

The background features a vibrant, abstract design with a color gradient from dark blue on the left to bright yellow and white on the right. The design consists of overlapping, wavy horizontal bands and a radial pattern of lines emanating from a bright white point on the right side, creating a sense of motion and energy.

CISCO *Live!*

Let's go



The bridge to possible

Pig-in-the-middle

TLS Decryption and Encrypted Visibility Engine Deep Dive on Cisco Secure Firewall

Christopher Grabowski, Technical Marketing Engineer, Technical Leader

Your TLS Speaker



Christopher Grabowski
Technical Marketing Engineer
CCIE Security #42466

Based in Warsaw, Poland

With Cisco since May 2012

Started with TAC Security, then Advanced Services,
now Technical Marketing Engineer

Focusing on Identity Firewall, SDA/ACI Integration and
TLS Decryption

Enjoys cooking and spending time with the family

Housekeeping...



It's a 90 minutes session...

Delta to 2023 delivery:
(based on your feedback):

- TLS 1.3
- Inbound Decryption
- QUIC focus

Download the PDF
version of this deck.
There is a ton of
hidden slides and a
BONUS section!



All slides = Death
by PowerPoint...

Do you still have a Christmas Tree at home 😊?

As per NIST experiment that's what happens when it sparks:



After 2 seconds...



After 4 seconds...



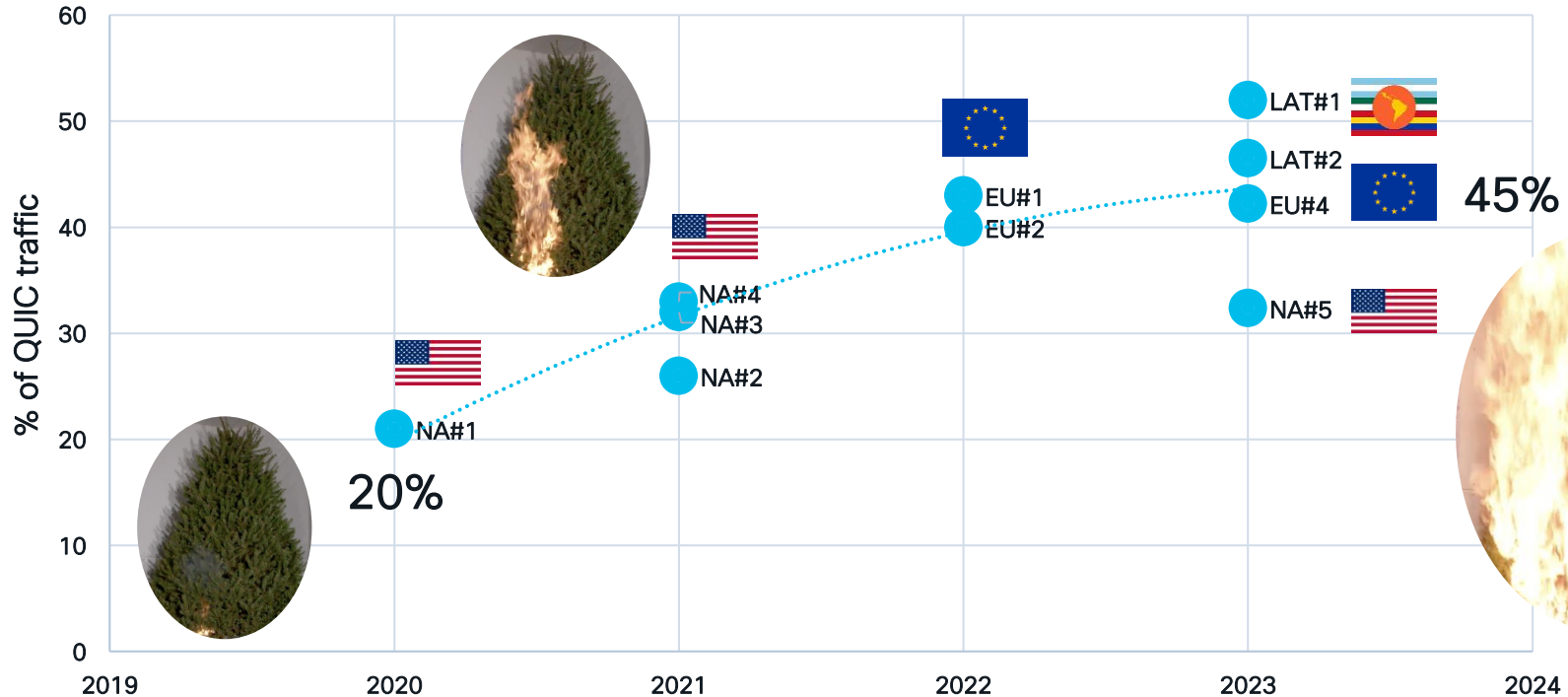
After 9 seconds...

QUIC is Growing Across the World

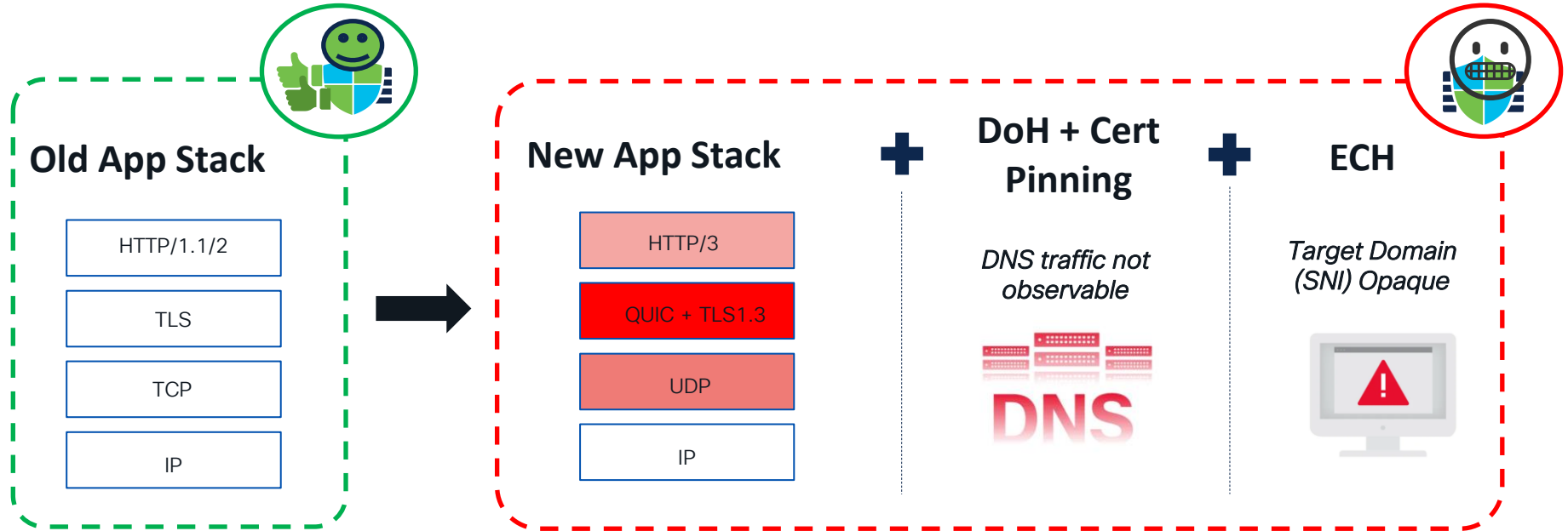
For more stats on QUIC check:
BRKSPM-2024

QUIC traffic evolution data 2020-2023

(Source: BRKSPM-2024)



The Nightmare Slowly Becomes a Reality...



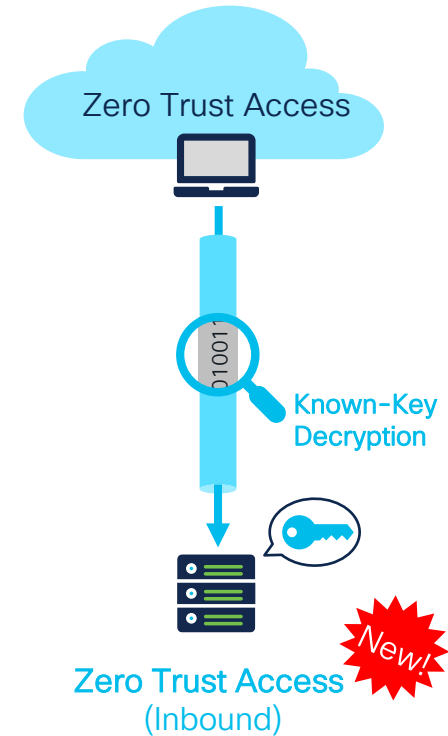
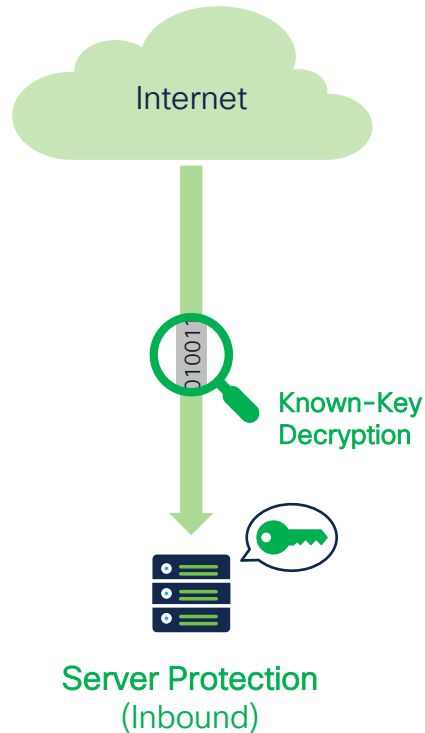
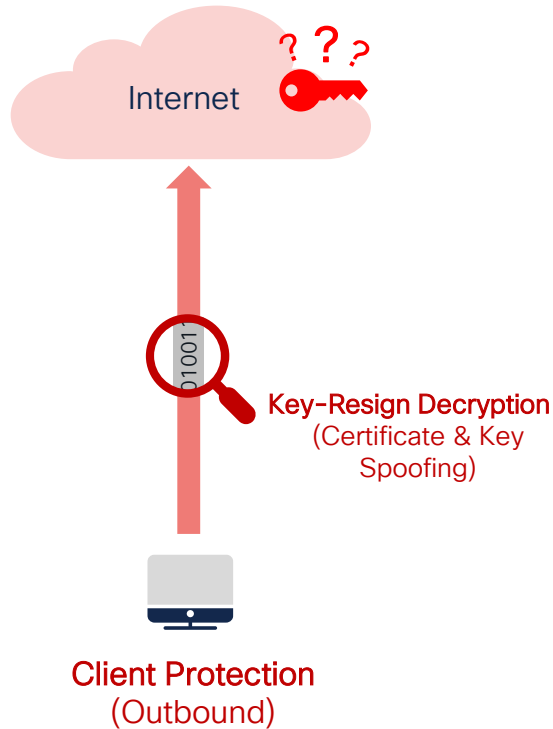
Agenda

- TLS Decryption Under the Hood
 - Client Protection
 - Server Protection
 - Zero Trust Access Clientless
- Challenges Posed by QUIC
- Encrypted Visibility Engine Overview

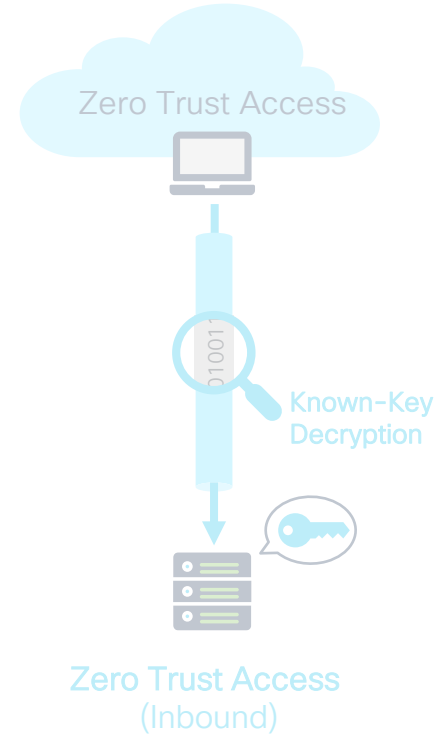
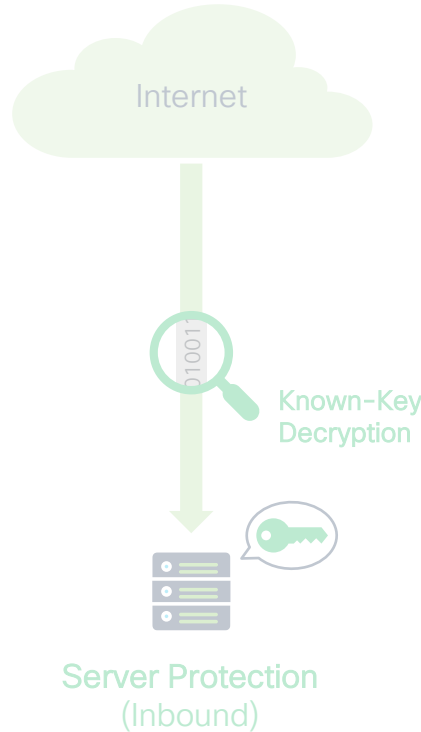
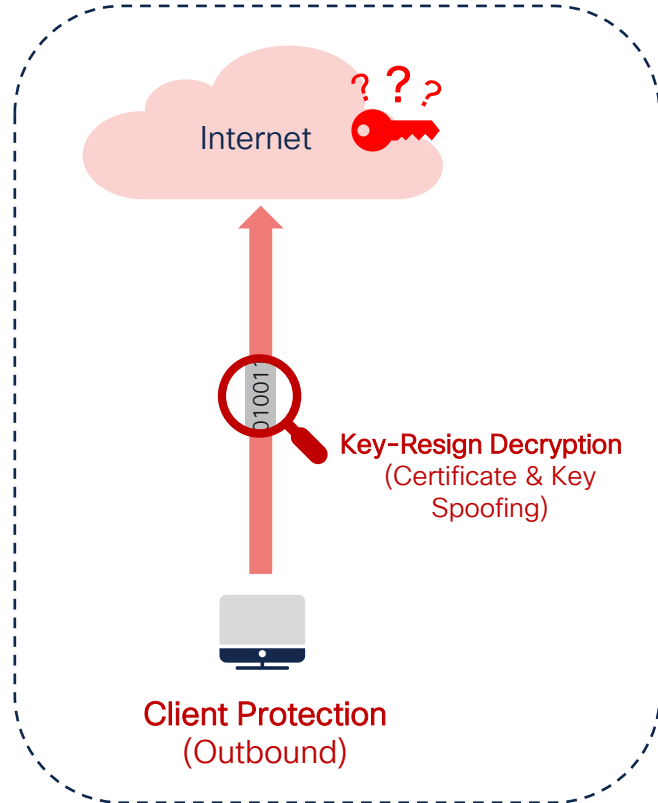
TLS 1.3 Decryption Under the Hood

Three Flavors of TLS Decryption

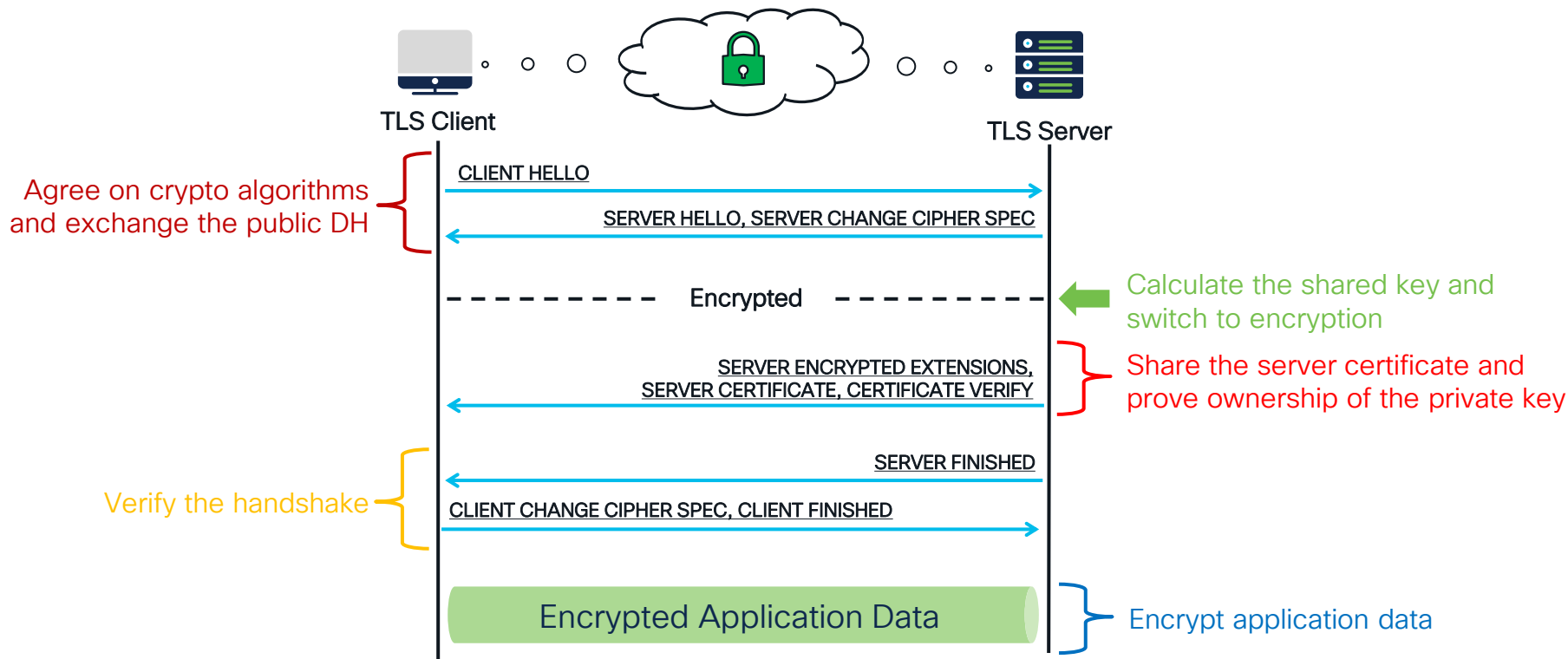
Decryptable protocols:
HTTPs, POP3s, SMTPs, FTPs,
IMAPs



