TURN IT UP

cisco Live!

#CiscoLive

The bridge to possible

TLS Server Identity Discovery

Cisco Secure Firewall (Threat Defense)

Veronika Klauzova, Cisco NetSec Technical Marketing Engineer

BRKSEC-2106

About Speaker

Name

- Veronika Klauzova

Position:

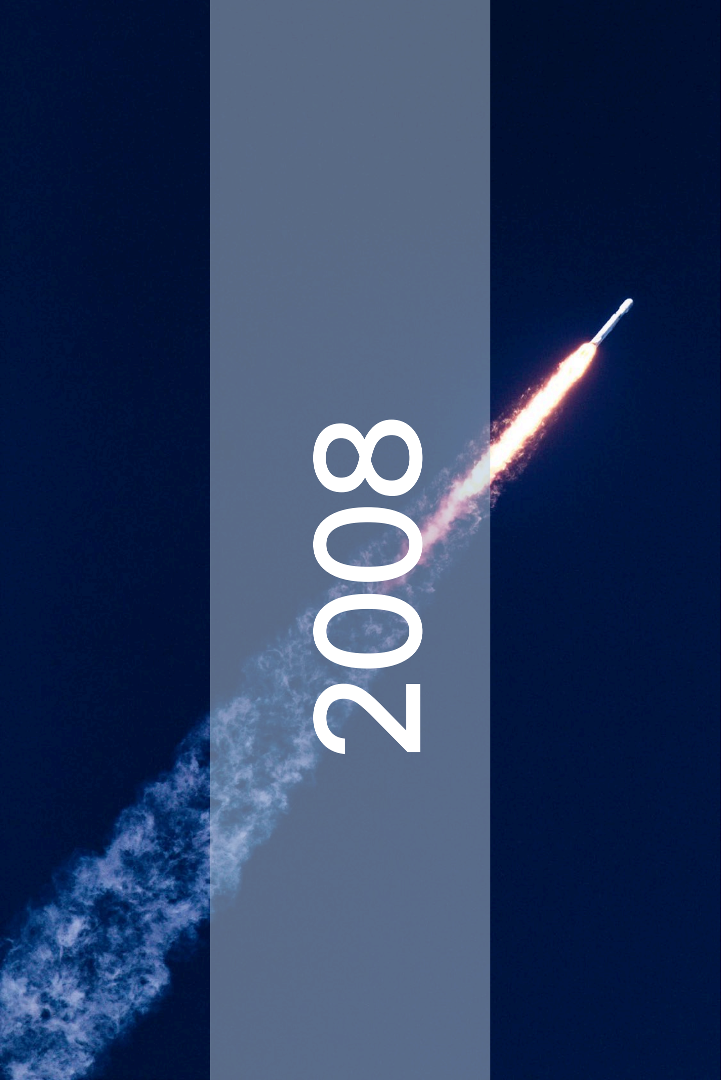
- Technical Marketing Engineer at Cisco Security Business Group
- Cisco Employee since 2013

Free time

- Hiking, Traveling, System Linux administration, Youtube, Books

Agenda

- Introduction
- Feature configuration
- How it works
  - Understanding TLS Server Identity Discovery / Packet-flow details
  - Tracking TLS Probe sessions
- Use-Case Demo
- Closure



Introduction

TLS 1.3 Specification

- □ TLS 1.3 defined in [RFC 8446]
- □ 28 drafts, 10 years
- □ Standardized in August 2018 by IETF
- □ Protects against known attacks in TLS 1.2
- □ Designed to prevent eavesdropping
- □ Protect identities of client & server
- □ **Handshake messages after the ServerHello are encrypted**

Impact on Firewalls

- ❑ **Access control** and **SSL policy enforcement** based on **AppId** or **URL** filtering within a TLS 1.3 connection combined with a spoofed SNI can be circumvented by an intruder.
- ❑ **TLS server certificate** details are **encrypted** in the **TLS 1.3**
  - ❑ This makes the traffic dark to inspect
- ❑ The firewalls lose the ability to acquire a server certificate for TLS sessions in **plain text** to efficiently implement the necessary policies

Application detection and URL filtering

URL filtering and Application detection (AppID) rely on information in the TLS certificates to enforce Acces Control / Firewall rules and/or SSL Policy rules:

Client Hello

- Server Name Indication (SNI)

Server Certificate

- Common Name (CN)
- Subject Alternative Names (SANs)
- Organizational Unit (OU)

Clear text in TLS 1.2

Application detection and URL filtering

URL filtering and Application detection (AppID) rely on information in the TLS certificates to enforce Acces Control / Firewall rules and/or SSL Policy rules:

Client Hello

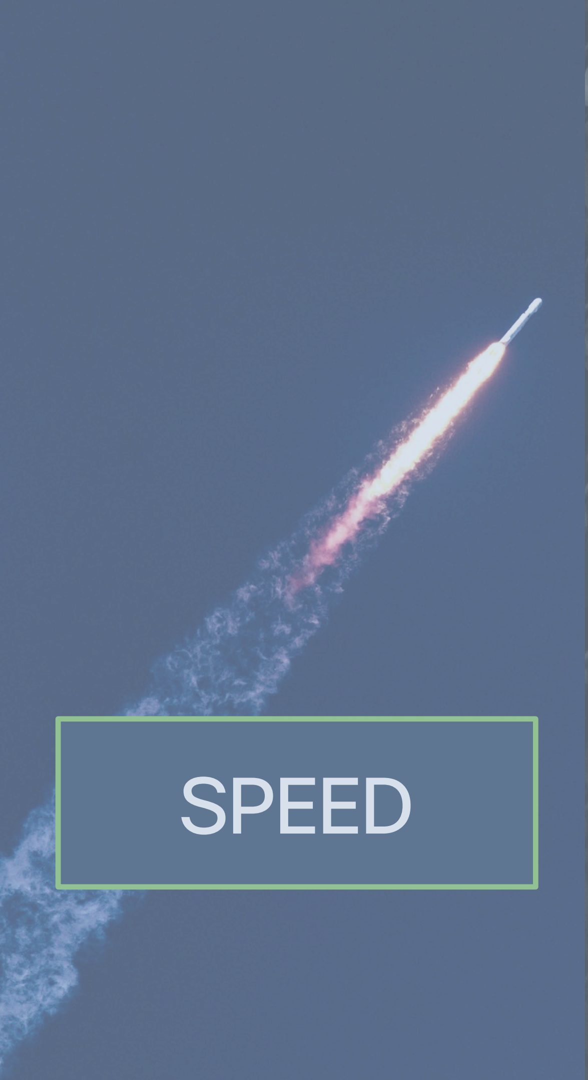
- Server Name Indication (SNI)

Clear text in TLS 1.3

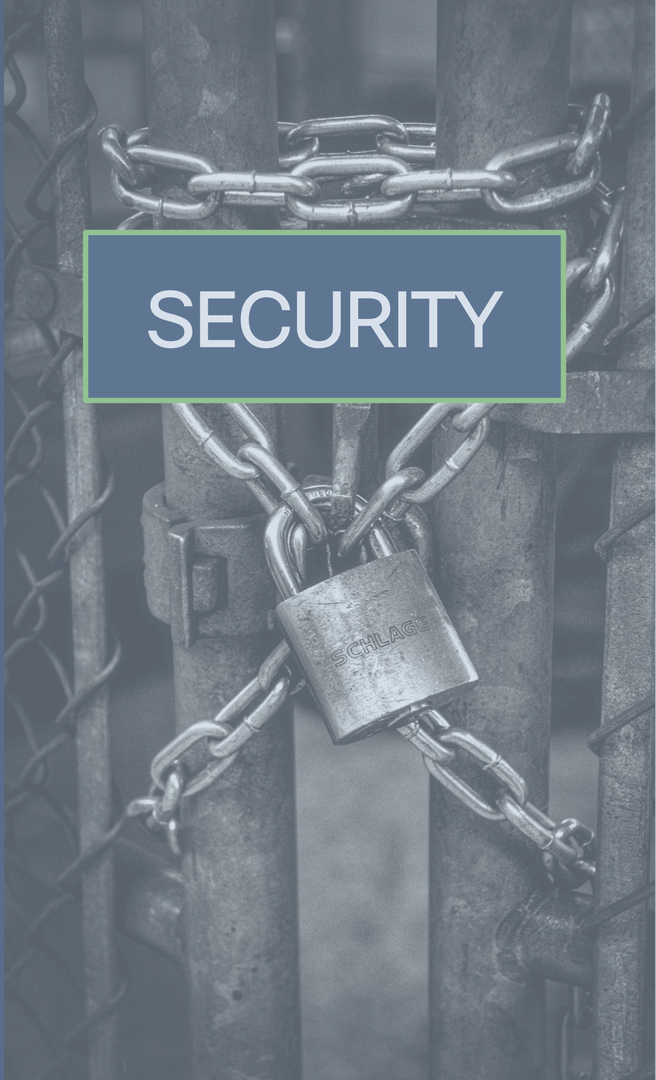
Server Certificate

- Common Name (CN)
- Subject Alternative Names (SANs)
- Organizational Unit (OU)

Encrypted in TLS 1.3



SPEED



SECURITY

Privacy, simplified.
DuckDuckGo
The search engine that doesn't track you. Help Spread DuckDuckGo!

PRIVACY

Understanding TLS Server Identity Discovery

Why?

Effective Security Policy Evaluation relies on Server Name Indication and Server Certificate

TLS 1.3 Encrypts TLS Certificates

TLS Client Hello Server Name Indicated can be Spoofed

What?

Server Identity Discovery makes Server Certificate Information available without performing decryption

Benefits

More effective and reliable match for the TLS policy evaluation

Detect and Block SNI Spoofing

Enhanced control and increased visibility into encrypted flows

TLS Server Identity Discovery Configuration

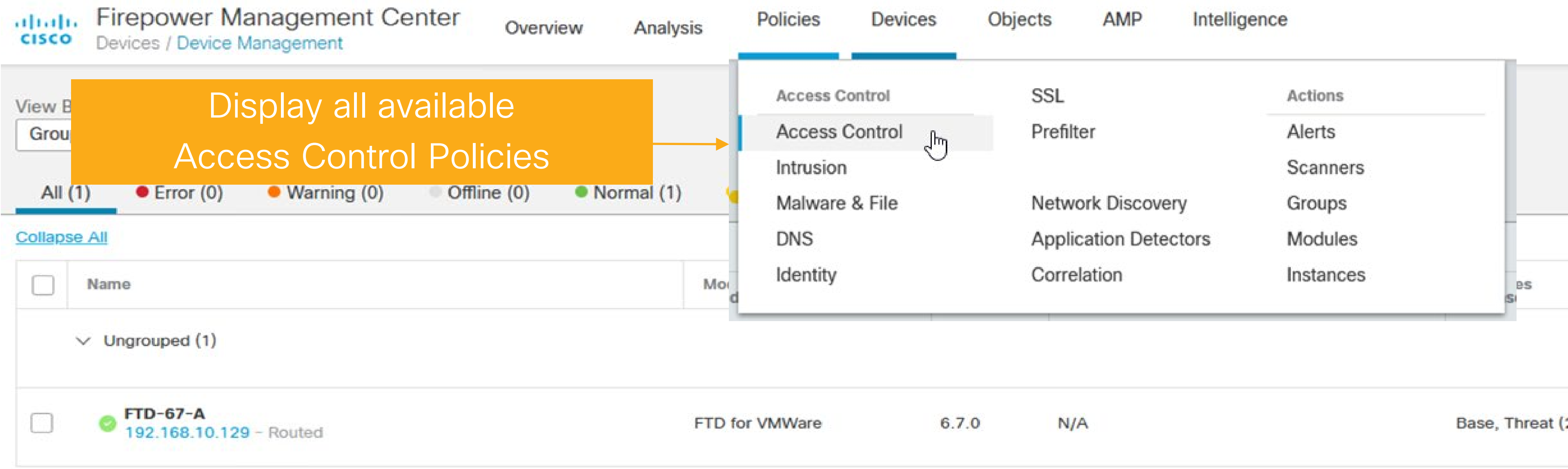
TLS Server Identity Discovery: Deployments

