.1.1.5.1.2 1.3.6.1.4.1.9.9.221.1.1
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

```

msgData: encryptedPDU (1)
  encryptedPDU: 879a16d23633400a0391c5280d226e0cec844d87101ba703...
    Decrypted ScopedPDU: 303b04198000009fec1c6dad4930a00ef1fec2301621a415...
      contextEngineID: 8000009fec1c6dad4930a00ef1fec2301621a4158bfc1f40...
      contextName:
      data: getBulkRequest (5)
        getBulkRequest
          request-id: 5620
          non-repeaters: 0
          max-repetitions: 16
          variable-bindings: 1 item
            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)
  
```

### 关键点

1. SNMP监控工具使用SNMP getBulkRequest查询和遍历父OID 1.3.6.1.4.1.9.9.221.1和相关OID。
2. FTD使用包含与1.3.6.1.4.1.9.9.221.1相关的OID的get-response响应每个getBulkRequest。

行动2.识别SNMP OID。

[SNMP目标导航器](#)显示OID 1.3.6.1.4.1.9.9.221.1属于名为CISCO-ENHANCED-MEMPOOL-MIB的管理信息库(MIB)，如下图所示：

Tools & Resources  
**SNMP Object Navigator**

HOME  
SUPPORT  
TOOLS & RESOURCES  
SNMP Object Navigator

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

Translate | Browse The Object Tree

Related Tools  
[Support Case Manager](#)  
[Cisco Community](#)  
[MIB Locator](#)

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

Enter OID or object name:  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	<a href="#">CISCO-ENHANCED-MEMPOOL-MIB</a> ; - <a href="#">View Supporting Images</a>

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)

要在Wireshark中以人工可读格式显示OID，请执行以下操作：

1. 下载MIB CISCO-ENHANCED-MEMPOOL-MIB及其依赖项，如图所示：

Tools & Resources  
**SNMP Object Navigator**

HOME  
SUPPORT  
TOOLS & RESOURCES  
SNMP Object Navigator

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

Related Tools  
[Support Case Manager](#)  
[Cisco Community](#)  
[MIB Locator](#)

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

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

Tools & Resources  
SNMP Object Navigator

HOME SUPPORT TOOLS & RESOURCES

TRANSLATE/BROWSE SEARCH DOWNLOAD MIBS MIB SUPPORT - SW

Help | Feedback

Related Tools  
Support Case Manager  
Cisco Community  
MIB Locator

**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	<a href="#">Download</a>	<a href="#">Download</a>	<a href="#">View Dependencies</a>
2. SNMPv2-TC	<a href="#">Download</a>	<a href="#">Download</a>	<a href="#">View Dependencies</a>
3. SNMPv2-CONF	Not Required	<a href="#">Download</a>	<a href="#">View Dependencies</a>
4. SNMP-FRAMEWORK-MIB	<a href="#">Download</a>	<a href="#">Download</a>	<a href="#">View Dependencies</a>
5. CISCO-SMI	<a href="#">Download</a>	<a href="#">Download</a>	<a href="#">View Dependencies</a>
6. ENTITY-MIB	<a href="#">Download</a>	<a href="#">Download</a>	<a href="#">View Dependencies</a>
7. HCNUM-TC	<a href="#">Download</a>	<a href="#">Download</a>	<a href="#">View Dependencies</a>
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	<a href="#">Download</a>	<a href="#">Download</a>	

2. 在Wireshark的编辑>首选项>名称解析窗口中，已选中启用OID解析。在SMI ( MIB和PIB路径 ) 窗口中，指定包含已下载MIB的文件夹和SMI ( MIB和PIB模块 )。CISCO-ENHANCED-MEMPOOL-MIB会自动添加到模块列表：

The screenshot shows the Wireshark configuration interface. In the 'Name Resolution' window, the 'Enable OID resolution' checkbox is checked. In the 'SMI Paths' window, the 'Directory path' is set to 'C:/Users/Administrator/Downloads/SNMPMIBS'. In the 'SMI Modules' window, 'CISCO-ENHANCED-MEMPOOL-MIB' is listed in the module name list.