Three Flavors of TLS Decryption



Understanding a TLS Session Flow – Client Side Finish



Decryption Policy – Rule Conditions

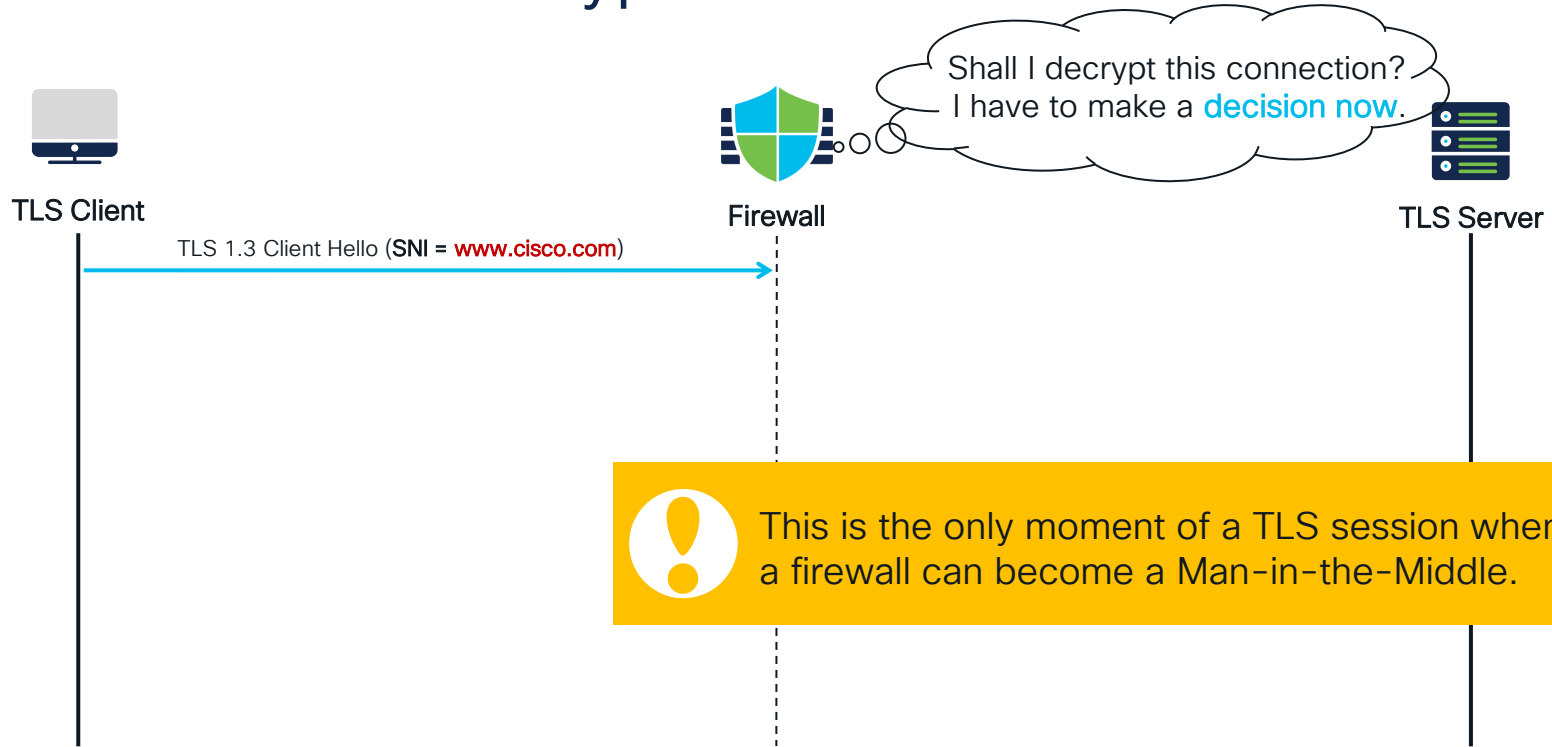
Before TLS Handshake

FTD has the L2-L4 information about the flow

Interface Zones, Networks, Geolocation, VLAN Tags, User Identity, Protocol and Ports

TLS handshake inspection not required

TLS Session Decryption Flow - Client Hello



URL Detectors(SNI)

Add Rule

Name ☒ Enable

Action

Zones Networks VLAN Tags Users Applications Ports **Category** Certificate DN Cert Status Cipher Suite Version Logging

Categories

- Any (Except Uncategorized)
- Finance**

Reputations

- Any**
- 5 - Trusted
- 4 - Favorable
- 3 - Neutral
- 2 - Questionable
- 1 - Untrusted

☒ Apply to unknown reputation

Selected Categories (0)

Make a decryption decision using an **URL categories** condition **matching the SNI** in the Client Hello.

Use **Reputation score** in your rules. E.g. decrypt requests to Questionable and Untrusted URLs only.

<https://www.talosintelligence.com/categories>

Subject Distinguished Name Condition

Configure your own **Distinguished Names objects** to match traffic.

Editing Rule - DND Custom DN List

Name: DND Custom DN List ☒ Enabled Move: below rule

Action: ☒ Do not decrypt

Zones Networks VLAN Tags Users Applications Ports Category Certificate **DN** Cert Status Cipher Suite Version Logging

Available DNs

- Cisco-Undecryptable-Sites
- CN_api.smarththings.com
- CN_apps.apple.com
- CN_ciscospark.com
- CN_citrixonline.com
- CN_core.windows.net
- CN_data.microsoft.com
- CN_data.toolbar.yahoo.com

Subject DNs (8)

- CN=*.wp.pl
- CN=*.microsoft.com
- CN=kokoworld.pl
- CN=*.kokoworld.pl
- CN=*.tuyaeu.com
- CN=*.edoapp.pl
- CN=*.easypack24.net
- CN=ws.batch.com

Issuer DNs (0)

any

Enter DN or CN Add

Enter DN or CN

Subject DN: ***.acme.com**

✓ Matches:

- www.acme.com**
- secure.acme.com**

✗ Does not match:

- www.sub.acme.com**
- top.svc.acme.com**

Application Detectors (SNI)

You can use Talos provided
Application Detectors.

Decryption Policy supports **SNI based Application Detectors only**, hence the lower number comparing to Access Control Policy.

Risks (Any Selected)	Count
<input type="checkbox"/> Very Low	725
<input type="checkbox"/> Low	603
<input type="checkbox"/> Medium	355
<input type="checkbox"/> High	193
<input type="checkbox"/> Very High	96
Business Relevance (Any Selected)	
<input type="checkbox"/> Very Low	795

Available Applications (1972)

Available Applications (3704)

Viewing 1-100 of 1972

Viewing 1-100 of 3704

Under the hood: CH Processing – Dump

```
[DEBUG]: 172.16.136.96 58478 -- 72.163.4.161 443 172.16.136.96 58478 -> 72.163.4.161 443 length [508]
[TRACE]: 172.16.136.96 58478 -- 72.163.4.161 443 client hello: len [508]
```

Packet details with **TCP/IP** source and destination.

Debug dumps the decoded content of the **original Client Hello**.

We can see this packet is a **Client Hello**.

```
version: 3.3
random: e37214889730bf40da321c39aafeca44547a2f620efea7b9d095726da77b676b
session id [32]: 5c3d46c7263f0f9d007ffb91665fb7e2498b74692043df0155aced5ae4119002
cipher_suites len[32]: 5a5a 0113 0213 0313 2bc0 2fc0 2cc0 30c0 a9cc a8cc 13c0 14c0 9c00 9d00 2f00 3500
compression_methods len[1]: 00
---extensions---
server_name[0]: len[18] server name indication: www.cisco.com
extended_master_secret[23]: len[0] Extended Master Secret: enabled
renegotiation_info[65281]: len[1]
supported_groups[10]: len[10] 2a2a 001d 0017 0018
ec_point_formats[11]: len[2] 00
session_ticket[35]: len[0] Session ticket is Empty
alpn_extension[16]: len[14] alpn_list_len[12] ALPN list Entries: h2 http/1.1
status_request[5]: len[5]
signature_algorithms[13]: len[18] 0403 0804 0401 0503 0805 0501 0806 0601
key_share[51]: len[43] groups: x25519(29)
psk_key_exchange_modes[45]: len[2]
supported_versions[43]: len[7] 7a7a 0304 0303
compress_certificate[27]: len[3]
padding_extension[21]: len[202]
```

You can review the client offered **cipher suites**.

Confirm the **Server Name Indication (SNI)**.

Presence of **Key Share** and **Supported Versions** indicates the client is proposing **TLS 1.3** connection.

Under the hood: CH Processing – Evaluate Policy

Check if the received **Client Hello matches** the configured **SSL policy**.

3.4.161 443 Evaluating client hello modify decision for flow

[INFO]: 172.16.136.96 58478 -- 72.163.4.161 443 did not get back a cert for client hello

There is **no certificate cached for the SNI** in the Client Hello.

The **SNI** is used to detect the **Application**.

3.4.161 443 Processing ACTION DECRYPT AND RESIGN rule 13 (Decrypt

The firewall **evaluates the SSL policy until a match** if found.

3.4.161 443 Populating appid cache sni = www.cisco.com sni mismatch = no

[INFO]: 172.16.136.96 58478 -- 72.163.4.161 443 **Matched rule:** Decrypt cisco.com APP, decision is modify

A "decryption" rule was hit. The firewall **makes a decision to modify** the Client Hello and **install itself as a MITM**.

Under the hood: CH Processing – Evaluate Policy

Check if the received **Client Hello matches** the configured **SSL policy**.

.4.161 443 Evaluating client hello modify decision for flow

[INFO]: 172.16.136.96 58478 -- 72.163.4.161 443 did not get back a cert for client hello

There is **no certificate cached for the SNI** in the Client Hello.

The **SNI** is used to detect the **Application**.

.161 443 Processing ACTION DECRYPT AND RESIGN rule 2 (Decrypt cisco.com

The firewall **evaluates the SSL policy until a match** if found.

3.4.161 443 Populating appid cache sni = www.cisco.com sni mismatch = no

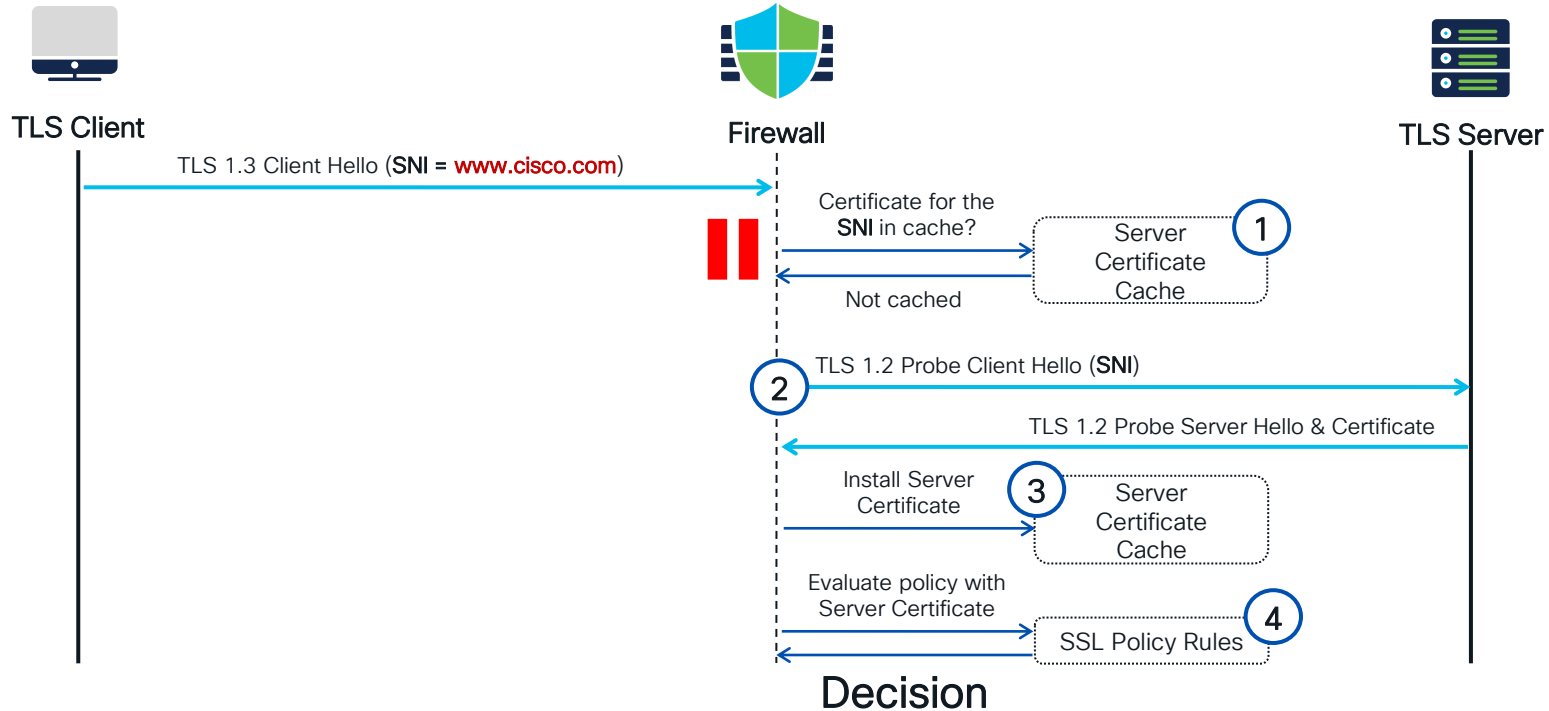
[INFO]: 172.16.136.96 58478 -- 72.163.4.161 443 Matched rule: Decrypt cisco.com APP, decision is modify

A "decryption" rule was hit. The firewall **makes an initial decision to modify** the Client Hello and **install itself as a MITM**.

[INFO]: 172.16.136.96 58478 -- 72.163.4.161 443 Rule 2 (Decrypt cisco.com APP) has requested adaptive probe

The firewall requests a **TLS 1.2 probe** to get server's certificate.

TLS Session Decryption Flow – TLS 1.2 Probing



Under the hood: TLS Probing

[INFO] We receive a **Client Hello towards a non-cached TLS server.**

[INFO] 1 443 Evaluating client hello modify decision for flow 172.16.136.96 58478 -- 72.163.4.161 443
[INFO] 1 443 did not get back a cert for client hello

The policy evaluation results in a **modify decision**...

[INFO]: 172.16.136.96 58478 -- 72.163.4.161 443 ...nevertheless, **we trigger the TLS Probe** to acquire the certificate.

[INFO]: 172.16.136.96 58478 -- 72.163.4.161 443 Matched rule: **Decrypt cisco.com APP, decision is modify**

[INFO]: 172.16.136.96 58478 -- 72.163.4.161 443 Rule 2 (Decrypt cisco.com APP) has requested adaptive probe

[DEBUG] Firewall sends the Client Hello over a **new TCP socket.**

[DEBUG] 43 172.16.136.96 15844 -> 72.163.4.161 443 length [517]
[TRACE] 43 client_hello: len [508]
[DEBUG] 43 client hello modify automatic for probe flow

TLS 1.2
Client Hello

The **certificate** received over the Probe connection **is cached.**

[TRACE] 51 443 certificate: len [4275]

[TRACE]: 172.16.136.96 15844 -- 72.163.4.161 443 **original certificate added to cache**

Server
Certificate

The **Probe connection** serves no additional purpose; hence it **gets reset.**

[TRACE] 61 443 certificate visibility **probe flow being reset**

[TRACE]: 172.16.136.96 58478 -- 72.163.4.161 443 F:c>s **retry packet**, reset record to prevent

Firewall **re-evaluates the policy having the certificate available** in the cache.

[INFO] 1 443 Evaluating CH modify decision for flow 172.16.136.96 58478 -- 72.163.4.161 443
[INFO] 1 443 Got back a cert for client hello

The **initial Client hello is re-injected** for processing.