Mode	Type	Snort 2	Snort 3
Routed	Standalone	FMC	FDM
Transparent	High-Availability	FDM	
Inline Set	Cluster		

- Cisco Threat Defense 6.7+
- Compatible with VRF feature

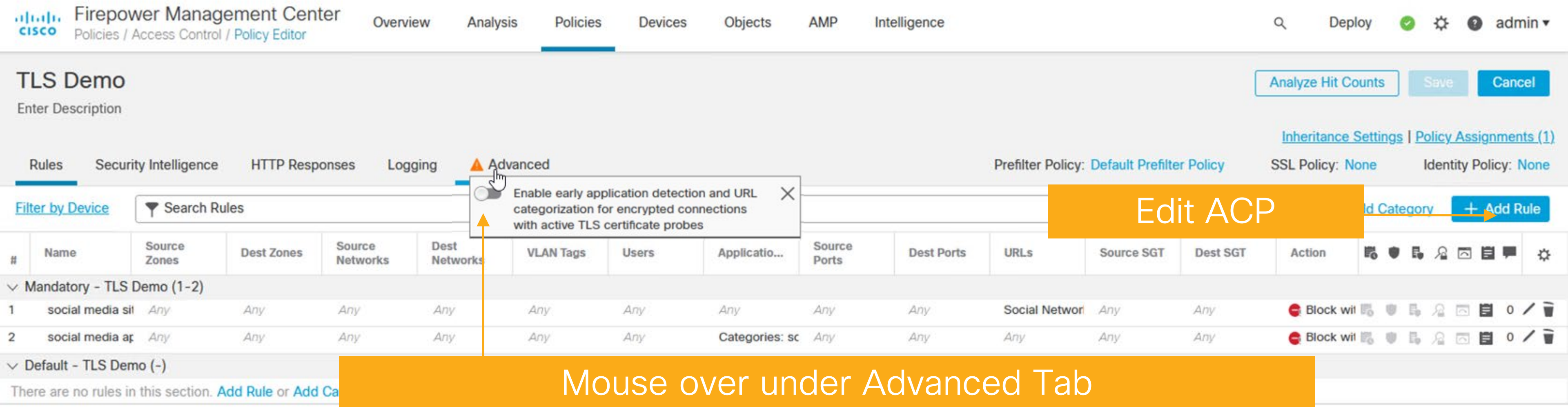
- Disabled by default
- Works with/without SSL Policy
- Does not require any special license

TLS Server Identity Discovery



Display all available Access Control Policies

TLS Server Identity Discovery


Firepower Management Center
Policies / Access Control / Policy Editor
Overview
Analysis
Policies
Devices
Objects
AMP
Intelligence
Deploy
admin

TLS Demo

Enter Description

Prefilter Policy: Default Prefilter Policy

SSL Policy: None

Identity Policy: None

Enable early application detection and URL categorization for encrypted connections with active TLS certificate probes

#	Name	Source Zones	Dest Zones	Source Networks	Dest Networks	VLAN Tags	Users	Applicatio...	Source Ports	Dest Ports	URLs	Source SGT	Dest SGT	Action	
Mandatory - TLS Demo (1-2)															
1	social media sit	Any	Any	Any	Any	Any	Any	Any	Any	Any	Social Networ	Any	Any	Block wit	0
2	social media ap	Any	Any	Any	Any	Any	Any	Categories: sc	Any	Any	Any	Any	Any	Block wit	0
Default - TLS Demo (-)															

There are no rules in this section. Add Rule or Add Ca

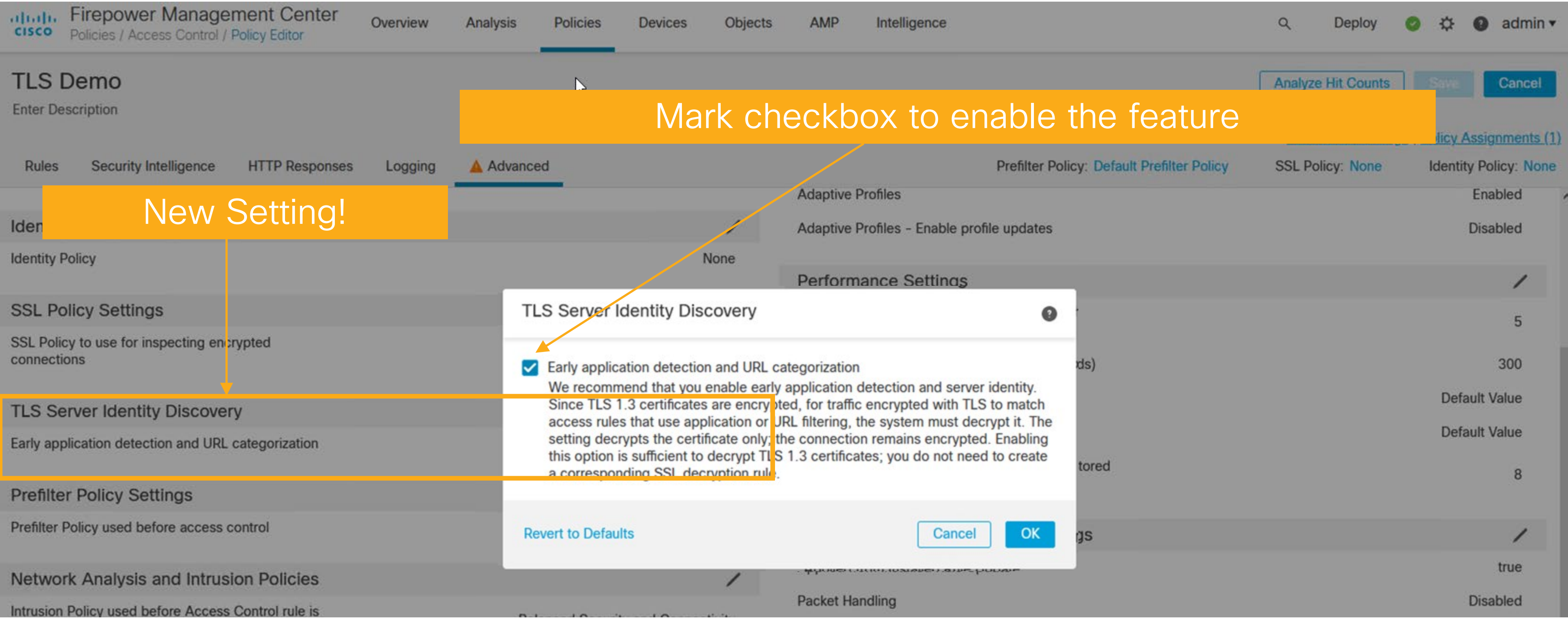
Edit ACP

Mouse over under Advanced Tab

–

We can enable TLS Server Identity Discovery here or continue furher by clicking on Advanced tab

TLS Server Identity Discovery



Mark checkbox to enable the feature

New Setting!

TLS Demo

Enter Description

Prefilter Policy: Default Prefilter Policy SSL Policy: None Identity Policy: None

Identity Policy: None

SSL Policy Settings

SSL Policy to use for inspecting encrypted connections

TLS Server Identity Discovery

Early application detection and URL categorization

Prefilter Policy Settings

Prefilter Policy used before access control

Network Analysis and Intrusion Policies

Intrusion Policy used before Access Control rule is

Adaptive Profiles: Enabled

Adaptive Profiles - Enable profile updates: Disabled

Performance Settings

5

300

Default Value

Default Value

8

true

Packet Handling: Disabled

TLS Server Identity Discovery

☑ Early application detection and URL categorization

We recommend that you enable early application detection and server identity. Since TLS 1.3 certificates are encrypted, for traffic encrypted with TLS to match access rules that use application or URL filtering, the system must decrypt it. The setting decrypts the certificate only; the connection remains encrypted. Enabling this option is sufficient to decrypt TLS 1.3 certificates; you do not need to create a corresponding SSL decryption rule.

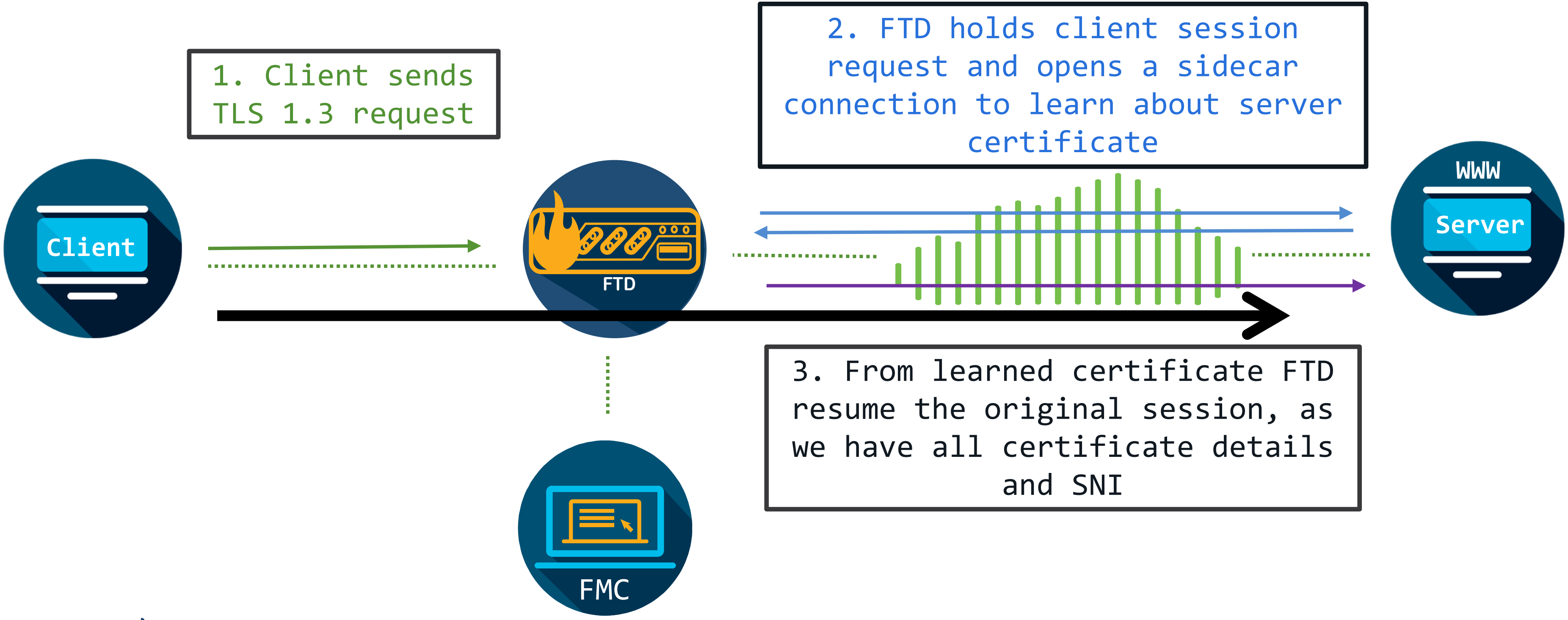
Revert to Defaults Cancel OK

How it works with Packet Flow Details

–

TLS Server Identity Discovery

High Level Overview
How does TLS probe works?