3. 重新启动Wireshark后，OID解析激活：

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.0
3	0.176	SNMP	192.168.10.10	65484	161	192.168.5.254	197	getBulkRequest CISCO-ENHANCED-MEMPOOL-MIB::cempMIBObjects
4	0.176	SNMP	192.168.5.254	161	65484	192.168.10.10	192	report SNMP-USER-BASED-SM-MIB::usmStatsNotInTimeWindows.0
5	0.325	SNMP	192.168.10.10	65484	161	192.168.5.254	199	getBulkRequest CISCO-ENHANCED-MEMPOOL-MIB::cempMIBObjects
6	0.326	SNMP	192.168.5.254	161	65484	192.168.10.10	678	get-response CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolType.1.1 CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolType
7	0.490	SNMP	192.168.10.10	65484	161	192.168.5.254	205	getBulkRequest CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.8
8	0.490	SNMP	192.168.5.254	161	65484	192.168.10.10	560	get-response CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolAlternate.1.1 CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoc
9	0.675	SNMP	192.168.10.10	65484	161	192.168.5.254	205	getBulkRequest CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolValid.1.8
10	0.767	SNMP	192.168.5.254	161	65484	192.168.10.10	610	get-response CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolUsed.1.1 CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolUsed
11	0.945	SNMP	192.168.10.10	65484	161	192.168.5.254	205	getBulkRequest CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolFree.1.8
12	0.946	SNMP	192.168.5.254	161	65484	192.168.10.10	584	get-response CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolUsedOvrflw.1.1 CISCO-ENHANCED-MEMPOOL-MIB::cempMemPc
13	1.133	SNMP	192.168.10.10	65484	161	192.168.5.254	205	getBulkRequest CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolUsed.1.8
14	1.134	SNMP	192.168.5.254	161	65484	192.168.10.10	600	get-response CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolUsed.1.1 CISCO-ENHANCED-MEMPOOL-MIB::cempMemP

```

✓ CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.1 (1.3.6.1.4.1.9.9.221.1.1.1.3.1.1): System memory
Object Name: 1.3.6.1.4.1.9.9.221.1.1.1.3.1.1 (CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.1)
CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName: System memory
✓ CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.2 (1.3.6.1.4.1.9.9.221.1.1.1.3.1.2): System memory
Object Name: 1.3.6.1.4.1.9.9.221.1.1.1.3.1.2 (CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.2)
CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName: System memory
✓ CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.3 (1.3.6.1.4.1.9.9.221.1.1.1.3.1.3): MEMPOOL_MSGLYR
Object Name: 1.3.6.1.4.1.9.9.221.1.1.1.3.1.3 (CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.3)
CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName: MEMPOOL_MSGLYR
✓ CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.4 (1.3.6.1.4.1.9.9.221.1.1.1.3.1.4): MEMPOOL_HEAPCACHE_1
Object Name: 1.3.6.1.4.1.9.9.221.1.1.1.3.1.4 (CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.4)
CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName: MEMPOOL_HEAPCACHE_1
✓ CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.5 (1.3.6.1.4.1.9.9.221.1.1.1.3.1.5): MEMPOOL_HEAPCACHE_0
Object Name: 1.3.6.1.4.1.9.9.221.1.1.1.3.1.5 (CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.5)
CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName: MEMPOOL_HEAPCACHE_0
✓ CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.6 (1.3.6.1.4.1.9.9.221.1.1.1.3.1.6): MEMPOOL_DMA_ALT1
Object Name: 1.3.6.1.4.1.9.9.221.1.1.1.3.1.6 (CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.6)
CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName: MEMPOOL_DMA_ALT1
✓ CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.7 (1.3.6.1.4.1.9.9.221.1.1.1.3.1.7): MEMPOOL_DMA
Object Name: 1.3.6.1.4.1.9.9.221.1.1.1.3.1.7 (CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.7)
CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName: MEMPOOL_DMA
✓ CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.8 (1.3.6.1.4.1.9.9.221.1.1.1.3.1.8): MEMPOOL_GLOBAL_SHARED
Object Name: 1.3.6.1.4.1.9.9.221.1.1.1.3.1.8 (CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName.1.8)
CISCO-ENHANCED-MEMPOOL-MIB::cempMemPoolName: MEMPOOL_GLOBAL_SHARED

```

根据捕获文件的解密输出，SNMP监控工具会定期（10秒间隔）轮询有关FTD上的内存池利用率的数据。如TechNote文章[ASA SNMP Polling for Memory-Related Statistics](#)中所述，使用SNMP轮询全局共享池(GSP)利用率会导致高CPU使用率。在这种情况下，从捕获中可以清楚地看出，作为SNMP getBulkRequest基元的一部分，全局共享池利用率会定期轮询。

为了最大限度地减少SNMP进程导致的CPU占用量，建议遵循本文中提到的SNMP CPU占用量缓解步骤，并避免轮询与GSP相关的OID。如果不对与GSP相关的OID进行SNMP轮询，则不会观察到由SNMP进程导致的CPU占用，并且溢出率会显著降低。

## 相关信息

- [Cisco Firepower管理中心配置指南](#)
- [明确Firepower威胁防御访问控制策略规则操作](#)
- [使用Firepower威胁防御捕获和Packet Tracer](#)
- [了解Wireshark](#)

## 关于此翻译

思科采用人工翻译与机器翻译相结合的方式将此文档翻译成不同语言，希望全球的用户都能通过各自的语言得到支持性的内容。

请注意：即使是最好的机器翻译，其准确度也不及专业翻译人员的水平。

Cisco Systems, Inc. 对于翻译的准确性不承担任何责任，并建议您总是参考英文原始文档（已提供链接）。