[INFO]: 172.16.136.96 58478 -- 72.163.4.161 443 **Matched rule: Decrypt cisco.com APP, decision is modify**

The final policy evaluation results in a **modify decision**

Under the hood: Certificate Cache

```
> system support ssl-cache-export
Getting server certificates...Done. File ssl_server_certs.txt was successfully
created at /ngfw/var/common/ folder.

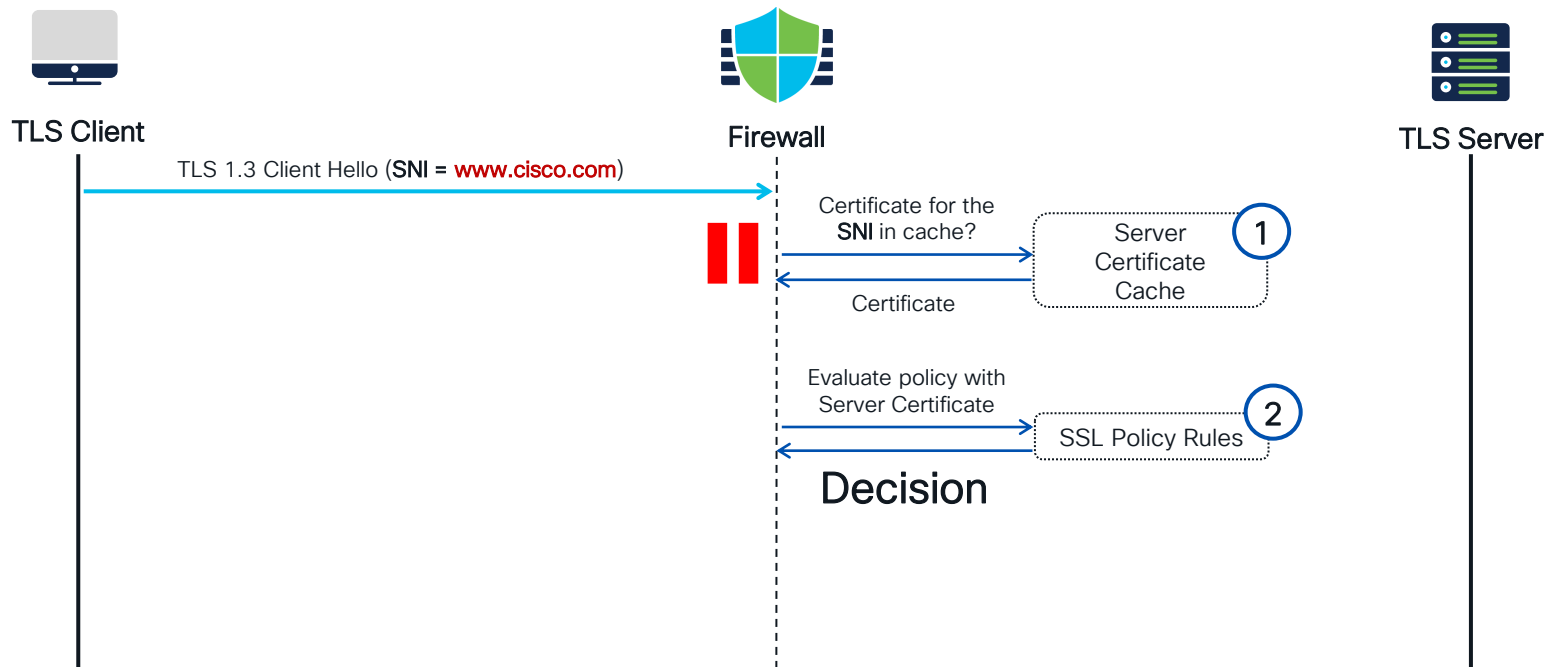
> expert

admin@csftd:~$ cat /ngfw/var/common/ssl_server_certs.txt

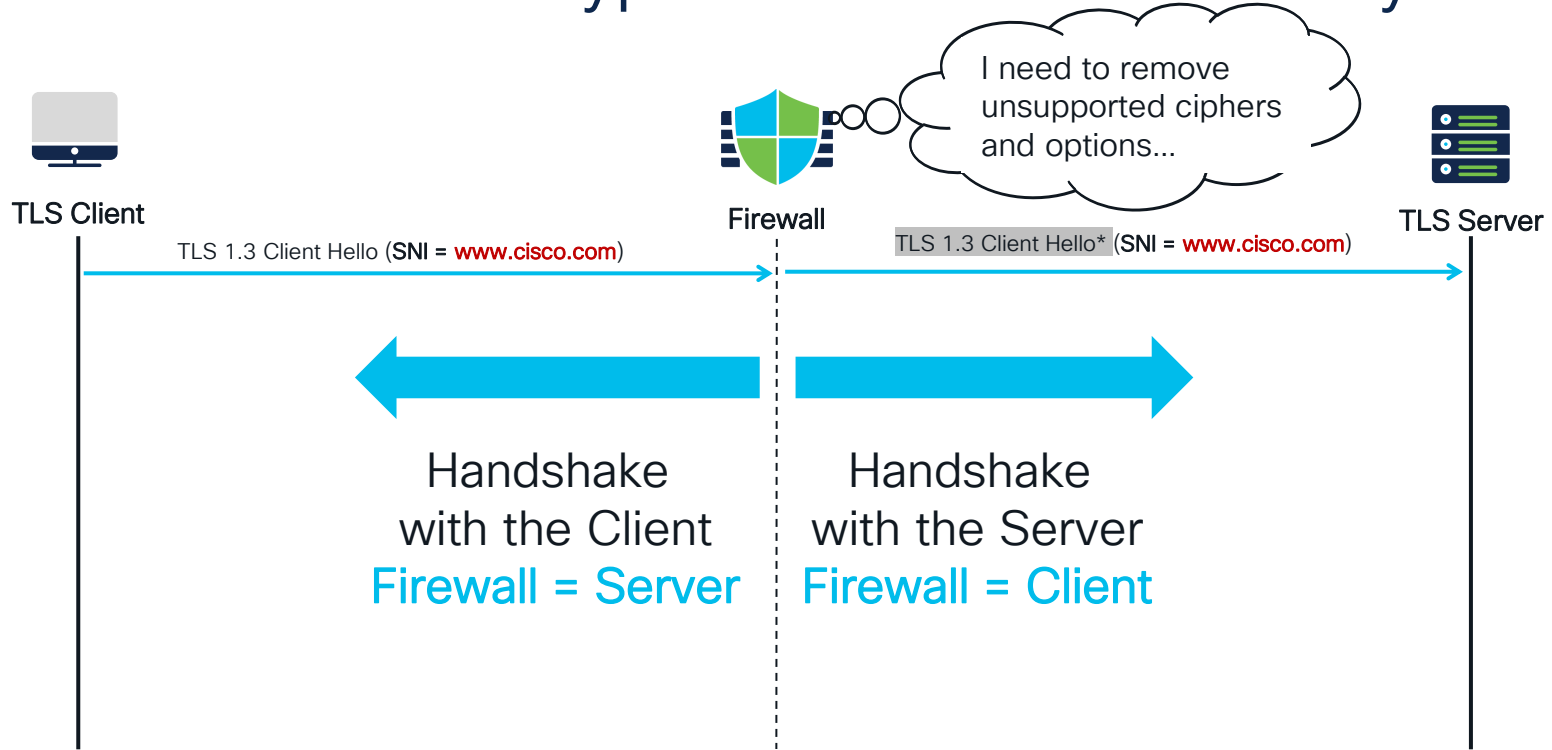
Cache Age (s): 25
Subject: C=US,ST=California,L=San Jose,O=Cisco Systems Inc.,CN=www.cisco.com

-----BEGIN CERTIFICATE-----
MIIIgZCCB2ugAwIBAgIQQAGFnDubtz9v1VdOjAC69DANBgkqhkiG9w0BAQsFADBy
[...]
a9eI9KBs/gfDPWXqn0rrGXPhSixOmBPpzVPpSl9Y9/KLrxYk9jA8pOQ=
-----END CERTIFICATE-----
```

TLS Session Decryption Flow - TLS 1.2 Probe (Subsequent Connection)



TLS Session Decryption Flow – CH Modify



* - modified message

Under the hood: CH Processing - Modify

```
version: 3.3
random: e372...676b
session id [32]: 5c3d...9002
cipher_suites len[32]: 5a5a 0113 0213 0313 2bc0 2fc0 2cc0 30c0
a9cc a8cc 13c0 14c0 9c00 9d00 2f00 3500
compression_methods len[1]: 00
---extensions---
grease[19018]: len[0]
server_name[0]: len[18] server name indication: www.cisco.com
extended_master_secret[23]: len[0] Extended Master Secret: enabled
renegotiation_info[65281]: len[1]
supported_groups[10]: len[10] 2a2a 001d 0017 0018
ec_point_formats[11]: len[2] 00
session_ticket[35]: len[0] Session ticket is Empty
alpn_extension[16]: len[14] alpn_list_len[12]
                        ALPN list Entries: h2 http/1.1

status_request[5]: len[5]
signature_algorithms[13]: len[18] 0403 0804 0401 0503 0805 0501
0806 0601
signed_cert_timestamp[18]: len[0]
key_share[51]: len[43] groups: grease(39578) x25519(29)
psk_key_exchange_modes[45]: len[2]
supported_versions[43]: len[7] 1a1a 0304 0303
compress_certificate[27]: len[3]
unknown[17513]: len[5]
grease[64250]: len[1]
```

```
version: 3.3
random: 0292...71f6 (NEW VALUE)
session id [0]: <ZEROIZED>
cipher_suites len[20]: 5a5a 0113 0213 0313 2bc0 2fc0 2cc0 30c0
a9cc-a8cc-13c0 14c0 9c00 9d00 2f00 3500
compression_methods len[1]: 00
---extensions---
grease[19018]: len[0]
server_name[0]: len[18] server name indication: www.cisco.com
extended_master_secret[23]: len[0] Extended Master Secret: enabled
renegotiation_info[65281]: len[1]
supported_groups[10]: len[8] 2a2a 001d 0017 0018
ec_point_formats[11]: len[2] 00
session_ticket[35]: len[0] Session ticket is Empty
alpn_extension[16]: len[14] alpn_list_len[12]
                        ALPN list Entries: h2 http/1.1

status_request[5]: len[5]
signature_algorithms[13]: len[18] 0403 0804 0401 0503 0805 0501
0806 0601
signed_cert_timestamp[18]: len[0]
key_share[51]: len[43] groups: grease(39578) x25519(29)
psk_key_exchange_modes[45]: len[2]
supported_versions[43]: len[7] 1a1a 0304 0303
compress_certificate[27]: len[3]
unknown[17513]: len[5]
grease[64250]: len[1]
```

Generate a **new Random** and
zeroize Session ID.

Remove **unsupported Cipher Suites.**

Remove **unsupported/redundant
extensions.**

Under the hood: CH Processing – Modify (TLS 1.2 Downgrade)

```

version: 3.3
random: e372...676b
session id [32]: 5c3d...9002
cipher_suites len[32]: fafa 0113 0213 0313 2bc0 2fc0 2cc0 30c0
a9cc a8cc 13c0 14c0 9c00 9d00 2f00 3500
compression_methods len[1]: 00
---extensions---
grease[19018]: len[0]
server_name[0]: len[18] server name indication: www.cisco.com
extended_master_secret[23]: len[0] Extended Master Secret: enabled
renegotiation_info[65281]: len[1]
supported_groups[10]: len[10] 9a9a 001d 0017 0018
ec_point_formats[11]: len[2] 00
session_ticket[35]: len[0] Session ticket is Empty
alpn_extension[16]: len[14] alpn_list_len[12]
                        ALPN list Entries: h2 http/1.1

status_request[5]: len[5]
signature_algorithms[13]: len[18] 0403 0804 0401 0503 0805 0501
0806 0601
signed_cert_timestamp[18]: len[0]
key_share[51]: len[43] groups: grease(39578) x25519(29)
psk_key_exchange_modes[45]: len[2]
supported_versions[43]: len[7] 7a7a 0304 0303
compress_certificate[27]: len[3]
unknown[17513]: len[5]
grease[64250]: len[1]
padding_extension[21]: len[202]

```

```

version: 3.3
random: 0292...71f6 (NEW VALUE)
session id [0]: <ZEROIZED>
cipher_suites len[20]: fafa-0113-0213-0313-2bc0 2fc0 2cc0 30c0
a9cc-a8cc-13c0 14c0 9c00 9d00 2f00 3500
compression_methods len[1]: 00
---extensions---
grease[19018]: len[0]
server_name[0]: len[18] server name indication: www.cisco.com
extended_master_secret[23]: len[0] Extended Master Secret: enabled
renegotiation_info[65281]: len[1]
supported_groups[10]: len[8] 9a9a 001d 0017 0018
ec_point_formats[11]: len[2] 00
session_ticket[35]: len[0] Session ticket is Empty
alpn_extension[16]: len[14] alpn_list_len[12]
                        ALPN list Entries: h2 http/1.1

```

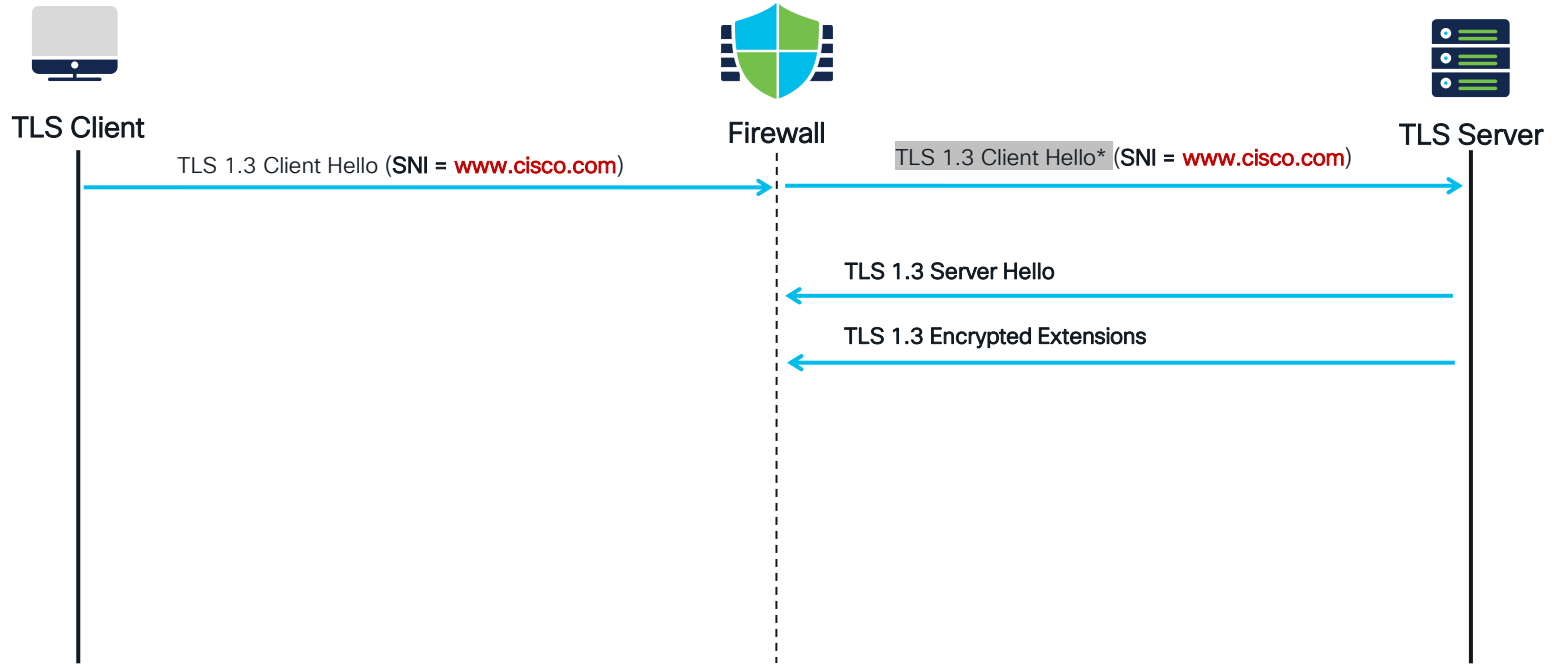
Remove TLS 1.3 extensions to **downgrade the session to TLS 1.2.**

```

status_request[5]:
signature_algorithms[13]: len[18] 0403 0804 0401 0503 0805 0501
0806 0601
signed_cert_timestamp[18]: len[0]
key_share[51]: len[43] groups: grease(39578) x25519(29)
psk_key_exchange_modes[45]: len[2]
supported_versions[43]: len[7] 7a7a 0304 0303
compress_certificate[27]: len[3]
unknown[17513]: len[5]
grease[64250]: len[1]
padding_extension[21]: len[202]

```

TLS Session Decryption Flow – Server Response



Under the hood: Server Hello & Extensions

```
[DEBUG]: Debug dumps the decoded content of
[TRACE]: the Server Hello.
443 72.163.4.161 443 -> 172.16.136.96 58478 length [5429]
443 server hello: len [87]
```

```
version: 3.3
random: 653f6f12ff7da7c8ba1d7bc49b835230e36bc24daf224c11cbef614921239b7a
session id [0]:
cipher_suite: [1302] TLS_AES_256_GCM_SHA384
---extensions---
supported_versions[43]: len[2] 0304
key_share[51]: len[36] group: x25519(29)
```

We can see this packet contains a **Server Hello**.

The **cipher suite** chosen by the server.

```
[DEBUG]: 172.16.136.96 58478 -- 72.163.4.161 443 negotiated params, version [tls 1.3] cipher suite
[TLS_AES_256_GCM_SHA384]
```

TLS 1.3 chosen by the server.

You will find evidence of the firewall **rewriting messages** when forwarding to the client.

```
163.4.161 443 Replacing key share for x25519(29)
```

```
[DEBUG]: 172.16.136.96 58478 -- 72.163.4.161 443 72.163.4.161 443 -> 172.16.136.96 58478 length [30]
```

```
08 00 00 15 00 13 00 00 00 00 00 10 00 0b 00 09
08 68 74 74 70 2f 31 2e 31
```

```
.....
.http/1.1
```

Encrypted **ALPN HTTP/1.1** extension.

```
[DEBUG]: 172.16.136.96 58478 -- 72.163.4.161 443 Processing tls13 encrypted extensions message
[DEBUG]: 172.16.136.96 58478 -- 72.163.4.161 443 Found extension type alpn_extension
```

Block Weak Ciphers and TLS/SSL Versions

Once **Server Hello is received** the firewall can match on **TLS/SSL versions**...