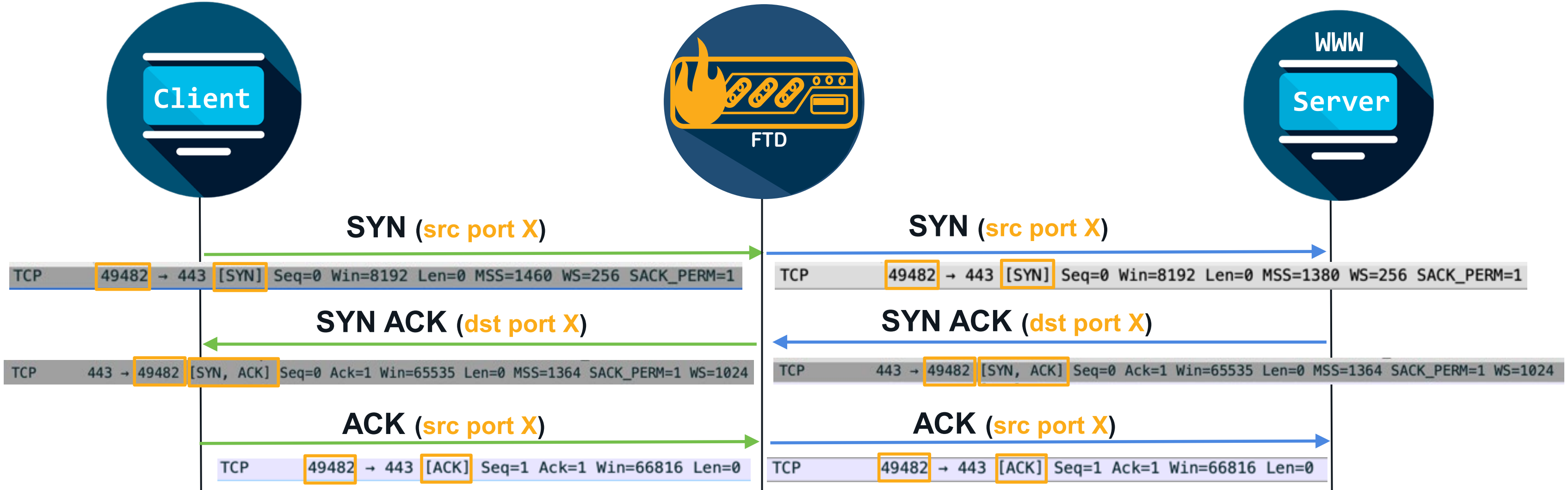


1. Client sends TLS 1.3 request

2. FTD holds client session request and opens a sidecar connection to learn about server certificate

3. From learned certificate FTD resume the original session, as we have all certificate details and SNI

Packet Flow Diagram



SYN (src port X)

```
TCP        49482 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
```

SYN (src port X)

```
TCP        49482 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1380 WS=256 SACK_PERM=1
```

SYN ACK (dst port X)

```
TCP        443 → 49482 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1364 SACK_PERM=1 WS=1024
```

SYN ACK (dst port X)

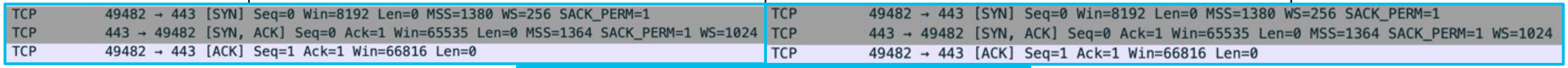
```
TCP        443 → 49482 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1364 SACK_PERM=1 WS=1024
```

ACK (src port X)

```
TCP        49482 → 443 [ACK] Seq=1 Ack=1 Win=66816 Len=0
```

ACK (src port X)

```
TCP        49482 → 443 [ACK] Seq=1 Ack=1 Win=66816 Len=0
```

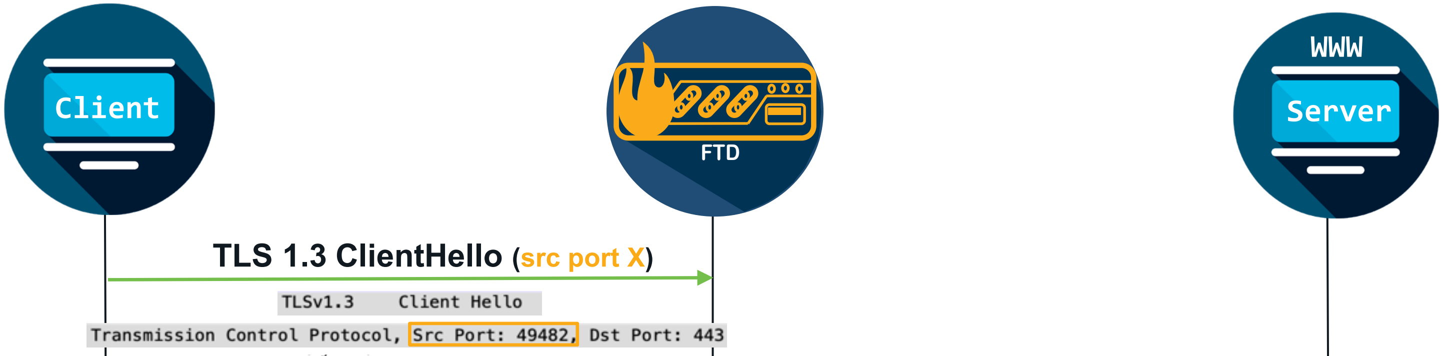


```
TCP        49482 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1380 WS=256 SACK_PERM=1
TCP        443 → 49482 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1364 SACK_PERM=1 WS=1024
TCP        49482 → 443 [ACK] Seq=1 Ack=1 Win=66816 Len=0
```

```
TCP        49482 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1380 WS=256 SACK_PERM=1
TCP        443 → 49482 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1364 SACK_PERM=1 WS=1024
TCP        49482 → 443 [ACK] Seq=1 Ack=1 Win=66816 Len=0
```

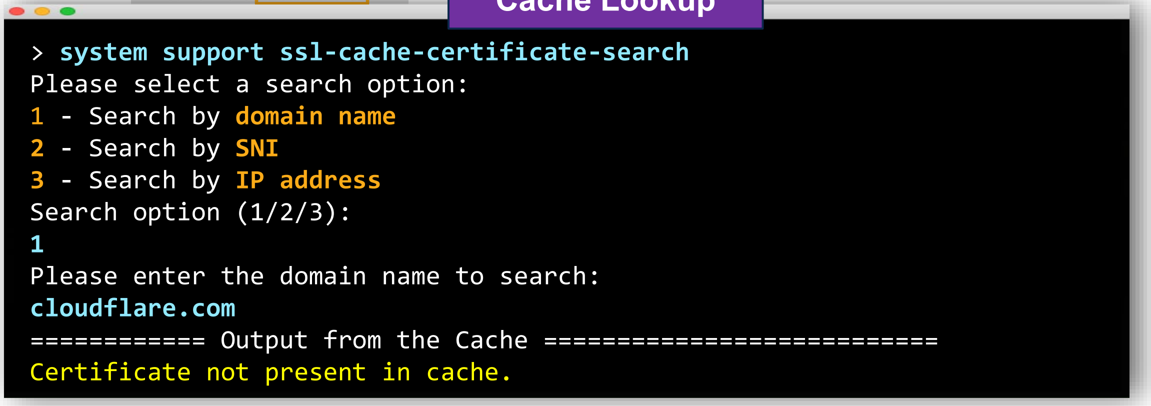
TCP 3-way handshake

Packet Flow Diagram



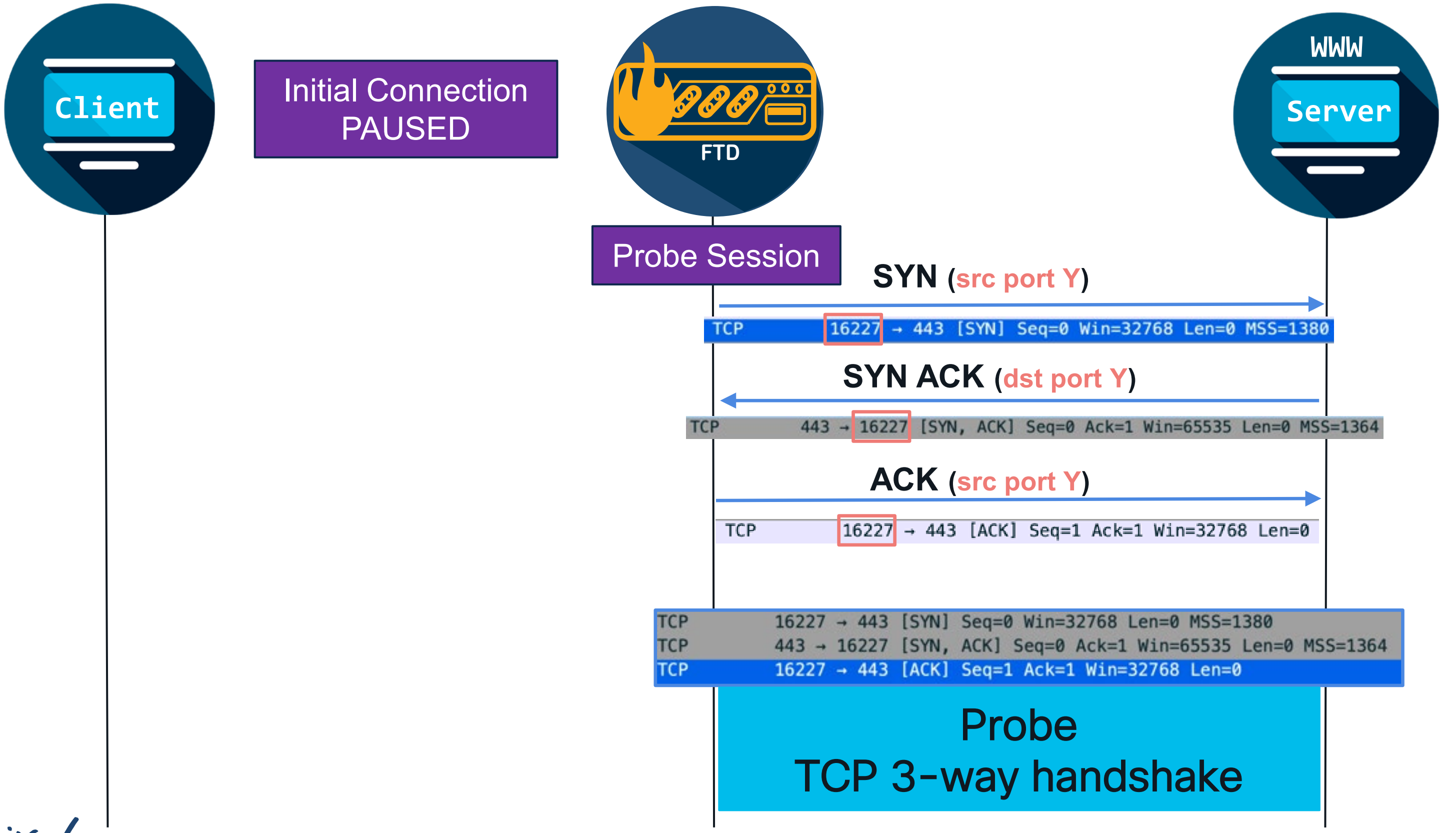
```
TLSv1.3      Client Hello
Transmission Control Protocol, Src Port: 49482, Dst Port: 443
  ▾ Server Name Indication extension
      Server Name list length: 17
      Server Name Type: host_name (0)
      Server Name length: 14
      Server Name: cloudflare.com
```