Add Rule

Name: Weak Version ☒ Enabled Insert: below rule

Action: Block

Zones Networks VLAN Tags Users Applications Ports Category Certificate DN Cert Status Cipher Suite **Version** Logging

☒ SSL v3.0
☒ TLS v1.0
☒ TLS v1.1
☐ TLS v1.2
☐ TLS v1.3

[Revert to Defaults](#)

...as well as on **Cipher Suites** that the server selected.

Add Rule

Name: Weak Cipher ☒ Enabled Insert: below rule 9

Action: Block

Zones Networks VLAN Tags Users Applications Ports Category Certificate DN Cert Status **Cipher Suite** Version Logging

Available Cipher Suites

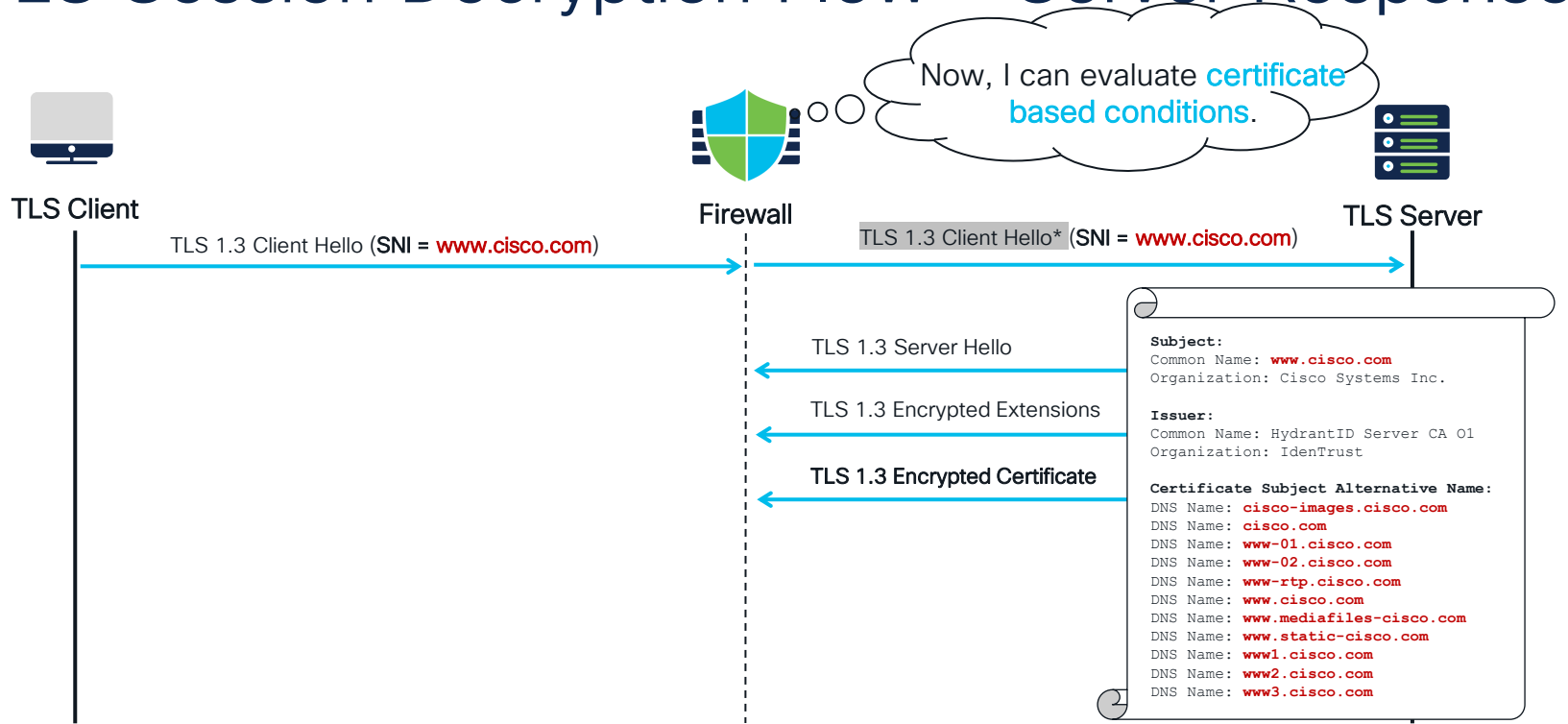
SSL2_DES_192_EDE3_CBC_WITH_MD5
SSL2_DES_64_CBC_WITH_MD5
SSL2_IDEA_128_CBC_WITH_MD5
SSL2_RC2_CBC_128_CBC_WITH_MD5
SSL2_RC4_128_EXPORT40_WITH_MD5
SSL2_RC4_128_WITH_MD5
SSL2_RC4_64_WITH_MD5
TLS_DH_Annon_EXPORT_WITH_RC4_40...

[Add to Rule](#)

Selected Cipher Suites (0)
any

[Cancel](#) [Add](#)

TLS Session Decryption Flow – Server Response



* – modified message

Under the hood: Server Certificate

```
[DEBUG]: 172.16.136.96 58478 -- 72.163.4.161 443 72.163.4.161 443 -> 172.16.136.96 58478 length [4220]
[...]
```

65 6e 74 72 75 73 74 2e 63 6f 6d 2f 63 72 6c 2f	entrust.com/crl/
68 79 64 72 61 6e 74 69 64 63 61 6f 31 2e 63 72	hydrantidcaol.cr
6c 30 82 01 26 06 03 55 1d 11 04 82 01 1d 30 82	10...&..U.....0.
01 19 82 09 63 69 73 63 6f 2e 63 6f 6d 82 0d 77cisco.com..w
77 77 2e 63 69 73 63 6f 2e 63 6f 6d 82 0e 77 77	<u>ww.cisco.com..ww</u>

```
[...]
```

You can see the [Server Certificate](#) dump.

The firewall [decodes the received certificate](#) for further analysis.

...161.443 Parsing tls13 certificate message
...161.443 certificate: len [4211]

```
[TRACE]: 172.16.136.96 58478 -- 72.163.4.161 443 the certificate is valid, status Valid
```

```
[DEBUG]: 172.16.136.96 58478 -- 72.163.4.161 443 sni matches the certificate
```

```
[INFO]: Evaluating rules for flow 172.16.136.96:59857 -> 89.238.73.97:443
```

```
[INFO]: Matched rule: Decrypt cisco.com App, verdict is DecryptResign
```

```
[TRACE]: 172.16.136.96 58478 -- 72.163.4.161 443 policy verdict [DecryptResign]
```

Certificate Conditions – Server Certificate

Add Rule

Name: ☒ Enabled Insert: below rule 2

Action: Do not decrypt

Zones Networks VLAN Tags Users Applications Ports Category **Certificate** DN Cert Status Cipher Suite Version Logging

Available Certificates

Search by name or value

Server_Certificate

Selected Certificates (0)

any

At this stage we can compare a previously uploaded certificate with the one received from the server.

Certificate Conditions – Issuer DN

Add Rule

Name ☒ Enabled Insert below rule 4

Action ☒ Do not decrypt

Zones Networks VLAN Tags Users Applications Ports Category Certificate **DN** Cert Status Cipher Suite Version Logging

Available DNs +

Search by name or value

Subject DNs (0)

any

Issuer DNs (0)

any

CN_ciscopark.com
CN_citrixonline.com
CN_core.windows.net
CN_data.microsoft.com
CN_data.toolbar.yahoo.com

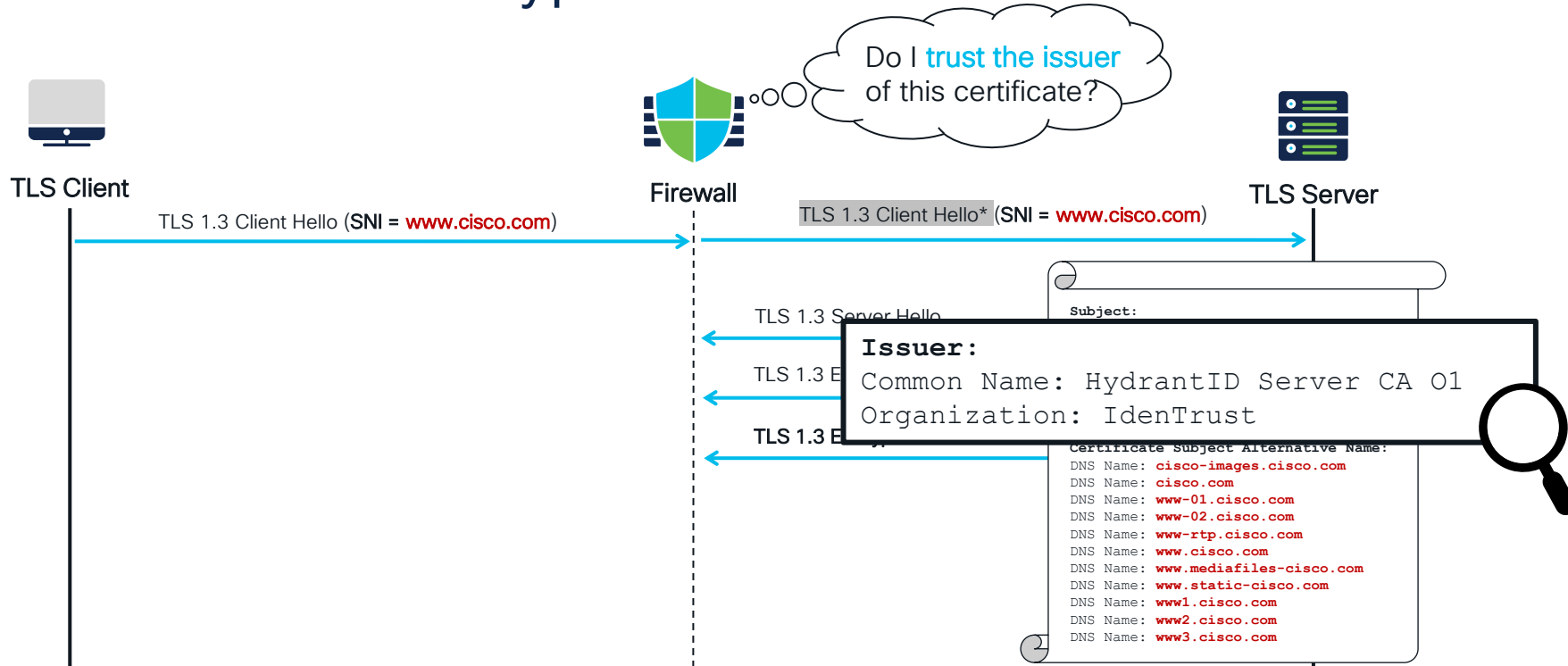
Enter DN or CN Add

Enter DN or CN Add

Cancel Add

Since we now know the certificate of the server, we can match the on the Issuer DN.

TLS Session Decryption Flow – Certificate Check



* - modified message

Under the hood: Server Certificate

```
[DEBUG]: 172.16.136.96 58478 -- 72.163.4.161 443 72.163.4.161 443 -> 172.16.136.96 58478 length [4220]
[...]
```

65 6e 74 72 75 73 74 2e	63 6f 6d 2f 63 72 6c 2f	entrust.com/crl/
68 79 64 72 61 6e 74 69	64 63 61 6f 31 2e 63 72	hydrantidcaol.cr
6c 30 82 01 26 06 03 55	1d 11 04 82 01 1d 30 82	10..&...U.....0.
01 19 82 09 63 69 73 63	6f 2e 63 6f 6d 82 0d 77cisco.com..w
77 77 2e 63 69 73 63 6f	2e 63 6f 6d 82 0e 77 77	ww.cisco.com..ww

```
[...]

[DEBUG]: 172.16.136.96 58478 -- 72.163.4.161 443 Parsing tls13 certificate message
[DEBUG]: 172.16.136.96 58478 -- 72.163.4.161 443 certificate: len [4211]

[TRACE]: 172.16.136.96 58478 -- 72.163.4.161 443 the certificate is valid, status Valid

[DEBUG]: 172.16.136.96 58478 -- 72.163.4.161 443 sni matches the certificate

[INFO]: Evaluating rules for flow 172.16.136.96:59857 -> 89.238.73.97:443

[INFO]: Matched rule: Decrypt cisco.com App

[TRACE]: 172.16.136.96 58478 -- 72.163.4.161 443
```

The certificate is considered valid if:

- the policy **trusts the CA** that issued the certificate
- the **signature is valid**
- the **issuer is valid**
- the certificate **wasn't revoked** (according to uploaded CRL list)
- the **validity date is OK**

Trusted CA Certificates

Firewall Management Center
Policies / Access Control / SSL Policy Editor

Overview Analysis Policies Devices Objects Integration

FTD III SSL Policy (Testing)

Enter Description

Rules **Trusted CA Certificates** Undecryptable Actions Advanced Settings

Available Trusted CAs

Search

- Certum-Trusted-Network-CA-2
- CFCA-EV-ROOT
- Cisco-Basic-Assurance-Root-CA-2099
- Cisco-ECC-Root-CA
- Cisco-Licensing-Root-CA
- Cisco-Root-CA-2048
- Cisco-Root-CA-2099
- Cisco-Root-CA-M1
- Cisco-Root-CA-M2
- Cisco-RXC-R2
- COMODO-ECC-Certification-Authority
- COMODO-RSA-Certification-Authority
- D-TRUST-Root-Class-3-CA-2-2009
- D-TRUST-Root-Class-3-CA-2-EV-2009
- DigiCert-Assured-ID-Root-CA
- DigiCert-Assured-ID-Root-G2

Selected Trusted CAs

- Cisco-Trusted-Authorities

Add to Policy

Save Cancel

Viewing 1-100 of 114

The firewall comes with a predefined set of Trusted CAs.

You can add and remove Trusted CAs as per your needs.

Under the hood: Untrusted Server Certificate (Example)

```
[DEBUG]: 172.16.12.20 64568 -- 173.37.145.84 443 173.37.145.84 443 -> 172.16.12.20 64568 length [5346]  
[TRACE]: 172.16.12.20 64568 -- 173.37.145.84 443 certificate: len [5246]
```

We receive the **Server Certificate** payload.

```
[TRACE]: 172.16.12.20 64568 -- 173.37.145.84 443 the certificate is not valid, status Certificate-Untrusted
```

The received server certificate **has not been signed by a firewall Trusted CA**. For that reason, it is deemed **Untrusted**.

```
[INFO]: 172.16.12.20 64568 -- 173.37.145.84 443 Evaluating rules for flow 172.16.12.20:64568 > 173.37.145.84:443  
173.37.145.84:443 Matched rule: Decrypt cisco.com APP, verdict is DecryptResign
```

By default, the firewall continues with the SSL policy evaluation **despite the certificate is Untrusted**.

In this example, we matched a **regular decryption rule**.

Block Untrusted Certificates

Editing Rule - Block Untrusted Certs

Name: ☒ Enabled [Move](#)

Action: Block

Relevant Fields: Zones, Networks, VLAN Tags, Users, Applications, Ports, Category, Certificate, DN, **Cert Status**, Cipher Suite, Version, Logging

Revoked:	<input type="text" value="Yes"/>	<input type="text" value="No"/>	<input type="text" value="Any"/>	Revert to Defaults
Valid:	<input type="text" value="Yes"/>	<input type="text" value="No"/>	<input type="text" value="Any"/>	
Invalid Issuer:	<input type="text" value="Yes"/>	<input type="text" value="No"/>	<input type="text" value="Any"/>	
Not Yet Valid:	<input type="text" value="Yes"/>	<input type="text" value="No"/>	<input type="text" value="Any"/>	
Invalid CRL:	<input type="text" value="Yes"/>	<input type="text" value="No"/>	<input type="text" value="Any"/>	
Self Signed:	<input type="text" value="Yes"/>	<input type="text" value="No"/>	<input type="text" value="Any"/>	
Invalid Signature:	<input type="text" value="Yes"/>	<input type="text" value="No"/>	<input type="text" value="Any"/>	Revert to Defaults
Expired:	<input type="text" value="Yes"/>	<input type="text" value="No"/>	<input type="text" value="Any"/>	
Invalid Certificate:	<input type="text" value="Yes"/>	<input type="text" value="No"/>	<input type="text" value="Any"/>	
Server Mismatch:	<input type="text" value="Yes"/>	<input type="text" value="No"/>	<input type="text" value="Any"/>	

[Cancel](#) [Save](#)

Select the **Block action** in your SSL Policy Rule.

Select **"Yes"** next to the **Invalid Certificate** condition. The rule will match when the certificate authority is **not in the Trusted CA list**.

Under the hood: Untrusted Server Certificate (Example)

The received server certificate **has not been signed by a firewall Trusted CA**. For that reason, it is deemed **Untrusted**.

```
45.84 443 173.37.145.84 443 --> 172.16.12.20 58774 length [5546]
45.84 443 certificate: len [5246]
45.84 443 the certificate is not valid, status Certificate-Untrusted
```

The firewall proceeds to **SSL Policy evaluation**.

```
45.84 443 Evaluating rules for flow 172.16.12.20:58774/7 -->
```

```
[DEBUG]: 172.16.12.20 58774 -- 173.37.145.84 443 Processing rule 15 (Block Untrusted Certs)
[TRACE]: 172.16.12.20 58774 -- 173.37.145.84 443 Server certificate status: [0100] Certificate-Untrusted
173.37.145.84 443 Matched rule: Block Untrusted Certs, verdict is Block
```

This time, the SSL Policy matched the **“Block Untrusted Certs”** rule, we’ve just created.

In the Unified Events view, we can see a **blocked connection** with **Cert Untrusted** in the SSL Certificate Status column.