Server Certificate Cache Lookup



```
> system support ssl-cache-certificate-search
Please select a search option:
1 - Search by domain name
2 - Search by SNI
3 - Search by IP address
Search option (1/2/3):
1
Please enter the domain name to search:
cloudflare.com
============ Output from the Cache ============================
Certificate not present in cache.
```

Packet Flow Diagram



```
TCP        16227 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1380
```

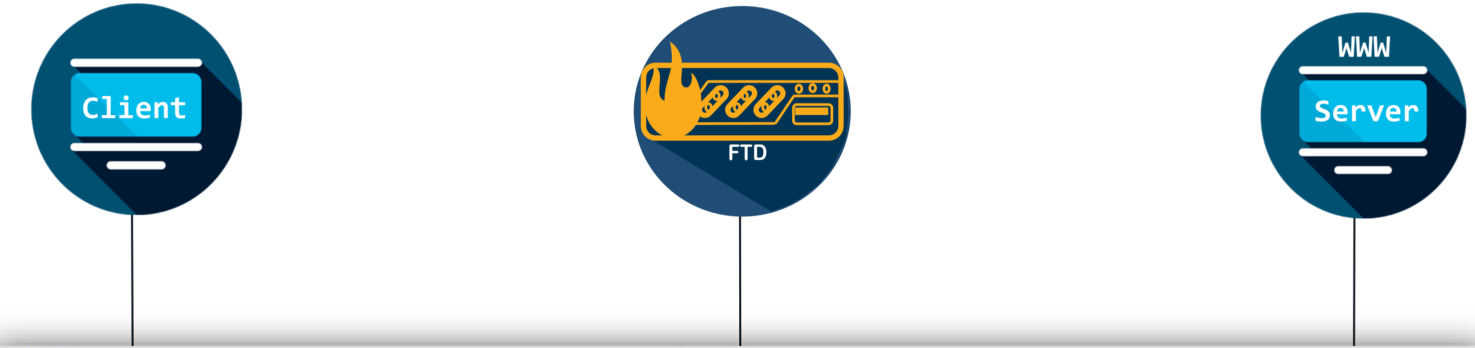
```
TCP        443 → 16227 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1364
```

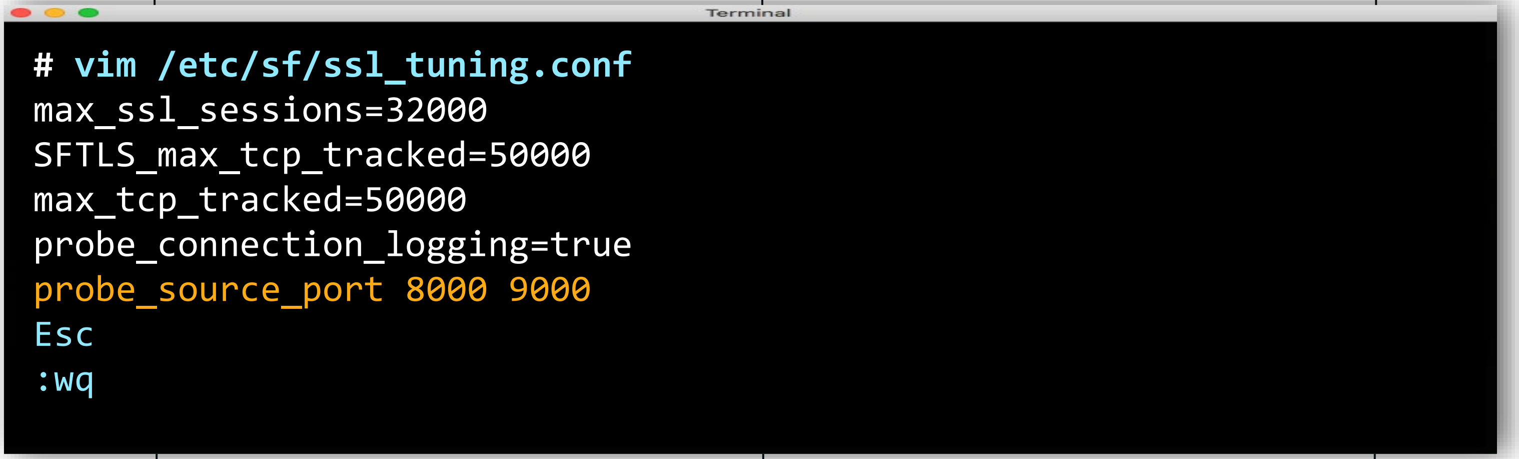
```
TCP        16227 → 443 [ACK] Seq=1 Ack=1 Win=32768 Len=0
```

```
TCP        16227 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1380
TCP        443 → 16227 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1364
TCP        16227 → 443 [ACK] Seq=1 Ack=1 Win=32768 Len=0
```

Changing Probe Source Port

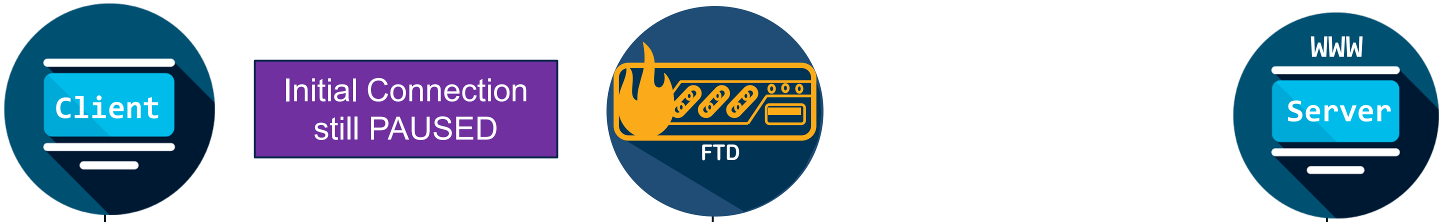
Packet Flow Diagram





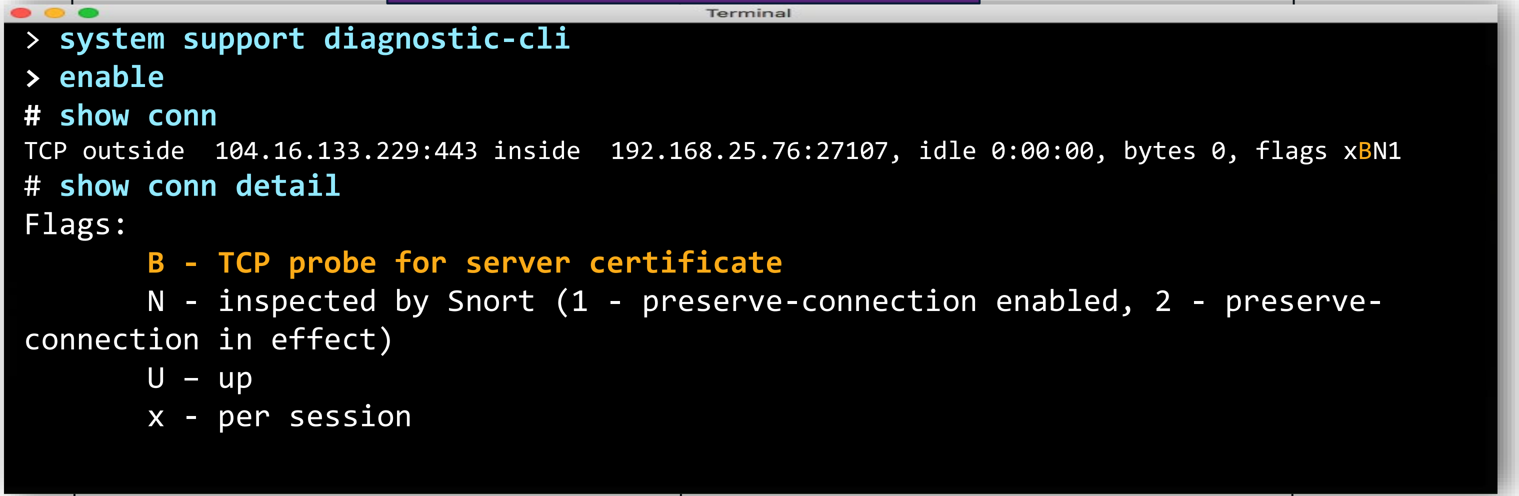
```
# vim /etc/sf/ssl_tuning.conf
max_ssl_sessions=32000
SFTLS_max_tcp_tracked=50000
max_tcp_tracked=50000
probe_connection_logging=true
probe_source_port 8000 9000
Esc
:wq
```