Firewall Management Center
Analysis / Unified Events

Overview Analysis Policies Devices

Destination IP 173.37.145.84 X Select...

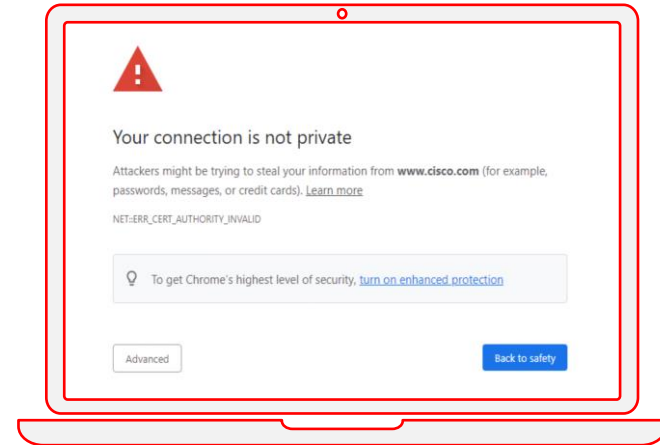
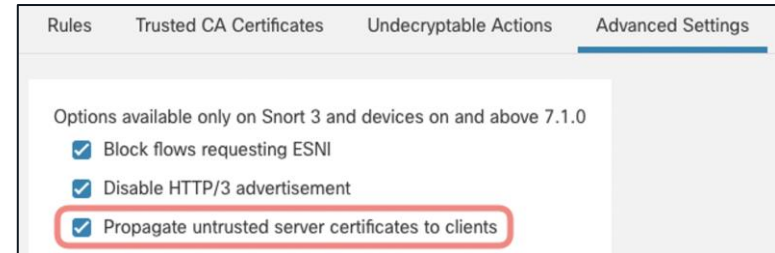
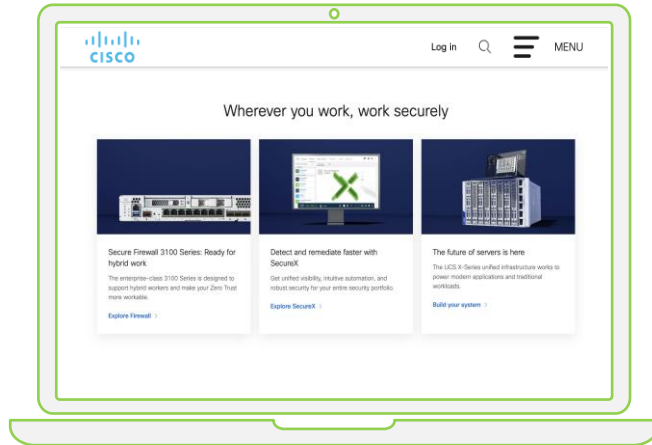
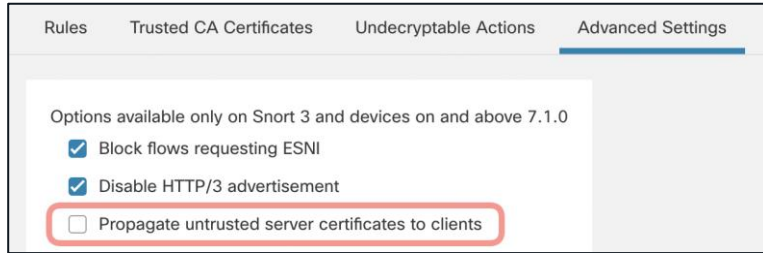
Showing all 6 events (6)

Time	Event Type	Action	SSL Version	SSL Certificate Status
2023-01-04 14:38:21	Connection	Block	TLSv1.2	Cert Untrusted

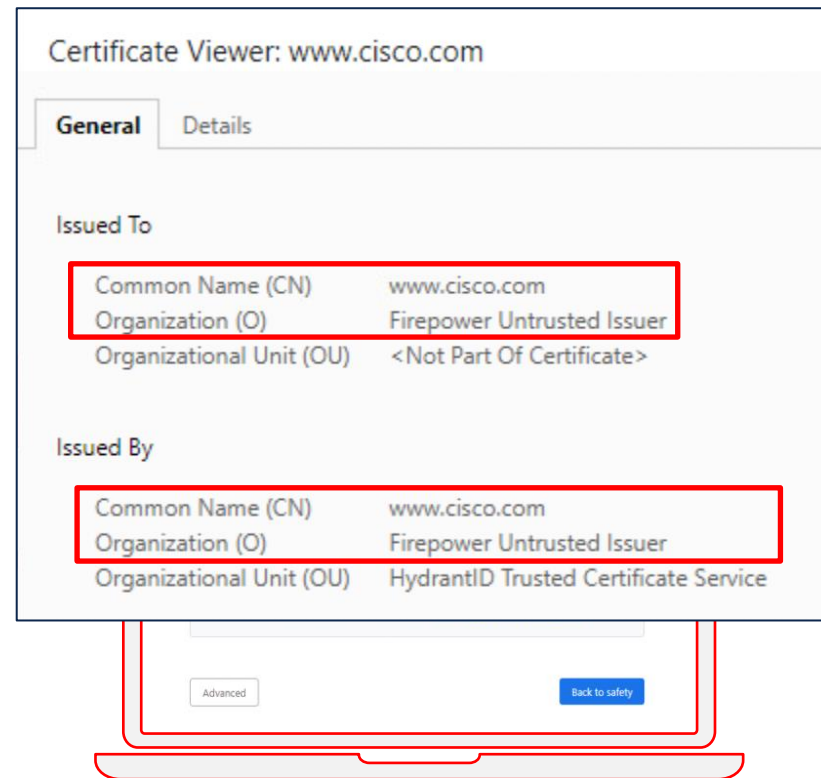
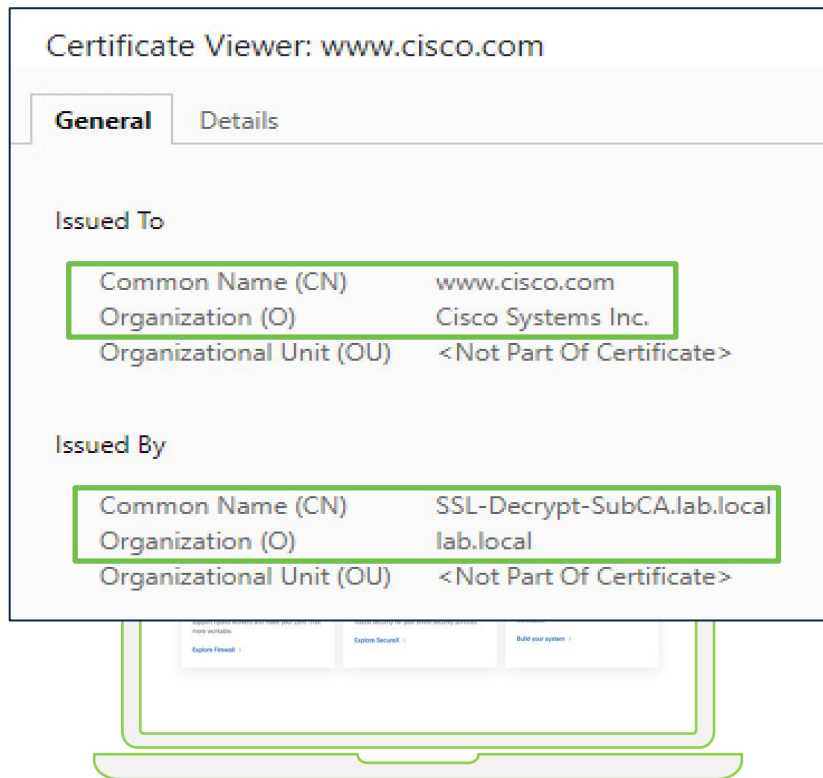
The SSL Flow Flags include an **UNTRUSTED_SERVER_CERTIFICATE** flag.

```
CLIENT_SESSION_ID_SEEN, SEC...
CH_CIPHERS_MODIFIED, CH_CURVES_MODIFIED, CH_PROCESSED,
TLS13_DOWNGRADED, SSL_DETECTED, SERVER_SESSION_ID_SEEN,
CH_EXTENSION_REMOVED, CLIENT_RANDOM_REPLACED, CLIENT_HELLO_SESSTKT,
UNTRUSTED_SERVER_CERTIFICATE, SERVER_RANDOM_REPLACED,
FULL_HANDSHAKE, CH_ALPN_HAS_H2, SH_PROCESSED
```

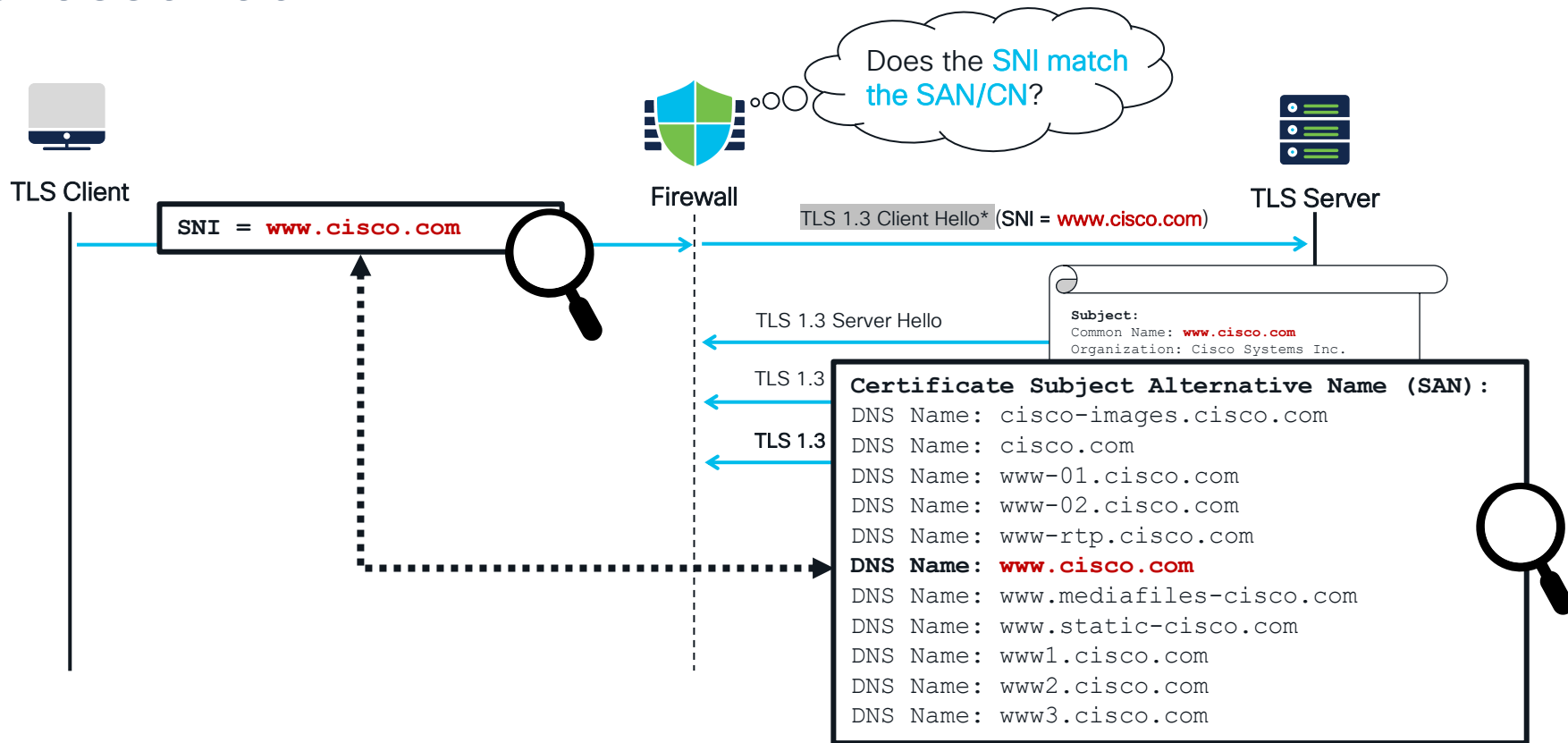
Propagate Untrusted Server Certificate to Clients



Propagate Untrusted Server Certificate to Clients



TLS Session Decryption Flow – CN/SAN and SNI Crosscheck



Under the hood: Server Certificate

```
[DEBUG]: 172.16.136.96 58478 -- 72.163.4.161 443 72.163.4.161 443 -> 172.16.136.96 58478 length [4220]
[...]
```

65 6e 74 72 75 73 74 2e	63 6f 6d 2f 63 72 6c 2f	entrust.com/crl/
68 79 64 72 61 6e 74 69	64 63 61 6f 31 2e 63 72	hydrantidcaol.cr
6c 30 82 01 26 06 03 55	1d 11 04 82 01 1d 30 82	10..&..U.....0.
01 19 82 09 63 69 73 63	6f 2e 63 6f 6d 82 0d 77cisco.com..w
77 77 2e 63 69 73 63 6f	2e 63 6f 6d 82 0e 77 77	ww.cisco.com..ww

```
[...]

[DEBUG]: 172.16.136.96 58478 -- 72.163.4.161 443 Parsing tls13 certificate message
[DEBUG]: 172.16.136.96 58478 -- 72.163.4.161 443 certificate: len [4211]

[TRACE]: 172.16.136.96 58478 -- 72.163.4.161 443 the certificate is valid, status Valid

[DEBUG]: 172.16.136.96 58478 -- 72.163.4.161 443 sni matches the certificate

[INFO]: Evaluating rules for flow 172.16.136.96:59857 -> 89.238.73.97:443

[INFO]: Matched rule: Decrypt cisco.com App, verdict is DecryptResign

[TRACE]: 172.16.136.96 58478 -- 72.163.4.161 443 policy verdict [DecryptResign]
```

The firewall crosschecks the server certificate if **SAN/CN matches the SNI** in TLS Client Hello.

Block Server SNI Mismatch

Select the **Block action** in your SSL Policy Rule.

Add Rule

Name: Block SNI Mismatch ☒ Enabled Insert: below rule 2

Action: **Block**

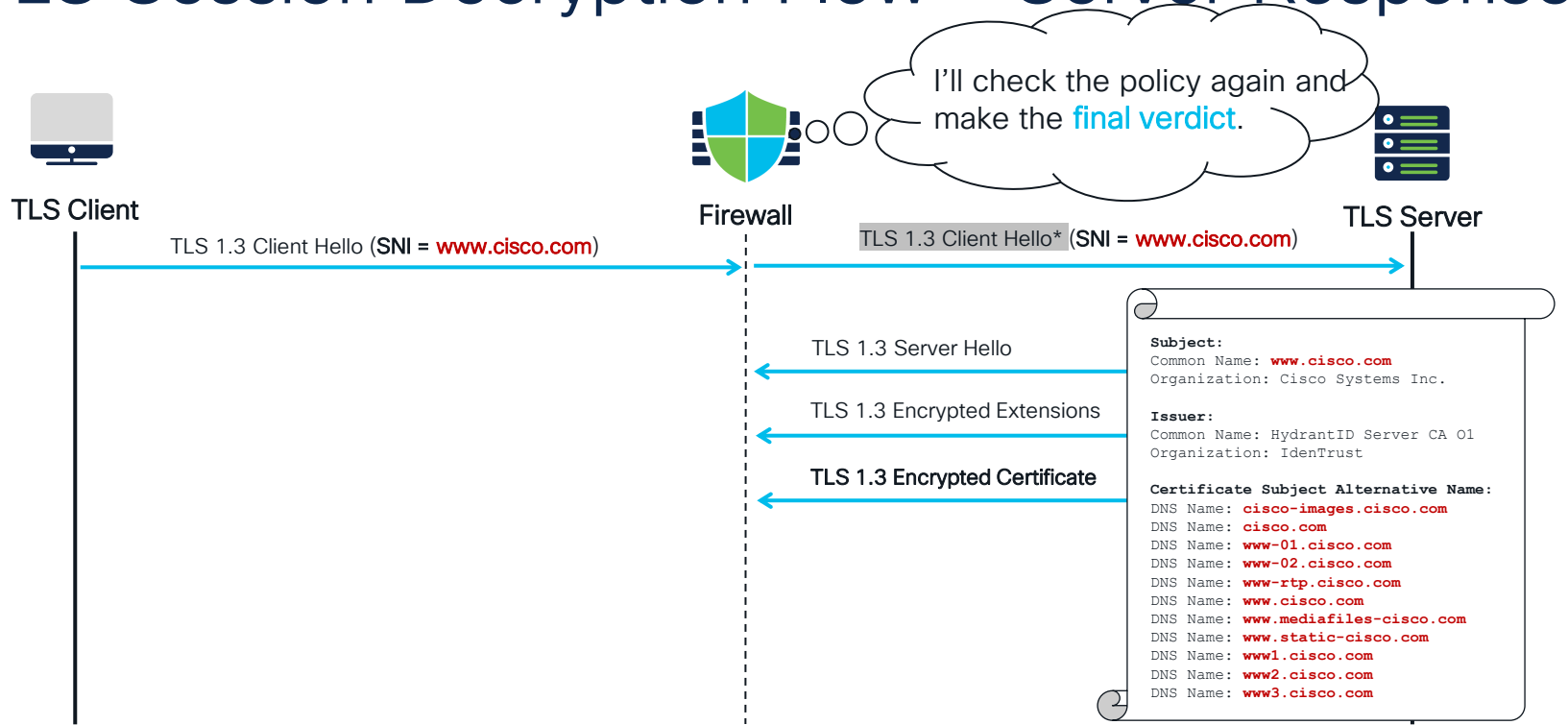
Zones Networks VLAN Tags Users Applications Ports Category Certificate DN **Cert Status** Cipher Suite Version Logging

Revoked:	Yes	No	Any	Self Signed:	Yes	No	Any	Revert to Defaults
Valid:	Yes	No	Any	Invalid Signature:	Yes	No	Any	
Invalid Issuer:	Yes	No	Any	Expired:	Yes	No	Any	
Not Yet Valid:	Yes	No	Any	Invalid Certificate:	Yes	No	Any	
Invalid CRL:	Yes	No	Any	Server Mismatch:	Yes	No	Any	

Select "Yes" next to the **Server Mismatch** condition. The rule will match when the **SNI in the Client Hello** doesn't match **Server's Certificate**.

Cancel Add

TLS Session Decryption Flow – Server Response



* – modified message

Under the hood: Server Certificate

```
[DEBUG]: 172.16.136.96 58478 -- 72.163.4.161 443 72.163.4.161 443 -> 172.16.136.96 58478 length [4220]
[...]
```

65 6e 74 72 75 73 74 2e	63 6f 6d 2f 63 72 6c 2f	entrust.com/crl/
68 79 64 72 61 6e 74 69	64 63 61 6f 31 2e 63 72	hydrantidcaol.cr
6c 30 82 01 26 06 03 55	1d 11 04 82 01 1d 30 82	10...&...U.....0.
01 19 82 09 63 69 73 63	6f 2e 63 6f 6d 82 0d 77cisco.com..w
77 77 2e 63 69 73 63 6f	2e 63 6f 6d 82 0e 77 77	ww.cisco.com..ww

```
[...]
```

```
[DEBUG]: 172.16.136.96 58478 -- 72.163.4.161 443 Parsing tls13 certificate message
[DEBUG]: 172.16.136.96 58478 -- 72.163.4.161 443 certificate: len [4211]
```

```
[TRACE]: 172.16.136.96 58478 -- 72.163.4.161 443 the certificate is valid,
```

```
[DEBUG]: 172.16.136.96 58478 -- 72.163.4.161 443 sni matches the certificate
```

```
[INFO]: Evaluating rules for flow 172.16.136.96:59857 -> 89.238.73.97:443
```

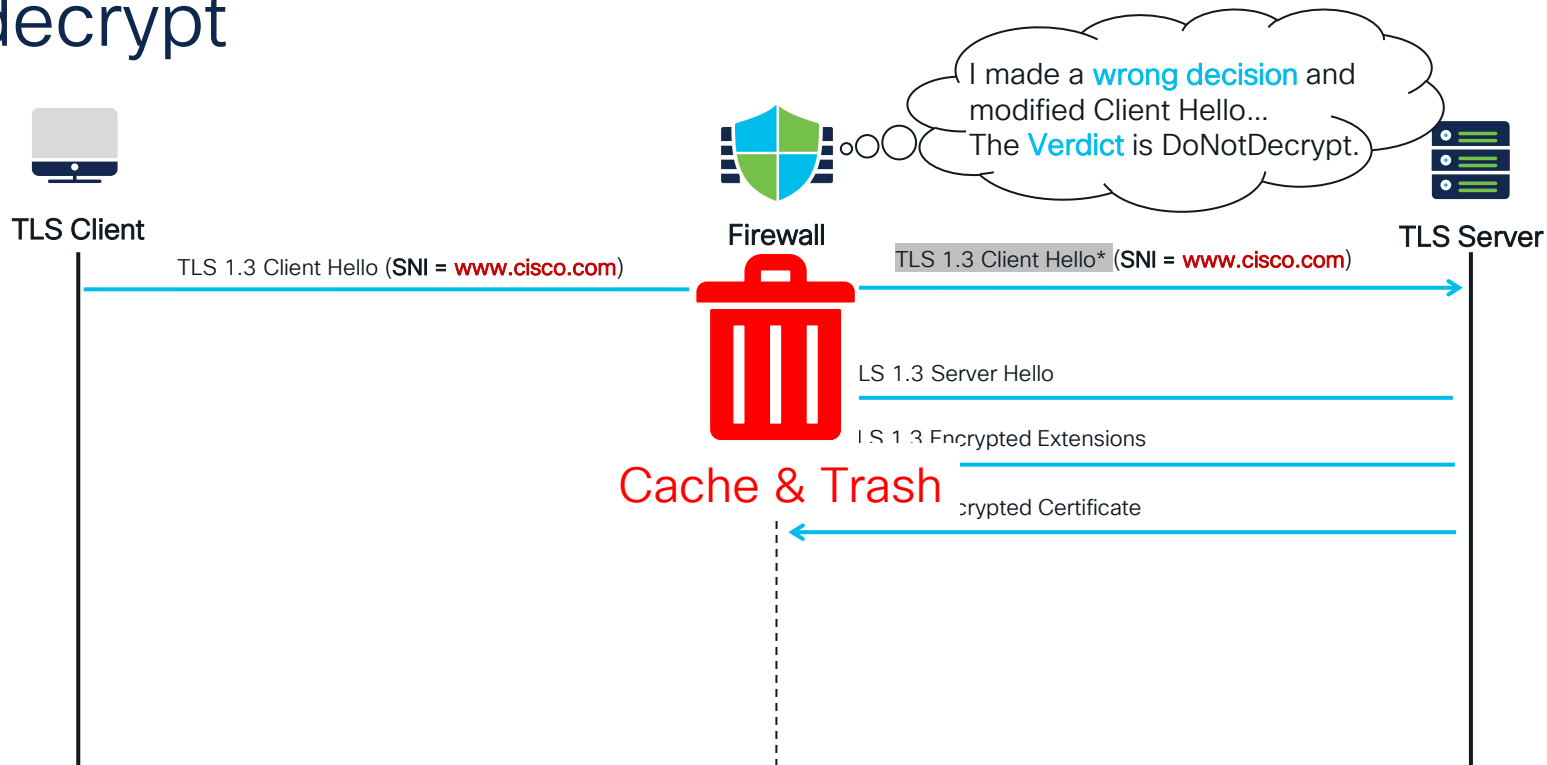
```
[INFO]: Matched rule: Decrypt cisco.com App, verdict is DecryptResign
```

```
[TRACE]: 172.16.136.96 58478 -- 72.163.4.161 443 policy verdict [DecryptResign]
```

Now, having both Client Hello and Server Certificate, the firewall. **re-evaluates the decryption policy**

Firewall concludes the **final verdict**.

TLS Session Decryption Flow – Late do not decrypt



Under the hood: SNI Mismatch – First Flow

```
[TRACE]: 172.16.12.20 58873 -- 172.16.15.11 443 client_hello: len [508]
```

```
[...]
```

```
server_name[0]: len[22] server_name indication: spoofed.lab.local
```

We receive a Client Hello with **spoofed SNI**.

There is **no certificate in the cache** for the spoofed.lab.local SNI.

```
15.11 443 did not get back a cert for client hello
```

```
[INFO]: 172.16.12.20 58873 -- 172.16.15.11 443 Matched rule: Decrypt spoofed.lab.local DN
```

Policy evaluation results in a **decryption decision** and we **modify the Client Hello**.

The firewall realizes the Client Hello **SNI doesn't match the SNI/CN** in the Server's certificate.

```
443 certificate: len [1385]
```

```
443 the sni does not match the certificate
```

The Server's **certificate is cached**.

The mismatch details are available in the debugs **SNI vs CN/SAN**.

```
15.11 443 original certificate added to cache
```

```
[D 15.11 443 Not an exact match comparing CN=csdac20.lab.local with SNI=spoofed.lab.local for rule lookup
```

[DEBUG] The rule we matched with Client Hello, **doesn't match now!!!**

```
11 443 Rule(Decrypt spoofed.lab.local DN) did not match subject name
```

We realize we've made a **WRONG DECISION**... The **VERDICT** is to not decrypt and **reset the connection**.

```
11 443 client hello modified when it should not have been  
policy verdict [BlockWithReset]
```

Under the hood: SNI Mismatch – Subsequent Flow

```
[TRACE]: 172.16.12.20 58882 -- 172.16.15.11 443 client_hello: len [508]
[...]
```

server_name[0]: len[22] server name indication: spoofed.lab.local

```
[...]
```

We receive a Client Hello with **spoofed SNI**.

We have a **certificate in the cache** for the **spoofed.lab.local SNI**.

```
[INFO]: 172.16.12.20 58882 -- 172.16.15.11 443 Got back a cert for client hello
```

```
[DEB] 172.16.12.20 58882 -- 172.16.15.11 443 The firewall realizes there is an SNI mismatch ahead of making the decision.
```

```
172.16.15.11 443 | Not an exact match comparing CN csdac20.lab.local with SNI spoofed.lab.local for rule lookup
```

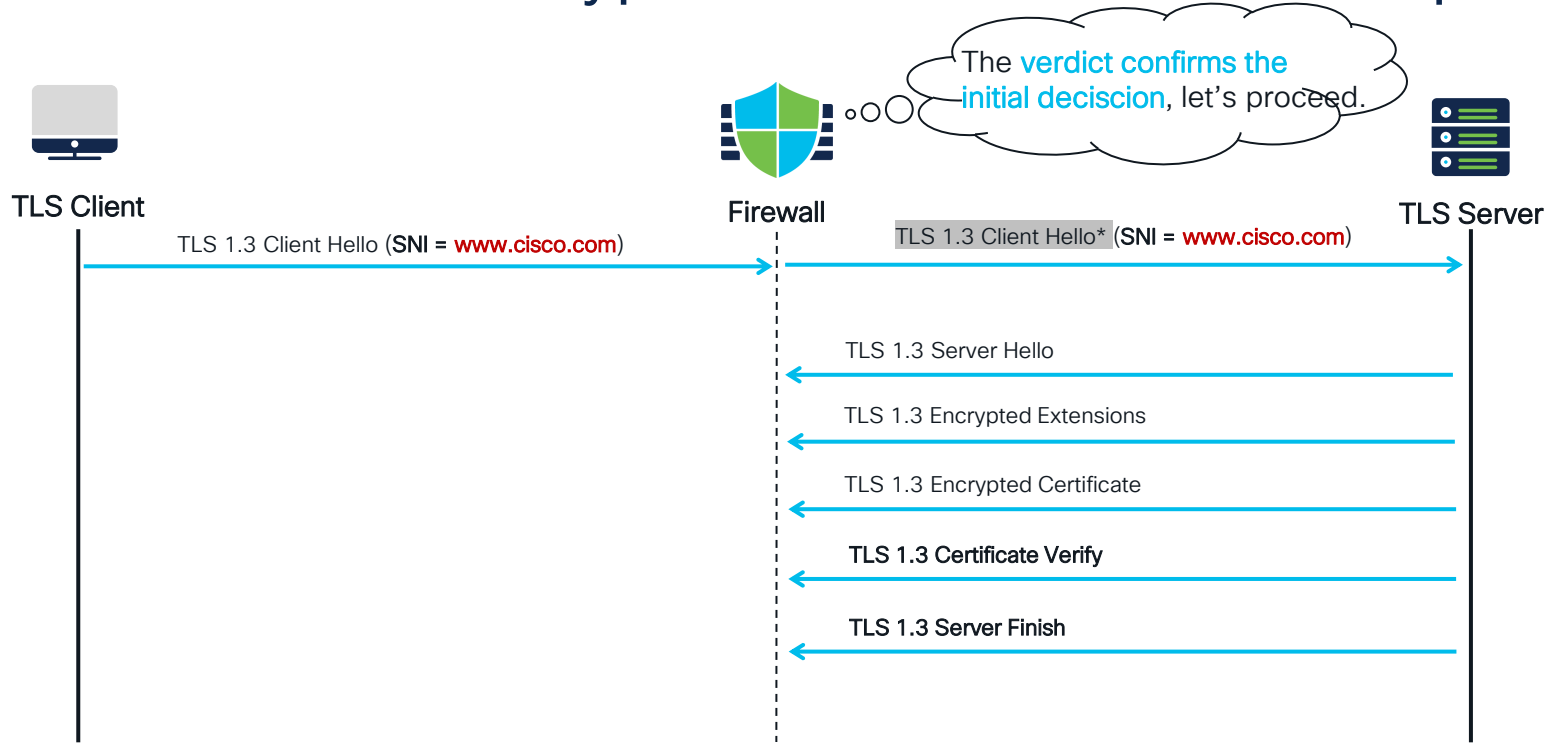
```
[DEB] 172.16.12.20 58882 -- 172.16.15.11 443 We evaluate the policy with the CN=csdac20.lab.local instead of SNI=spoofed.lab.local.
```

```
172.16.15.11 443 | Rule 16 (Decrypt spoofed.lab.local DN) did not match subject name
```

An **accurate decision** is made, and the client connection **will succeed**.

```
[INFO]: 172.16.12.20 58882 -- 172.16.15.11 443 No rules matched, decision is don't modify
```

TLS Session Decryption Flow – Server Response



* - modified message

Under the hood: Verify and Finished

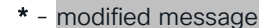
We get the **Server Verify** message.

```
[DEBUG]: 172.16.136.96 58478 -- 72.163.4.161 443 72.163.4.161 443 -> 172.16.136.96 58478 length [525]
[T The firewall do not bother with processing
  the Server Finished message.
[DEBUG]: 172.16.136.96 58478 -- 72.163.4.161 443 72.163.4.161 443 -> 172.16.136.96 58478 length [57]
[TRACE]: 172.16.136.96 58478 -- 72.163.4.161 443 tls13 session, skipping finished message processing

[DEBUG]: 172.16.136.96 58478 -- 72.163.4.161 443 72.163.4.161 443 -> 172.16.136.96 58478 Updating
connection event: Logging: true action: 6 rule: 1 status: 2 flow err: XTLS SUCCESS flags: 341808ac480a144
```

The Firewall drops the details of this crypto session in the **connection logs**.

CISCO *Live!*



Internal CA Certificate

Firewall Management Center
Objects / Object Management

Overview Analysis Policies Devices Objects Integration

Internal CAs

Internal certificate authority (CA) object represents the CA public key certificate of a CA your organization uses to issue server certificates with the internal CA.

Edit Internal Certificate Authority

Name: decrypt.emealab.local

Subject:

- Common Name: decrypt.emealab.local
- Organization: EMEA Lab
- Organization Unit: Laboratory

Issuer:

- Common Name: emealab-WIN-RCI9756298A-CA
- Organization:
- Organization Unit:

Not Valid Before: Mar 30 09:22:07 2023 GMT

Not Valid After: Mar 30 09:32:07 2025 GMT

Serial Number: 16:00:00:00:0b:7a:dd:9b:61:b8:91:40:31:00:00:00:00:0b

Download Cancel Save

The Internal Certificate Authority **issues (spoofs)** certificates **on the fly**.

The certificate of **Internal CA** must be signed **by a Root CA** trusted by clients.

Decrypt – Resign Certificate

Editing Rule – Decrypt cisco.com APP

Name: Decrypt cisco.com APP ☒ Enabled [Move](#)

Action: Decrypt - Resign with decrypt.emealab.local ☒ Replace Key Only

[Zones](#) [Networks](#) [VLAN Tags](#) [Users](#) [Applications](#) [Ports](#) [Category](#) [Certificate](#) [DN](#) [Cert Status](#) [Cipher Suite](#) [Version](#) [Logging](#)

Available Zones [↻](#)

Search by name

- Internet
- V134
- V136
- V137

[Add to Source](#) [Add to Destination](#)

Source Zones (0)

any

Destination Zones (0)

any

[Cancel](#) [Save](#)

You specify the **Decrypt-Resign Internal CA certificate** per decryption rule.

Under the hood: Certificate Resign (Example)

```
[DEBUG]: 172.16.12.20 64427 -- 173.37.145.84 443 173.37.145.84 443 -> 172.16.12.20 64427
[TRACE]: 172.16.12.20 64427 -- 173.37.145.84 443 certificate: len [5246]
```

We receive a **Server Certificate** message.

```
[TRACE] 172.16.12.20 64427 -- 173.37.145.84 443 the certificate is valid, status Valid
[DEBUG] 172.16.12.20 64427 -- 173.37.145.84 443 sni matches the certificate
[INFO]: 172.16.12.20 64427 -- 173.37.145.84 443 Evaluating rules for flow 172.16.12.20:64427>173.37.145.84:443
[INFO]: 172.16.12.20 64427 -- 173.37.145.84 443 Matched rule: Decrypt cisco.com APP, verdict is DecryptResign
```

The firewall checks the certificate, reevaluates the policy and makes the **final decrypt & resign Verdict**.

The original certificate uses **Elliptic Curve Public Key** with ECDSA-SHA256 signing algorithm.