Packet Flow Diagram



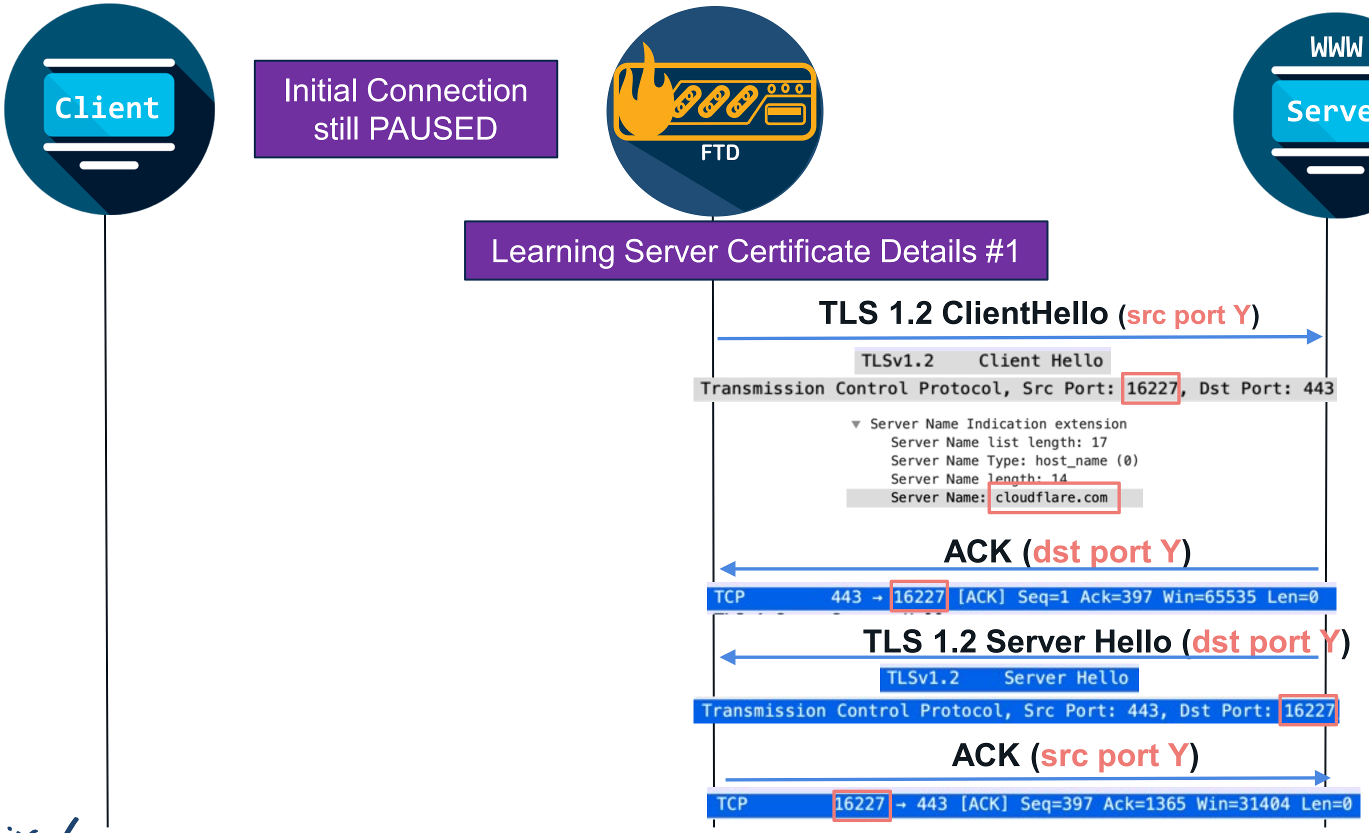
Initial Connection still PAUSED

Probe Session Connection Tracking



```
> system support diagnostic-cli
> enable
# show conn
TCP outside  104.16.133.229:443 inside  192.168.25.76:27107, idle 0:00:00, bytes 0, flags xBN1
# show conn detail
Flags:
       B - TCP probe for server certificate
       N - inspected by Snort (1 - preserve-connection enabled, 2 - preserve-
connection in effect)
       U – up
       x - per session
```

Packet Flow Diagram



Initial Connection still PAUSED

Learning Server Certificate Details #1

TLS 1.2 ClientHello (src port Y)

```
TLSv1.2     Client Hello
Transmission Control Protocol, Src Port: 16227, Dst Port: 443

▼ Server Name Indication extension
    Server Name list length: 17
    Server Name Type: host_name (0)
    Server Name length: 14
    Server Name: cloudflare.com
```

ACK (dst port Y)

```
TCP        443 → 16227 [ACK] Seq=1 Ack=397 Win=65535 Len=0
```

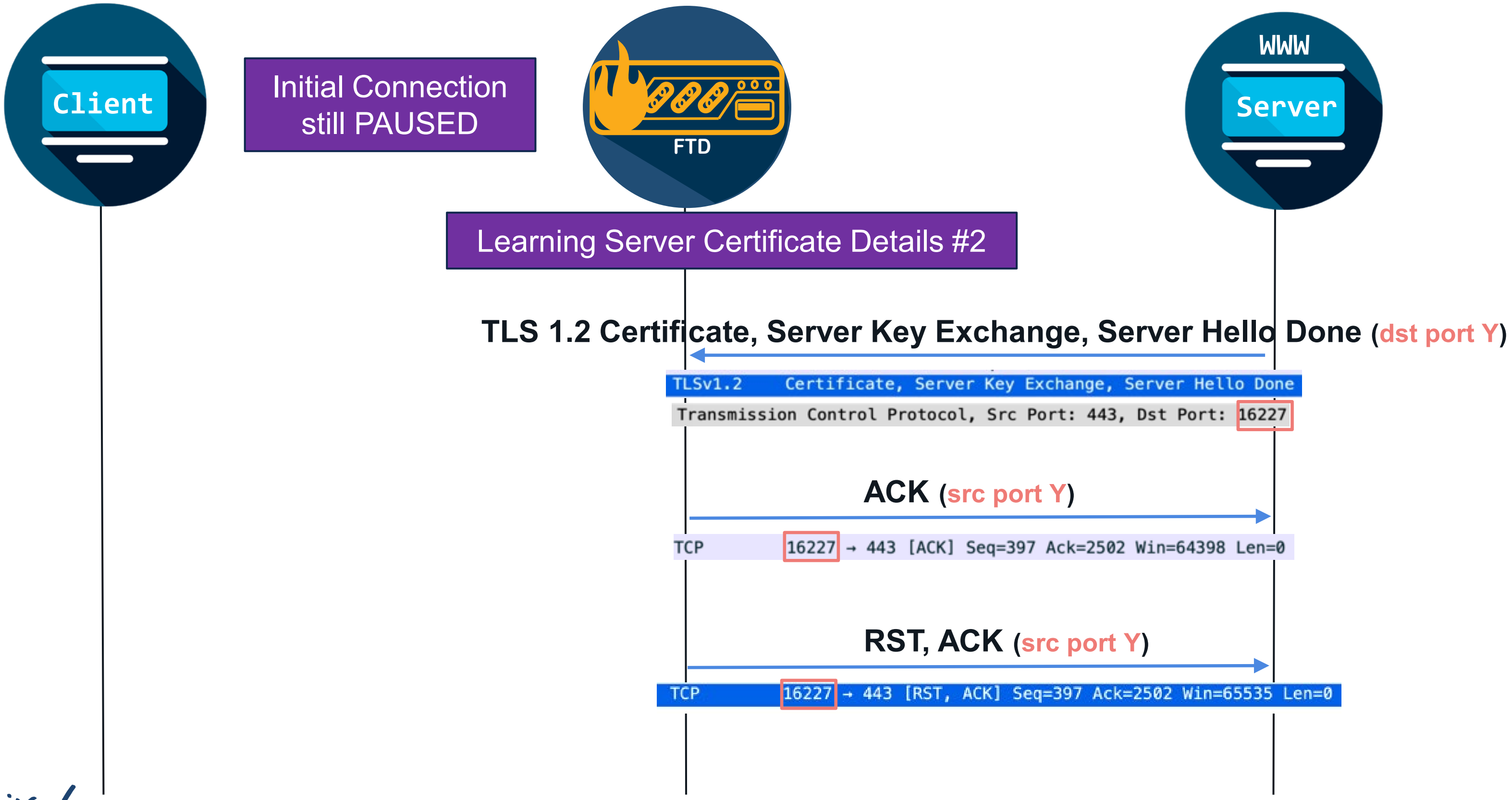
TLS 1.2 Server Hello (dst port Y)

```
TLSv1.2     Server Hello
Transmission Control Protocol, Src Port: 443, Dst Port: 16227
```

ACK (src port Y)

```
TCP        16227 → 443 [ACK] Seq=397 Ack=1365 Win=31404 Len=0
```

Packet Flow Diagram



Initial Connection still PAUSED

Learning Server Certificate Details #2

TLS 1.2 Certificate, Server Key Exchange, Server Hello Done (dst port Y)

```
TLSv1.2     Certificate, Server Key Exchange, Server Hello Done
Transmission Control Protocol, Src Port: 443, Dst Port: 16227
```

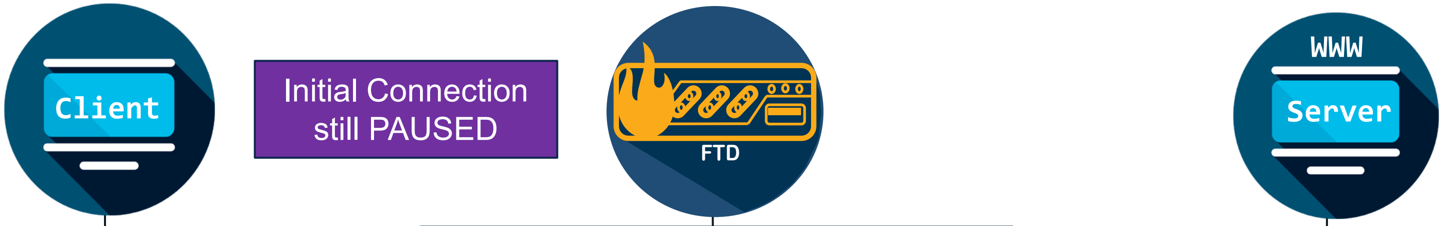
ACK (src port Y)

```
TCP         16227 → 443 [ACK] Seq=397 Ack=2502 Win=64398 Len=0
```

RST, ACK (src port Y)

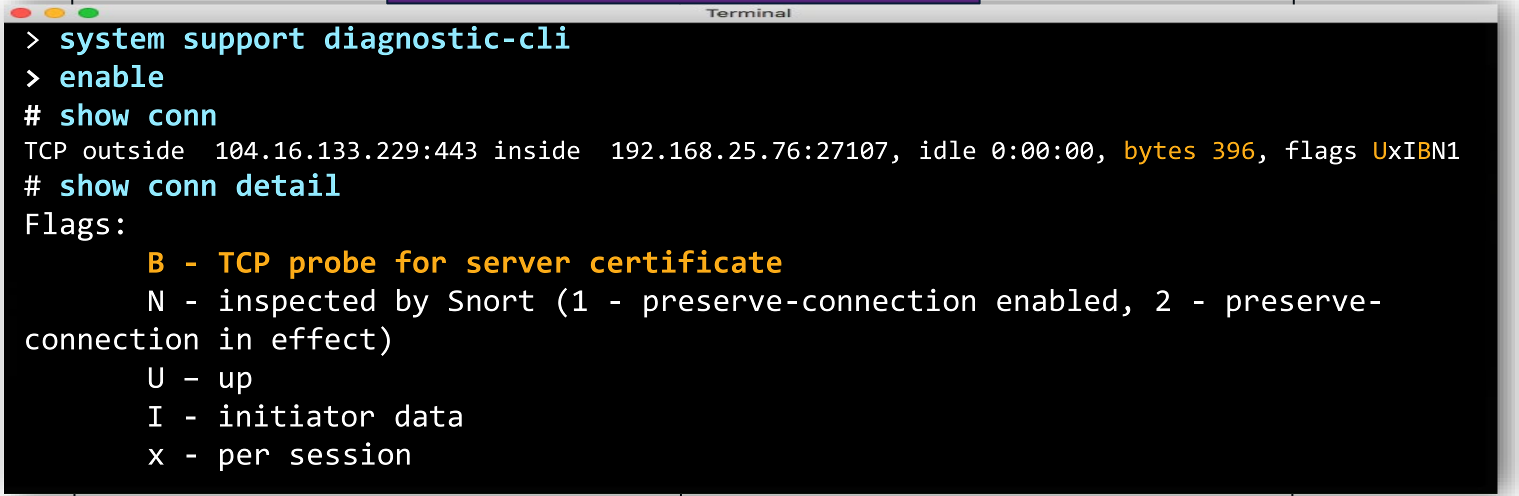
```
TCP         16227 → 443 [RST, ACK] Seq=397 Ack=2502 Win=65535 Len=0
```

Packet Flow Diagram



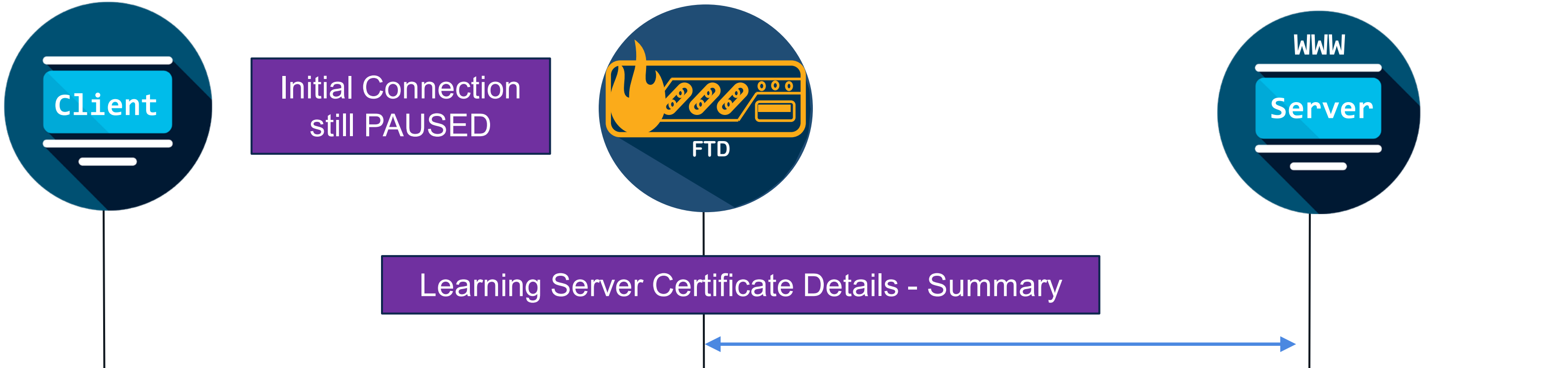
Initial Connection still PAUSED

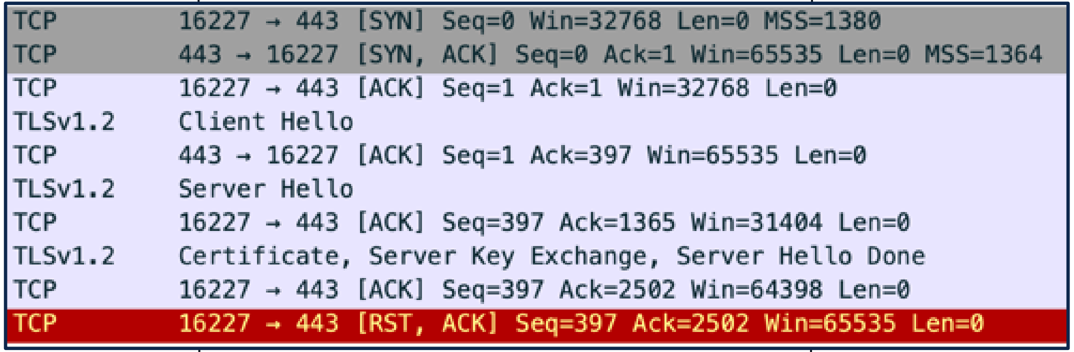
Probe Session Connection Tracking



```
> system support diagnostic-cli
> enable
# show conn
TCP outside  104.16.133.229:443 inside  192.168.25.76:27107, idle 0:00:00, bytes 396, flags UxIBN1
# show conn detail
Flags:
       B - TCP probe for server certificate
       N - inspected by Snort (1 - preserve-connection enabled, 2 - preserve-
connection in effect)
       U – up
       I - initiator data
       x - per session
```

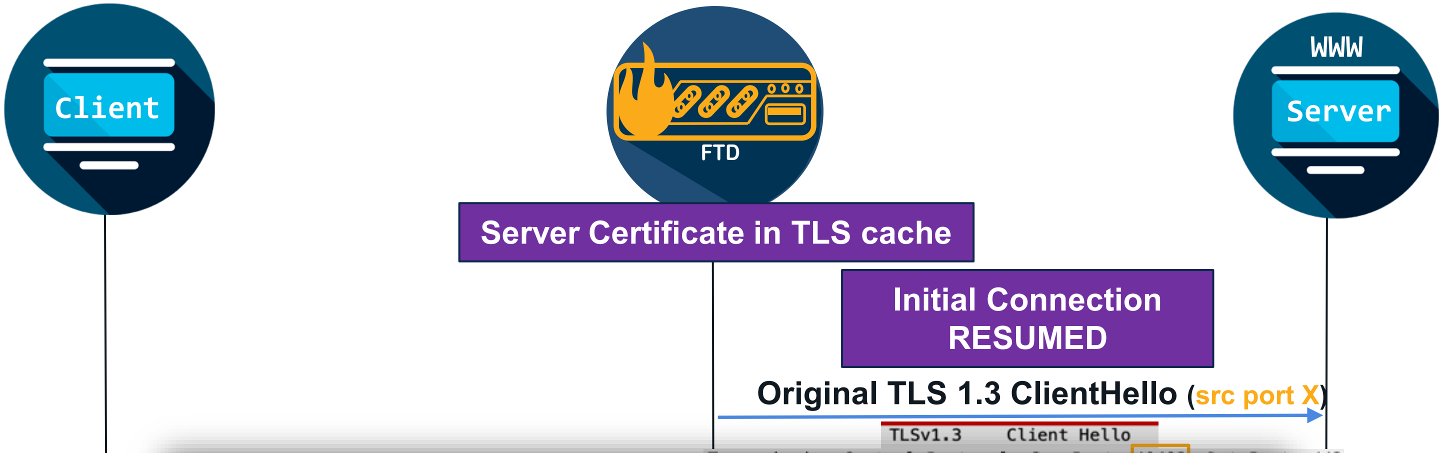
Packet Flow Diagram


Client
Initial Connection still PAUSED
FTD
WWW
Server
Learning Server Certificate Details - Summary



```
TCP        16227 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1380
TCP        443 → 16227 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1364
TCP        16227 → 443 [ACK] Seq=1 Ack=1 Win=32768 Len=0
TLSv1.2    Client Hello
TCP        443 → 16227 [ACK] Seq=1 Ack=397 Win=65535 Len=0
TLSv1.2    Server Hello
TCP        16227 → 443 [ACK] Seq=397 Ack=1365 Win=31404 Len=0
TLSv1.2    Certificate, Server Key Exchange, Server Hello Done
TCP        16227 → 443 [ACK] Seq=397 Ack=2502 Win=64398 Len=0
TCP        16227 → 443 [RST, ACK] Seq=397 Ack=2502 Win=65535 Len=0
```

Packet Flow Diagram

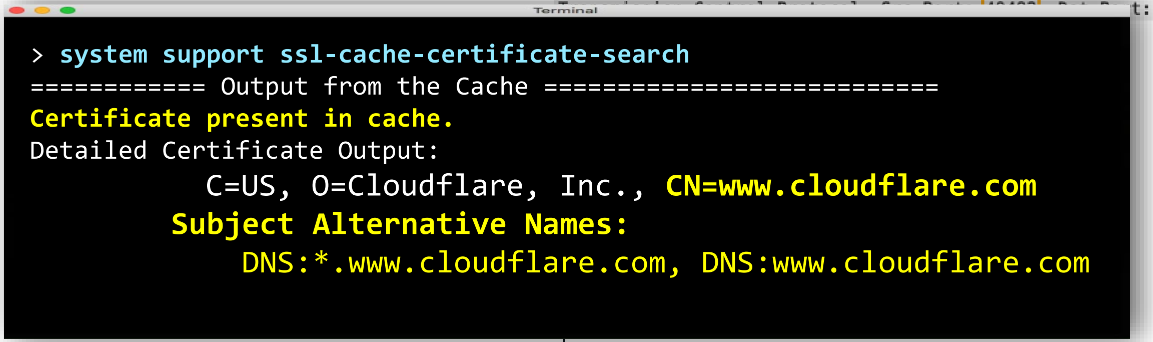


Server Certificate in TLS cache

Initial Connection RESUMED

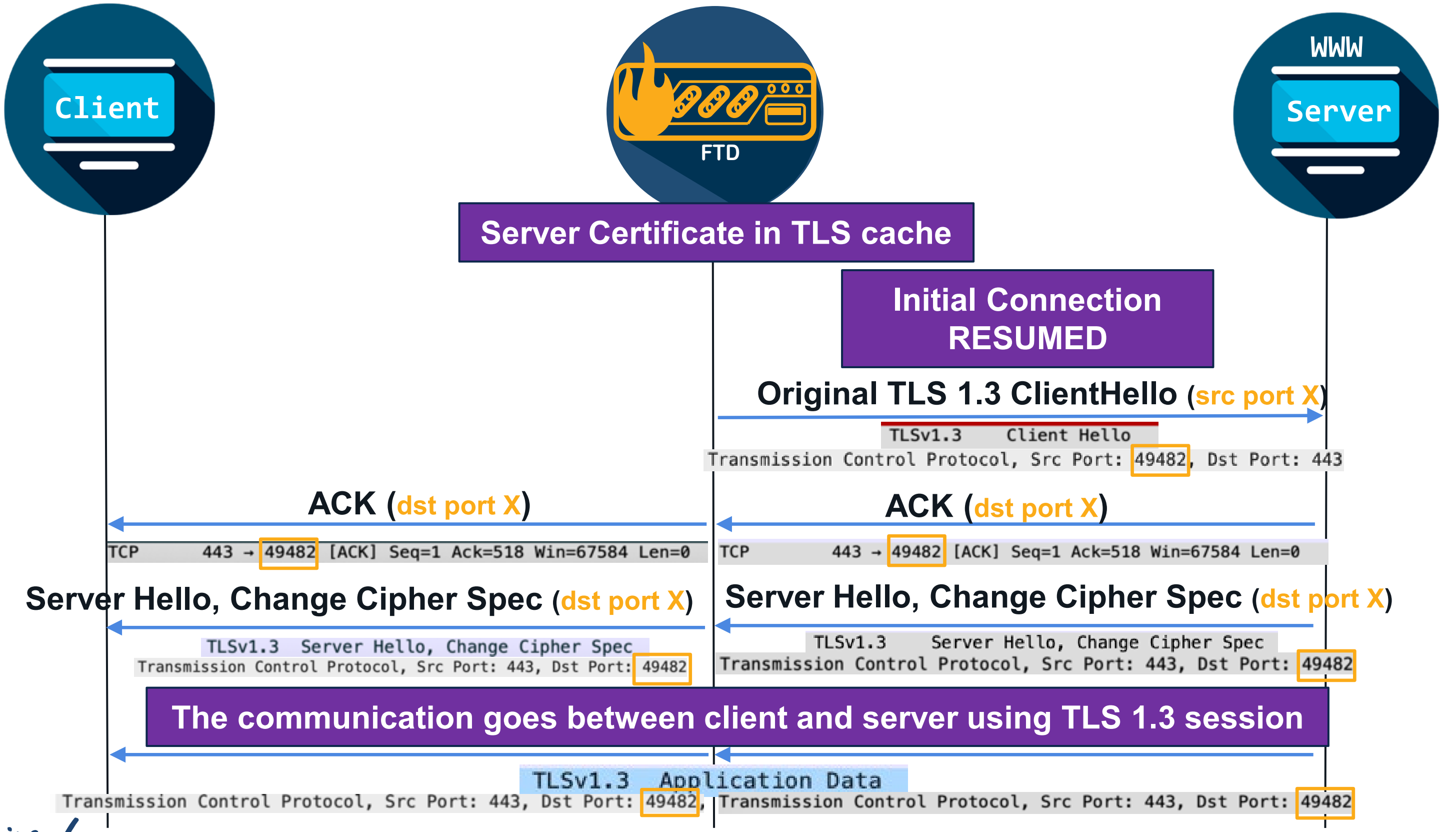
Original TLS 1.3 ClientHello (src port X)

```
TLSv1.3     Client Hello
                                        t: 443
```



```
> system support ssl-cache-certificate-search
============ Output from the Cache ============================
Certificate present in cache.
Detailed Certificate Output:
            C=US, O=Cloudflare, Inc., CN=www.cloudflare.com
          Subject Alternative Names:
              DNS:*.www.cloudflare.com, DNS:www.cloudflare.com
```