As our **Internal CA uses RSA Public key** with RSA-SHA256 signing algorithm we need to **downgrade to spoof the certificate**.

```
173.37.145.84 443 Resign - server cert key type is ecdsa, info 23, current sig_alg ecdsa-with-SHA256
173.37.145.84 443 Switching ecdsa to rsa2k
173.37.145.84 443 changing NID_ecdsa_with_SHA256 to NID_sha256WithRSAEncryption
173.37.145.84 443 Switched sig_alg from 794 to 668 for certificate signing
```

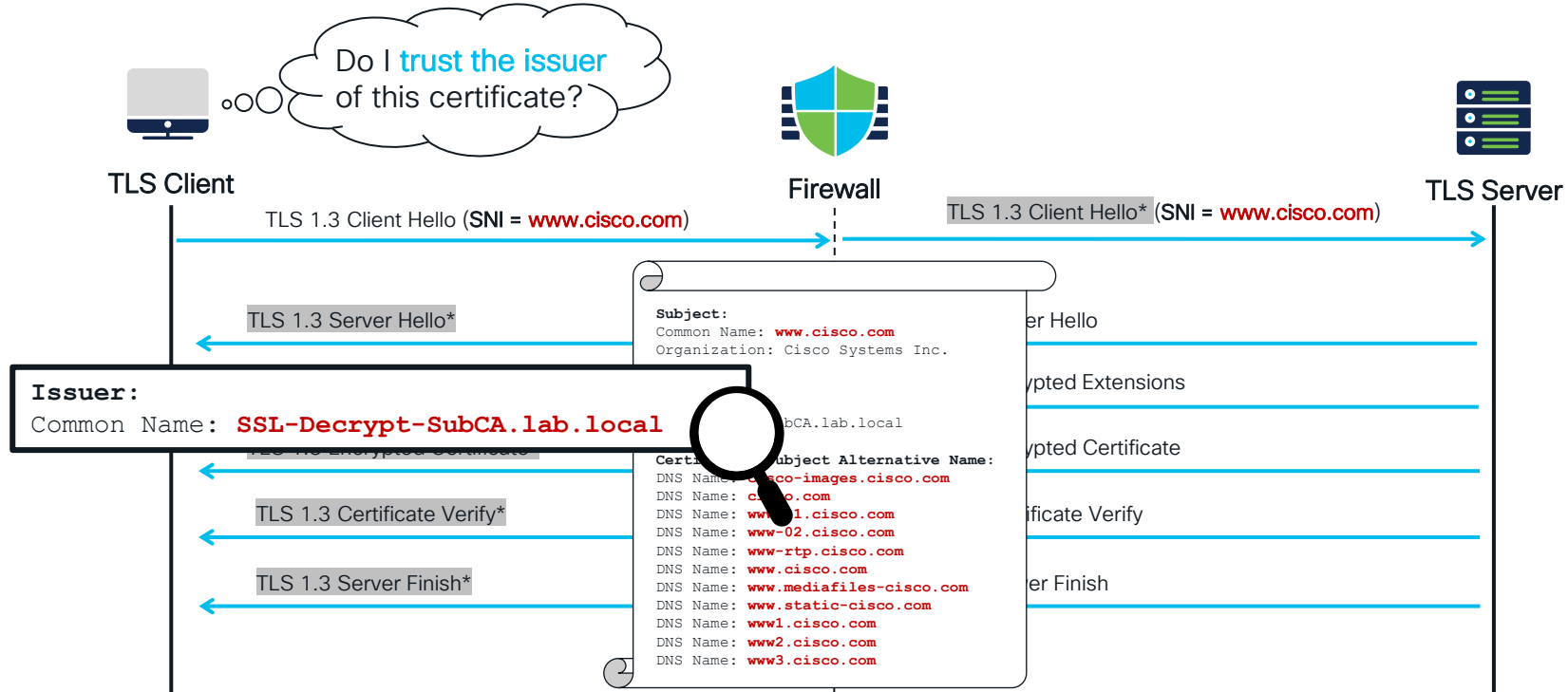
```
[DEBUG]: 172.16.12.20 64427 -- 173.37.145.84 443 cert not found in cache
```

The spoofed certificate is created and **sent to the client**.

The spoofed certificate is generated for the first time. **As it is expensive, we'll cache it** for future use once created.

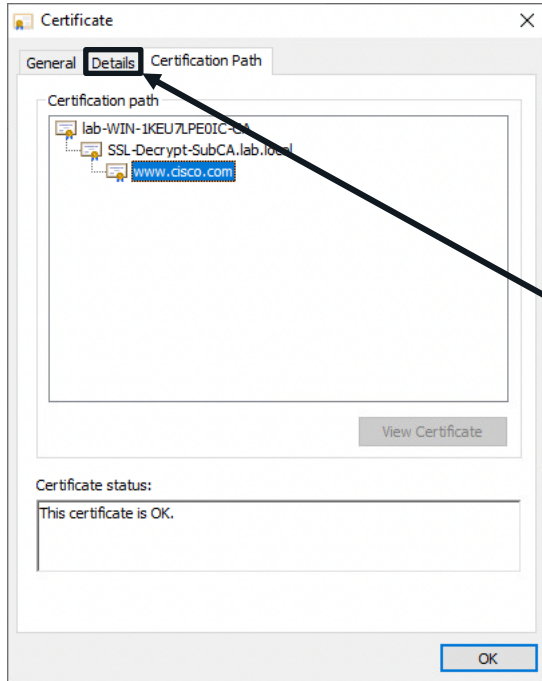
```
[INFO]: 172.16.12.20 64427 -- 173.37.145.84 443 Creating a new certificate
```

TLS Session Decryption Flow – Client Check



* – modified message

An Example of a Resigned Certificate



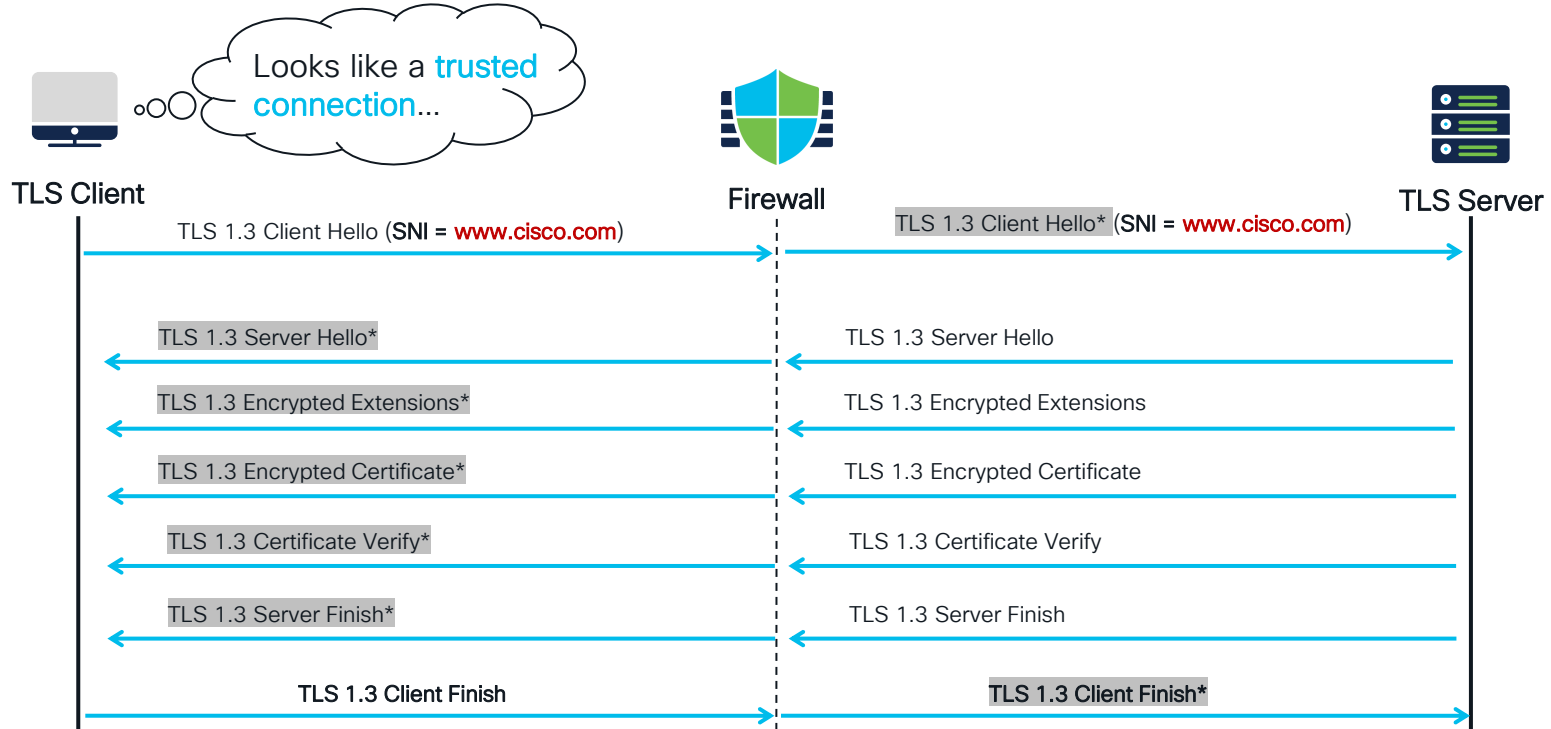
Resigned Certificate

Field	Value
Version	V3
Serial number	17bd3e63bfda13212edf9300
Signature algorithm	sha256RSA
Signature hash algorithm	sha256
Issuer	SSL-Decrypt-SubCA.lab.local, I...
Valid from	Wednesday, February 16, 20...
Valid to	Thursday, February 16, 2023 ...
Subject	US, California, San Jose, Cisco...
Public key	RSA (2048 Bits)
Public key parameters	05 00
Subject Alternative Name	DNS Name=cisco-images.cisco...
Subject Key Identifier	b18ceccd49a5dfd743e0a60f7...
Enhanced Key Usage	Server Authentication (1.3.6...
SCT List	v1, adf7bafa7cff10c88b9d3d9...
Key Usage	Digital Signature, Key Encipher...
Thumbprint	86f51f44a3c93ff68c4b0af1da...

Original Certificate

Field	Value
Version	V3
Serial number	40017f044e5f9214333d982de...
Signature algorithm	sha256RSA
Signature hash algorithm	sha256
Issuer	HydrantID Server CA O1, Hyd...
Valid from	Wednesday, February 16, 20...
Valid to	Thursday, February 16, 2023 ...
Subject	US, California, San Jose, Cisco...
Public key	RSA (2048 Bits)
Public key parameters	05 00
Authority Information Access	[1]Authority Info Access: Acc...
Authority Key Identifier	KeyID=89b89bb69eedfb0e6...
Certificate Policies	[1]Certificate Policy: Policy Ide...
CRL Distribution Points	[1]CRL Distribution Point: Distr...
Subject Alternative Name	DNS Name=cisco-images.cisco...
Subject Key Identifier	b18ceccd49a5dfd743e0a60f7...
Enhanced Key Usage	Server Authentication (1.3.6...
SCT List	v1, adf7bafa7cff10c88b9d3d9...
Key Usage	Digital Signature, Key Encipher...
Thumbprint	0dddb6ce30b00bd75adb198b...

TLS Session Decryption Flow – Client Check



Under the hood: Client Finished

Both client and server exchanged their Finished messages, proving the **session was successfully established**.

```
8.4.161 443 72.163.4.161 443 -> 172.16.136.96 58478 length [57]
8.4.161 443 tls13 session, skipping finished message processing

[DEBUG]: 172.16.136.96 58478 -- 72.163.4.161 443 72.163.4.161 tls13 session both finished seen-handling back flow

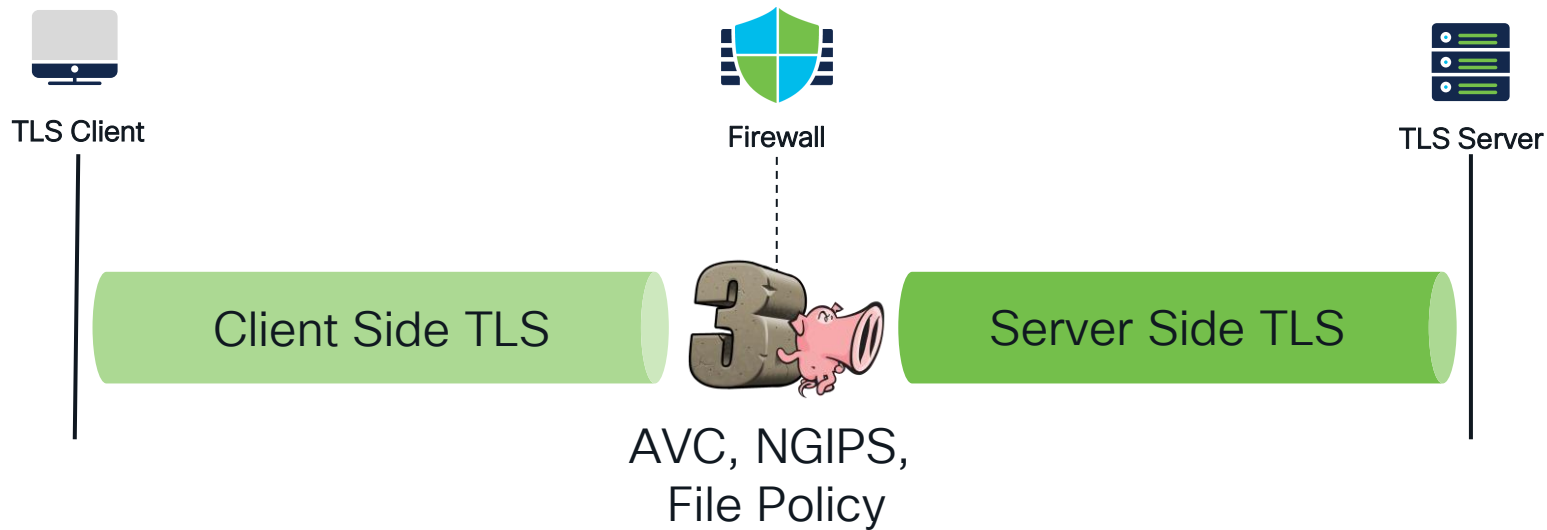
[DEBUG]: 172.16.136.96 58478 -- 72.163.4.161 443 72.163.4.161 Updating connection event: Logging: true action: 6
rule: 1 status: 2 flow_err: XTLS_SUCCESS flags: 341808ac490a144

[DEBUG]: 172.16.136.96 58478 -- 72.163.4.161 443 72.163.4.161 verdict pass
```

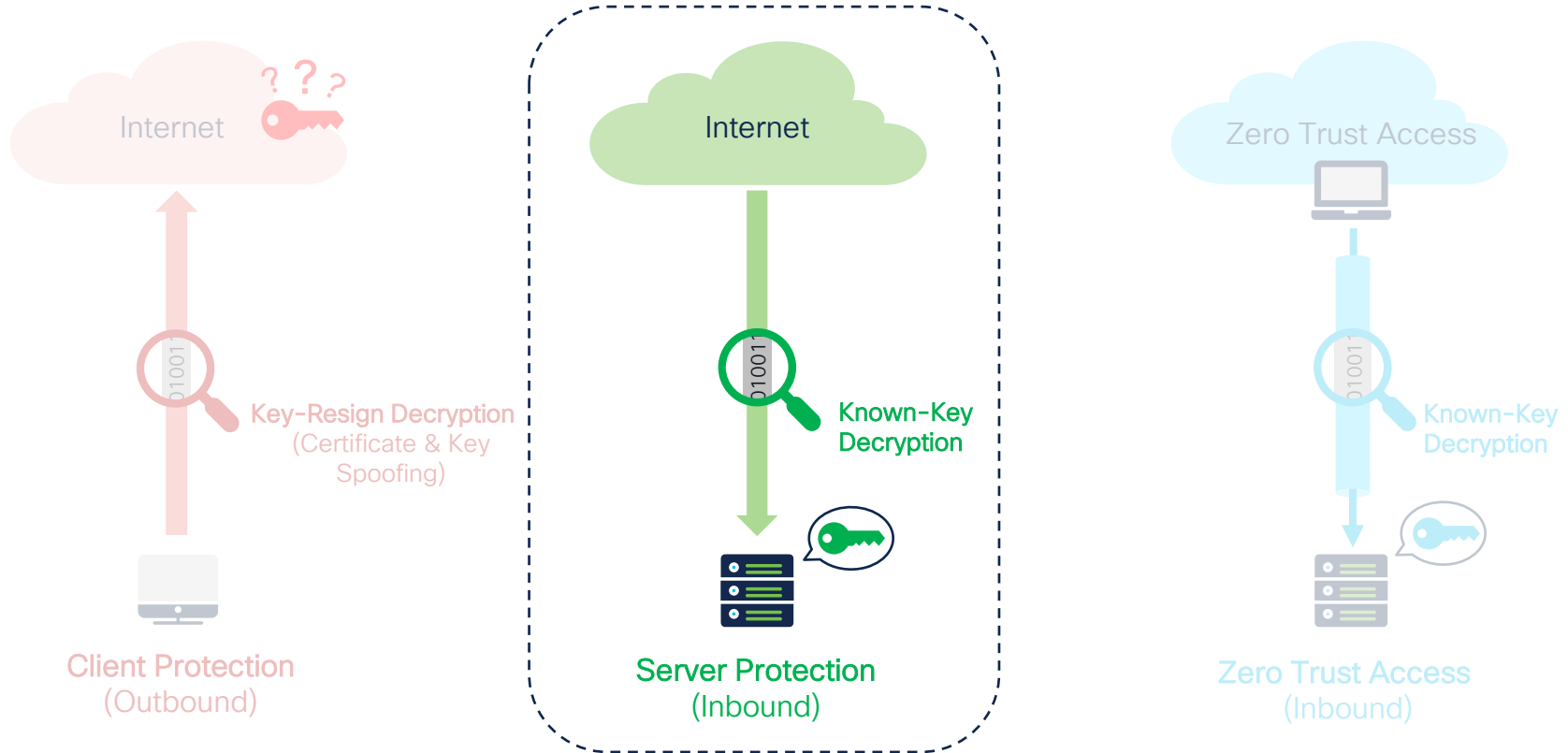
The Firewall updates the details of this crypto session in the **connection logs**.

The firewall is now installed as **Man-in-the-Middle**.