Packet Flow Diagram



Server Certificate in TLS cache

Initial Connection RESUMED

Original TLS 1.3 ClientHello (src port X)

```
TLSv1.3      Client Hello
Transmission Control Protocol, Src Port: 49482, Dst Port: 443
```

ACK (dst port X)

```
TCP        443 → 49482 [ACK] Seq=1 Ack=518 Win=67584 Len=0
```

ACK (dst port X)

```
TCP         443 → 49482 [ACK] Seq=1 Ack=518 Win=67584 Len=0
```

Server Hello, Change Cipher Spec (dst port X)

```
TLSv1.3  Server Hello, Change Cipher Spec
Transmission Control Protocol, Src Port: 443, Dst Port: 49482
```

Server Hello, Change Cipher Spec (dst port X)

```
TLSv1.3      Server Hello, Change Cipher Spec
Transmission Control Protocol, Src Port: 443, Dst Port: 49482
```

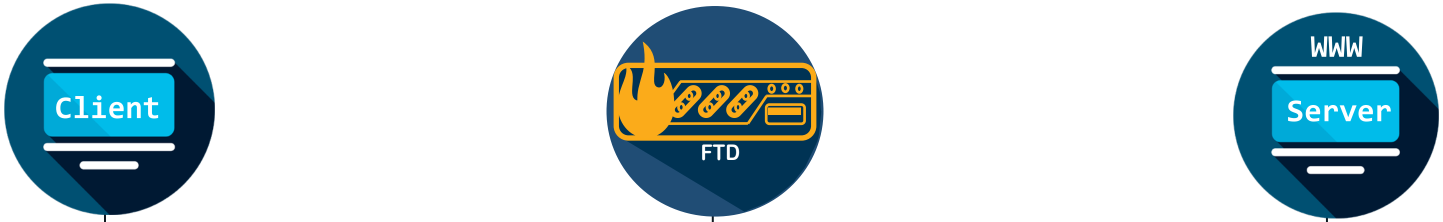
The communication goes between client and server using TLS 1.3 session

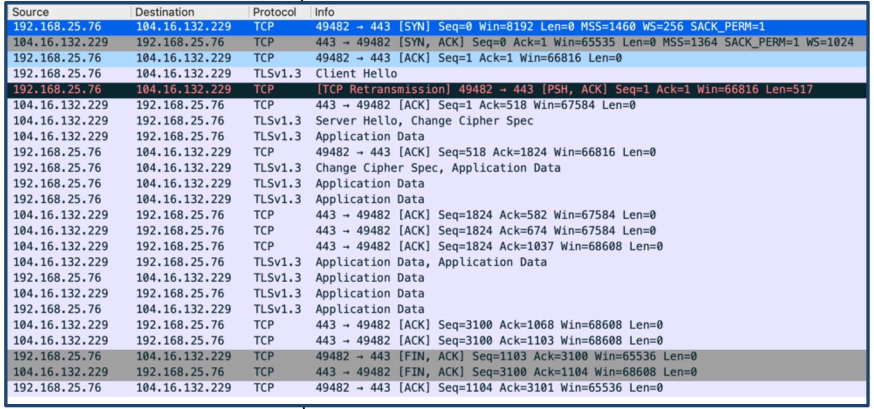
```
TLSv1.3  Application Data
Transmission Control Protocol, Src Port: 443, Dst Port: 49482, Transmission Control Protocol, Src Port: 443, Dst Port: 49482
```

Packet Flow Diagram

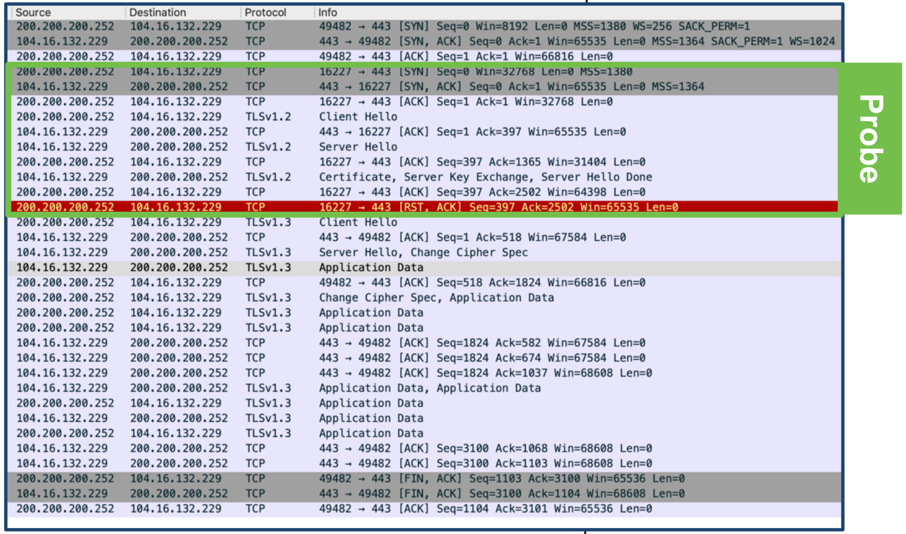
Summary

Certificate not found in the TLS Cache





Source	Destination	Protocol	Info
192.168.25.76	104.16.132.229	TCP	49482 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
104.16.132.229	192.168.25.76	TCP	443 → 49482 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1364 SACK_PERM=1 WS=1024
192.168.25.76	104.16.132.229	TCP	49482 → 443 [ACK] Seq=1 Ack=1 Win=66816 Len=0
192.168.25.76	104.16.132.229	TLSv1.3	Client Hello
192.168.25.76	104.16.132.229	TCP	[TCP Retransmission] 49482 → 443 [PSH, ACK] Seq=1 Ack=1 Win=66816 Len=517
104.16.132.229	192.168.25.76	TCP	443 → 49482 [ACK] Seq=1 Ack=518 Win=67584 Len=0
104.16.132.229	192.168.25.76	TLSv1.3	Server Hello, Change Cipher Spec
104.16.132.229	192.168.25.76	TLSv1.3	Application Data
192.168.25.76	104.16.132.229	TCP	49482 → 443 [ACK] Seq=518 Ack=1824 Win=66816 Len=0
192.168.25.76	104.16.132.229	TLSv1.3	Change Cipher Spec, Application Data
192.168.25.76	104.16.132.229	TLSv1.3	Application Data
192.168.25.76	104.16.132.229	TLSv1.3	Application Data
104.16.132.229	192.168.25.76	TCP	443 → 49482 [ACK] Seq=1824 Ack=582 Win=67584 Len=0
104.16.132.229	192.168.25.76	TCP	443 → 49482 [ACK] Seq=1824 Ack=674 Win=67584 Len=0
104.16.132.229	192.168.25.76	TCP	443 → 49482 [ACK] Seq=1824 Ack=1037 Win=68608 Len=0
104.16.132.229	192.168.25.76	TLSv1.3	Application Data, Application Data
192.168.25.76	104.16.132.229	TLSv1.3	Application Data
104.16.132.229	192.168.25.76	TLSv1.3	Application Data
192.168.25.76	104.16.132.229	TLSv1.3	Application Data
104.16.132.229	192.168.25.76	TCP	443 → 49482 [ACK] Seq=3100 Ack=1068 Win=68608 Len=0
104.16.132.229	192.168.25.76	TCP	443 → 49482 [ACK] Seq=3100 Ack=1103 Win=68608 Len=0
192.168.25.76	104.16.132.229	TCP	49482 → 443 [FIN, ACK] Seq=1103 Ack=3100 Win=65536 Len=0
104.16.132.229	192.168.25.76	TCP	443 → 49482 [FIN, ACK] Seq=3100 Ack=1104 Win=68608 Len=0
192.168.25.76	104.16.132.229	TCP	49482 → 443 [ACK] Seq=1104 Ack=3101 Win=65536 Len=0



Source	Destination	Protocol	Info
200.200.200.252	104.16.132.229	TCP	49482 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1380 WS=256 SACK_PERM=1
104.16.132.229	200.200.200.252	TCP	443 → 49482 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1364 SACK_PERM=1 WS=1024
200.200.200.252	104.16.132.229	TCP	49482 → 443 [ACK] Seq=1 Ack=1 Win=66816 Len=0
200.200.200.252	104.16.132.229	TCP	16227 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1380
104.16.132.229	200.200.200.252	TCP	443 → 16227 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1364
200.200.200.252	104.16.132.229	TCP	16227 → 443 [ACK] Seq=1 Ack=1 Win=32768 Len=0
200.200.200.252	104.16.132.229	TLSv1.2	Client Hello
104.16.132.229	200.200.200.252	TCP	443 → 16227 [ACK] Seq=1 Ack=397 Win=65535 Len=0
104.16.132.229	200.200.200.252	TLSv1.2	Server Hello
200.200.200.252	104.16.132.229	TCP	16227 → 443 [ACK] Seq=397 Ack=1365 Win=31404 Len=0
104.16.132.229	200.200.200.252	TLSv1.2	Certificate, Server Key Exchange, Server Hello Done
200.200.200.252	104.16.132.229	TCP	16227 → 443 [ACK] Seq=397 Ack=2502 Win=64398 Len=0
200.200.200.252	104.16.132.229	TCP	16227 → 443 [RST, ACK] Seq=397 Ack=2502 Win=65535 Len=0
200.200.200.252	104.16.132.229	TLSv1.3	Client Hello
104.16.132.229	200.200.200.252	TCP	443 → 49482 [ACK] Seq=1 Ack=518 Win=67584 Len=0
104.16.132.229	200.200.200.252	TLSv1.3	Server Hello, Change Cipher Spec
104.16.132.229	200.200.200.252	TLSv1.3	Application Data
200.200.200.252	104.16.132.229	TCP	49482 → 443 [ACK] Seq=518 Ack=1824 Win=66816 Len=0
200.200.200.252	104.16.132.229	TLSv1.3	Change Cipher Spec, Application Data
200.200.200.252	104.16.132.229	TLSv1.3	Application Data
200.200.200.252	104.16.132.229	TLSv1.3	Application Data
104.16.132.229	200.200.200.252	TCP	443 → 49482 [ACK] Seq=1824 Ack=582 Win=67584 Len=0
104.16.132.229	200.200.200.252	TCP	443 → 49482 [ACK] Seq=1824 Ack=674 Win=67584 Len=0
104.16.132.229	200.200.200.252	TCP	443 → 49482 [ACK] Seq=1824 Ack=1037 Win=68608 Len=0
104.16.132.229	200.200.200.252	TLSv1.3	Application Data, Application Data
200.200.200.252	104.16.132.229	TLSv1.3	Application Data
104.16.132.229	200.200.200.252	TLSv1.3	Application Data
200.200.200.252	104.16.132.229	TLSv1.3	Application Data
104.16.132.229	200.200.200.252	TCP	443 → 49482 [ACK] Seq=3100 Ack=1068 Win=68608 Len=0
104.16.132.229	200.200.200.252	TCP	443 → 49482 [ACK] Seq=3100 Ack=1103 Win=68608 Len=0
200.200.200.252	104.16.132.229	TCP	49482 → 443 [FIN, ACK] Seq=1103 Ack=3100 Win=65536 Len=0
104.16.132.229	200.200.200.252	TCP	443 → 49482 [FIN, ACK] Seq=3100 Ack=1104 Win=68608 Len=0
200.200.200.252	104.16.132.229	TCP	49482 → 443 [ACK] Seq=1104 Ack=3101 Win=65536 Len=0

Packet Flow Diagram

Summary

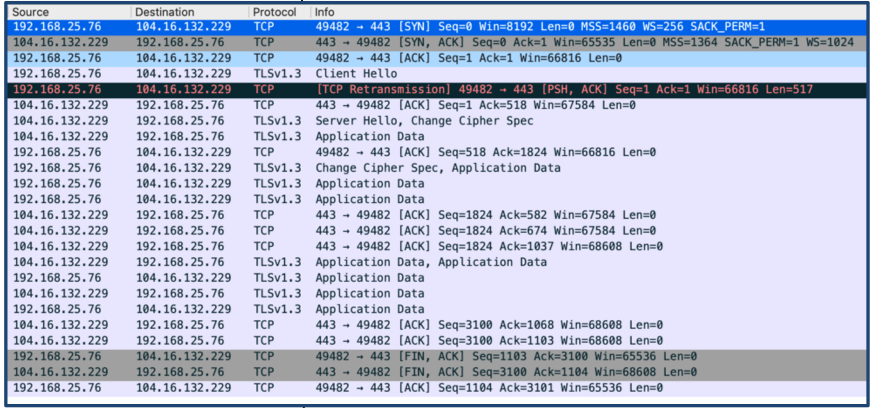
Certificate present in the TLS cache

Client

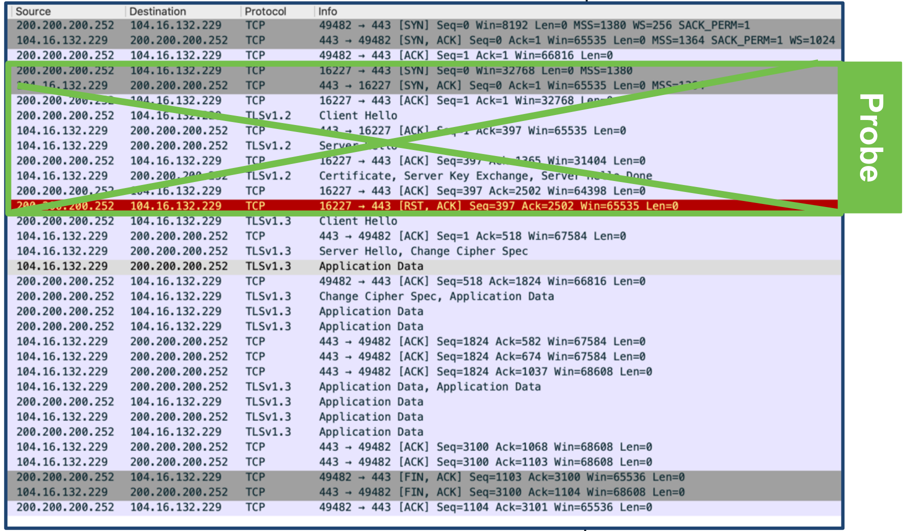
FTD

WWW

Server



```
Source          Destination     Protocol  Info
192.168.25.76   104.16.132.229  TCP       49482 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM=1
104.16.132.229  192.168.25.76   TCP       443 → 49482 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1364 SACK_PERM=1 WS=1024
192.168.25.76   104.16.132.229  TCP       49482 → 443 [ACK] Seq=1 Ack=1 Win=66816 Len=0
192.168.25.76   104.16.132.229  TLSv1.3   Client Hello
192.168.25.76   104.16.132.229  TCP       [TCP Retransmission] 49482 → 443 [PSH, ACK] Seq=1 Ack=1 Win=66816 Len=517
104.16.132.229  192.168.25.76   TCP       443 → 49482 [ACK] Seq=1 Ack=518 Win=67584 Len=0
104.16.132.229  192.168.25.76   TLSv1.3   Server Hello, Change Cipher Spec
104.16.132.229  192.168.25.76   TLSv1.3   Application Data
192.168.25.76   104.16.132.229  TCP       49482 → 443 [ACK] Seq=518 Ack=1824 Win=66816 Len=0
192.168.25.76   104.16.132.229  TLSv1.3   Change Cipher Spec, Application Data
192.168.25.76   104.16.132.229  TLSv1.3   Application Data
192.168.25.76   104.16.132.229  TLSv1.3   Application Data
104.16.132.229  192.168.25.76   TCP       443 → 49482 [ACK] Seq=1824 Ack=582 Win=67584 Len=0
104.16.132.229  192.168.25.76   TCP       443 → 49482 [ACK] Seq=1824 Ack=674 Win=67584 Len=0
104.16.132.229  192.168.25.76   TCP       443 → 49482 [ACK] Seq=1824 Ack=1037 Win=68608 Len=0
104.16.132.229  192.168.25.76   TLSv1.3   Application Data, Application Data
192.168.25.76   104.16.132.229  TLSv1.3   Application Data
104.16.132.229  192.168.25.76   TLSv1.3   Application Data
192.168.25.76   104.16.132.229  TLSv1.3   Application Data
104.16.132.229  192.168.25.76   TCP       443 → 49482 [ACK] Seq=3100 Ack=1068 Win=68608 Len=0
104.16.132.229  192.168.25.76   TCP       443 → 49482 [ACK] Seq=3100 Ack=1103 Win=68608 Len=0
192.168.25.76   104.16.132.229  TCP       49482 → 443 [FIN, ACK] Seq=1103 Ack=3100 Win=65536 Len=0
104.16.132.229  192.168.25.76   TCP       443 → 49482 [FIN, ACK] Seq=3100 Ack=1104 Win=68608 Len=0
192.168.25.76   104.16.132.229  TCP       49482 → 443 [ACK] Seq=1104 Ack=3101 Win=65536 Len=0
```



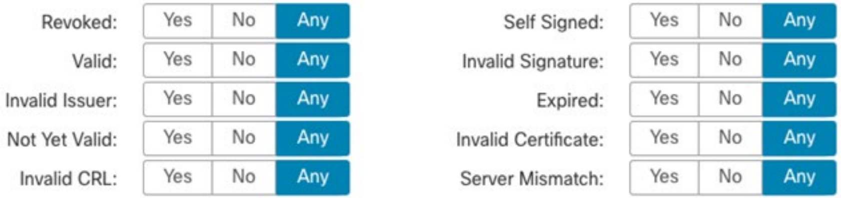
```
Source           Destination      Protocol  Info
200.200.200.252  104.16.132.229   TCP       49482 → 443 [SYN] Seq=0 Win=8192 Len=0 MSS=1380 WS=256 SACK_PERM=1
104.16.132.229   200.200.200.252  TCP       443 → 49482 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1364 SACK_PERM=1 WS=1024
200.200.200.252  104.16.132.229   TCP       49482 → 443 [ACK] Seq=1 Ack=1 Win=66816 Len=0
200.200.200.252  104.16.132.229   TCP       16227 → 443 [SYN] Seq=0 Win=32768 Len=0 MSS=1380
104.16.132.229   200.200.200.252  TCP       443 → 16227 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1364
200.200.200.252  104.16.132.229   TCP       16227 → 443 [ACK] Seq=1 Ack=1 Win=32768 Len=0
200.200.200.252  104.16.132.229   TLSv1.2   Client Hello
104.16.132.229   200.200.200.252  TCP       443 → 16227 [ACK] Seq=1 Ack=397 Win=65535 Len=0
104.16.132.229   200.200.200.252  TLSv1.2   Server Hello
200.200.200.252  104.16.132.229   TCP       16227 → 443 [ACK] Seq=397 Ack=1365 Win=31404 Len=0
104.16.132.229   200.200.200.252  TLSv1.2   Certificate, Server Key Exchange, Server Hello Done
200.200.200.252  104.16.132.229   TCP       16227 → 443 [ACK] Seq=397 Ack=2502 Win=64398 Len=0
200.200.200.252  104.16.132.229   TCP       16227 → 443 [RST, ACK] Seq=397 Ack=2502 Win=65535 Len=0
200.200.200.252  104.16.132.229   TLSv1.3   Client Hello
104.16.132.229   200.200.200.252  TCP       443 → 49482 [ACK] Seq=1 Ack=518 Win=67584 Len=0
104.16.132.229   200.200.200.252  TLSv1.3   Server Hello, Change Cipher Spec
104.16.132.229   200.200.200.252  TLSv1.3   Application Data
200.200.200.252  104.16.132.229   TCP       49482 → 443 [ACK] Seq=518 Ack=1824 Win=66816 Len=0
200.200.200.252  104.16.132.229   TLSv1.3   Change Cipher Spec, Application Data
200.200.200.252  104.16.132.229   TLSv1.3   Application Data
200.200.200.252  104.16.132.229   TLSv1.3   Application Data
104.16.132.229   200.200.200.252  TCP       443 → 49482 [ACK] Seq=1824 Ack=582 Win=67584 Len=0
104.16.132.229   200.200.200.252  TCP       443 → 49482 [ACK] Seq=1824 Ack=674 Win=67584 Len=0
104.16.132.229   200.200.200.252  TCP       443 → 49482 [ACK] Seq=1824 Ack=1037 Win=68608 Len=0
104.16.132.229   200.200.200.252  TLSv1.3   Application Data, Application Data
200.200.200.252  104.16.132.229   TLSv1.3   Application Data
104.16.132.229   200.200.200.252  TLSv1.3   Application Data
200.200.200.252  104.16.132.229   TLSv1.3   Application Data
104.16.132.229   200.200.200.252  TCP       443 → 49482 [ACK] Seq=3100 Ack=1068 Win=68608 Len=0
104.16.132.229   200.200.200.252  TCP       443 → 49482 [ACK] Seq=3100 Ack=1103 Win=68608 Len=0
200.200.200.252  104.16.132.229   TCP       49482 → 443 [FIN, ACK] Seq=1103 Ack=3100 Win=65536 Len=0
104.16.132.229   200.200.200.252  TCP       443 → 49482 [FIN, ACK] Seq=3100 Ack=1104 Win=68608 Len=0
200.200.200.252  104.16.132.229   TCP       49482 → 443 [ACK] Seq=1104 Ack=3101 Win=65536 Len=0
```

Walk-through Demo

Highlighlights

Cisco TLS Server Identity Discover allows us to

- Perform early detection of applications and URL categories within TLS1.3 without full decryption
- Provides Visibility Into TLS 1.3 flows
- Allows us to block TLS 1.3 traffic flows without full decryption based on various certificate options



Revoked: Yes No [✓ Any]

Valid: Yes No [✓ Any]

Invalid Issuer: Yes No [✓ Any]

Not Yet Valid: Yes No [✓ Any]

Invalid CRL: Yes No [✓ Any]

Self Signed: Yes No [✓ Any]

Invalid Signature: Yes No [✓ Any]

Expired: Yes No [✓ Any]

Invalid Certificate: Yes No [✓ Any]

Server Mismatch: Yes No [✓ Any]

- Detect & Block Spoofed SNI by intruder for TLS 1.3 flows

For Reference

- RFC TLS 1.3
- Cisco Blog – Network Security Efficacy in the age of pervasive TLS encryption
- FTD 6.7 maintains your at-risk security policies in a TLS 1.3 world
- 6.7 Configuration Guide
- Cisco NetSec – Secure Firewall Youtube Channel

The bridge to possible

Thank you

cisco Live!

#CiscoLive