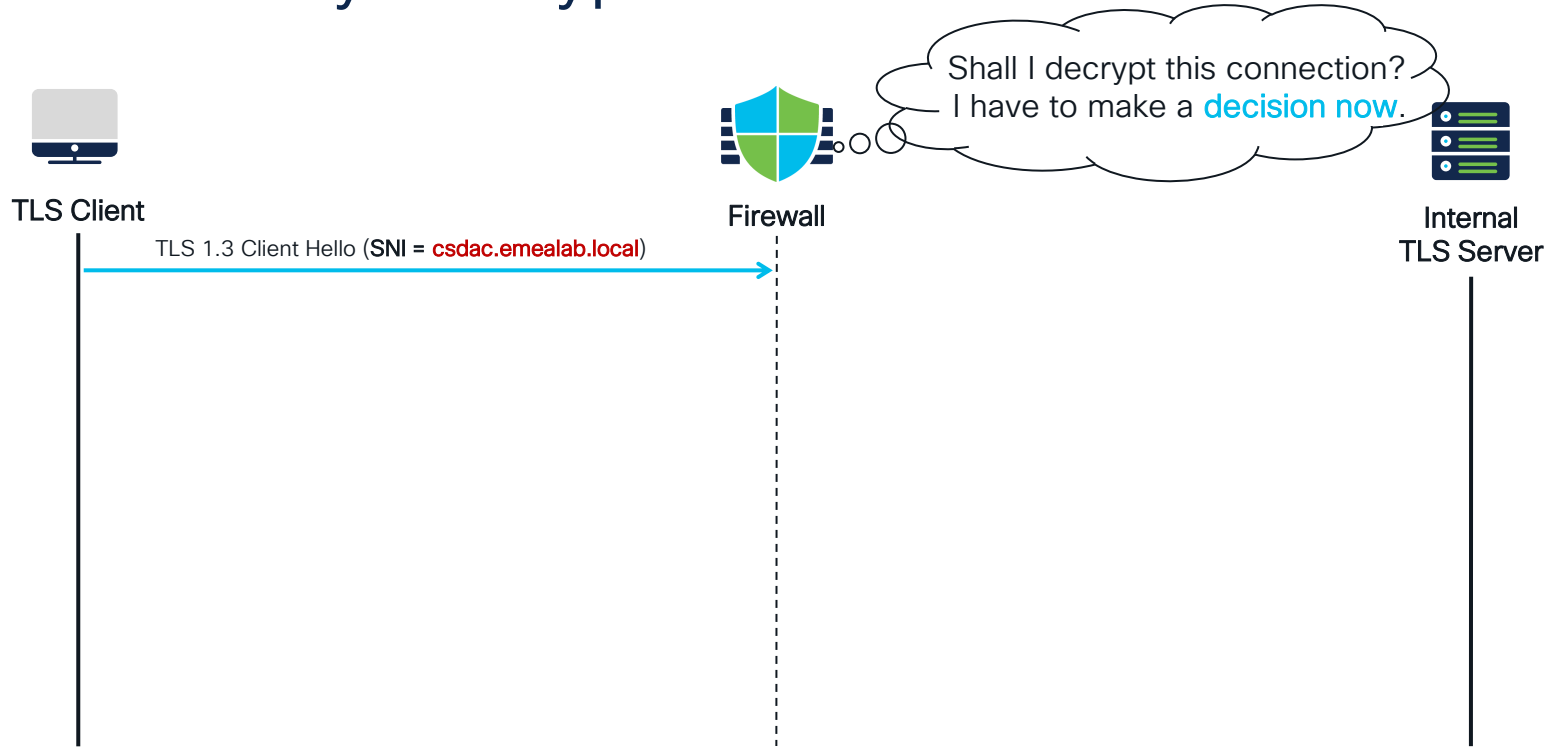
TLS Session Decryption Flow – PIG-in-the-Middle



Three Flavors of TLS Decryption



Known-Key Decryption: Client Hello



Known-Key Decryption: Rule Action

Editing Rule - Decrypt Ingress

Name: Decrypt Ingress ☒ Enabled Move: above rule 2

Action: Decrypt - Known Key with CSDAC-Certificate, FMC-Certifica

Zones Networks VLAN Tags Users Applications Ports Category Certificate DN Cert Status Cipher Suite Version Logging

Available Networks: Search by name or value

Networks: any

Geolocation: Add to Source Add to Destination

Source Networks (1): any

Destination Networks (1): HTTPS_Servers

Enter an IP address Add

Cancel Save

Select decrypt with **Known Key** action.

Known-Key Decryption: Initial Traffic Matching

Editing Rule - Decrypt Ingress

Name: Decrypt Ingress ☒ Enabled Move: above rule 2

Action: Decrypt - Known Key with CSDAC-Certificate, FMC-Certifica

Zones Networks VLAN Tags Users Applications Ports Category Certificate DN Cert Status Cipher Suite Version Logging

Available Networks +

Source Networks (1): any

Destination Networks (1): HTTPS_Servers

Enter an IP address Add Enter an IP address Add

any
IPv4
IPv6
any-ipv4
any-ipv6
ANY
AWS-HOST2
AWS-Host

In most cases you will use **Network and Port** conditions to match the traffic, as you protect your own servers.

You cannot use **DN and Certificate** conditions.

Best Practice: set network conditions to match traffic only to the servers you want to decrypt. Otherwise, you will likely experience late do-not-decrypt connection drops.

Known-Key Decryption: Discrepancy Between Network Conditions and Internal Certificates

REFERENCE

```
[TRACE]: 172.16.136.96 56041 -- 172.16.134.97 443 client_hello: len [560]
[INFO]: 172.16.136.96 56041 -- 172.16.134.97 443 Evaluating client hello modify decision for flow
[DEBUG]: 172.16.136.96 56041 -- 172.16.134.97 443 Rule 3 (Decrypt Ingress) match assumed based on destination
endpoint
[INFO]: 172.16.136.96 56041 -- 172.16.134.97 443 Matched rule: Decrypt Ingress, decision is modify
```

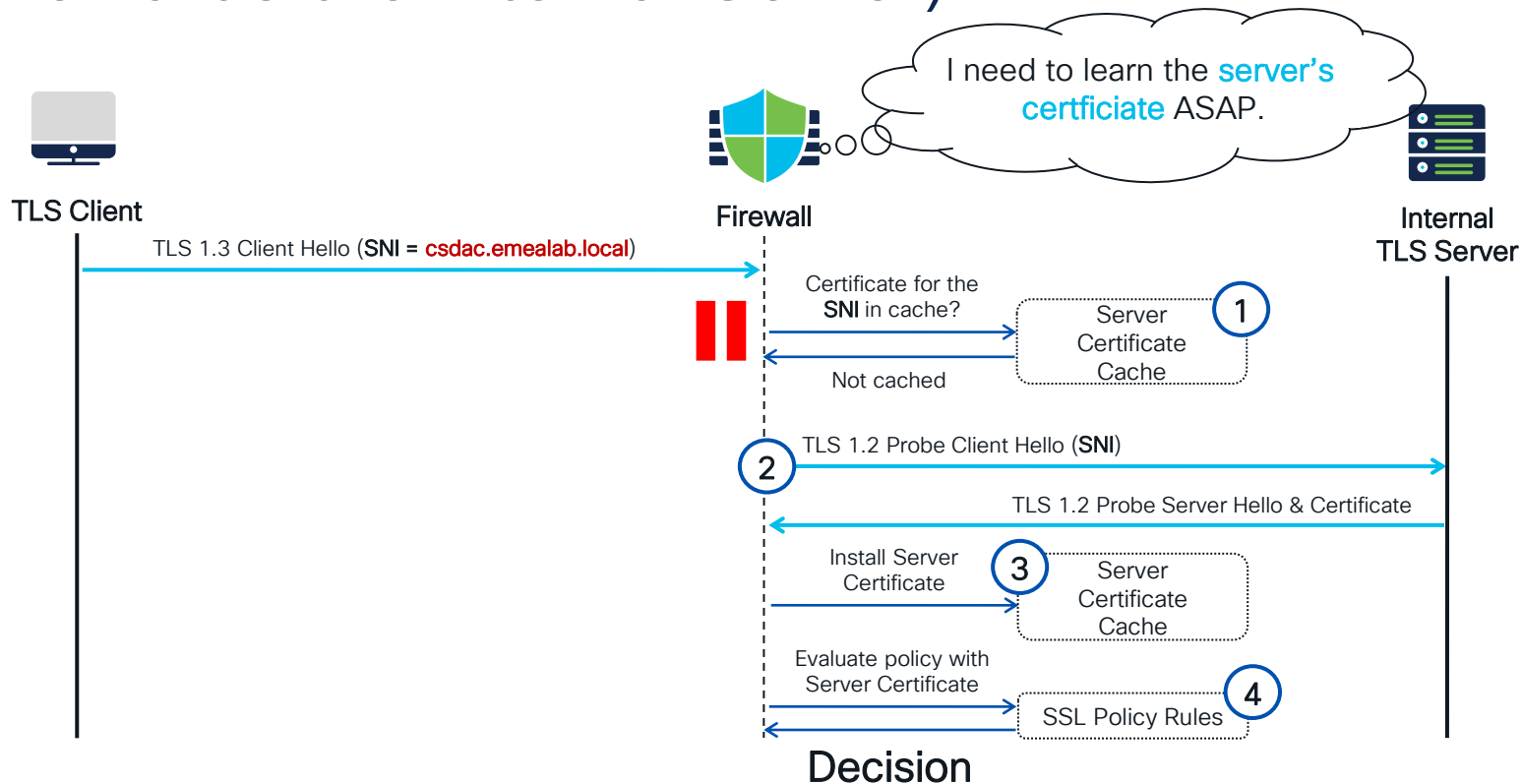
The target server certificate is not in the cache, hence the firewall matches Client Hello based on network conditions.

```
[TRACE]: 172.16.136.96 56041 -- 172.16.134.97 443 certificate: len [2160]
[DEBUG]: 172.16.136.96 56041 -- 172.16.134.97 443 sni matches the certificate
[TRACE]: 172.16.136.96 56041 -- 172.16.134.97 443 original certificate added to cache
[DEBUG]: 172.16.136.96 56041 -- 172.16.134.97 443 Rule 3 (Decrypt Ingress) did not match internal certificate
[INFO]: 172.16.136.96 56041 -- 172.16.134.97 443 Matched default policy action DoNotDecrypt
[ERROR]: 172.16.136.96 56041 -- 172.16.134.97 443 client hello modified when it should not have been
[TRACE]: 172.16.136.96 56041 -- 172.16.134.97 443 policy verdict [BlockWithReset]
```

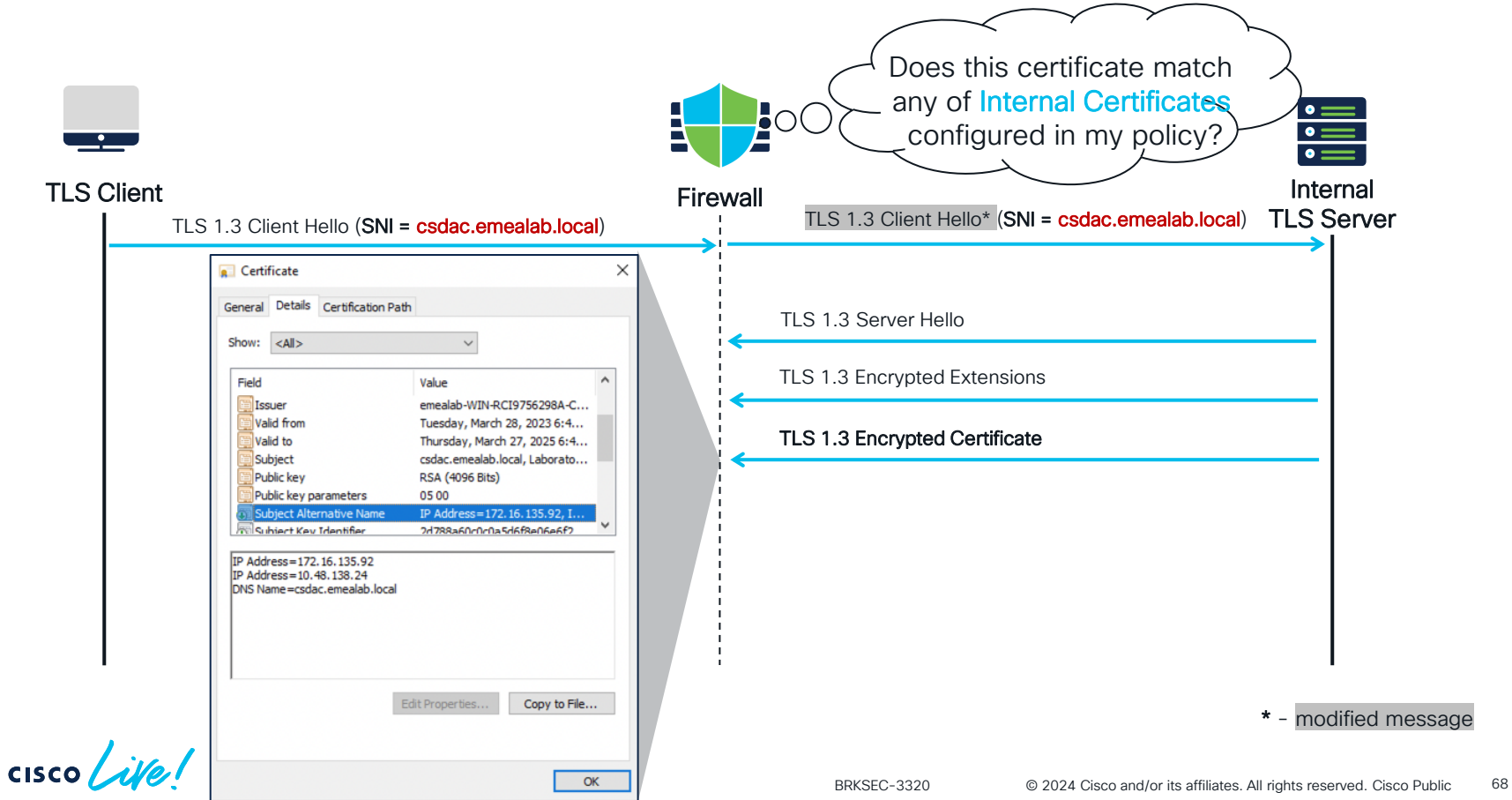
Once the server sends back the certificate it turns out it doesn't match any of the internal certs configured for this rule.

The verdict contradicts the initial decision and firewall resets the connection.

Known-Key Decryption: TLS 1.2 Probe (Towards the Internal Server)



Known-Key Decryption: Server Response



Known-Key Decryption: Selecting The Internal Certificates with Private Key

Editing Rule - Decrypt Ingress

Name: Decrypt Ingress ☒ Enabled Move: above rule 2

Action: Decrypt - Known Key with CSDAC-Certificate, FMC-Certificate

Available Networks: Search by name or value

Source Networks (1): any

Select Internal Certificate Objects

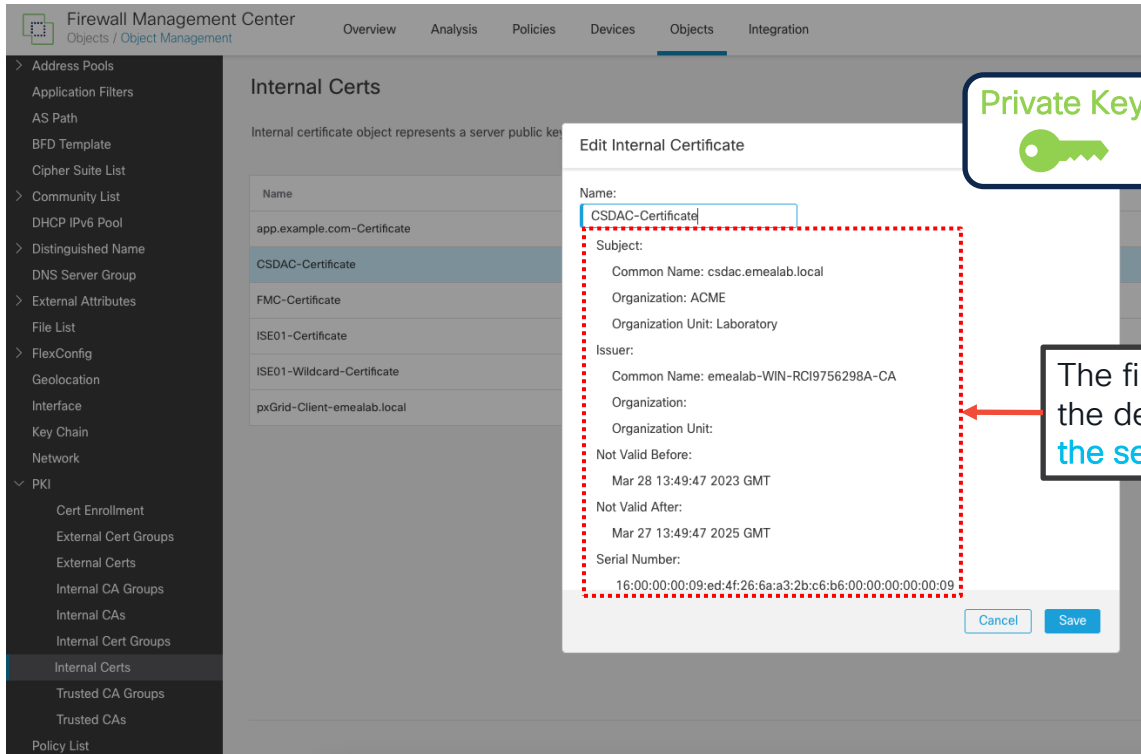
Available Certificates: Search by name or value

Selected Certificates (2): CSDAC-Certificate, FMC-Certificate

You can select multiple Internal Certificates with Private Key in a single rule.

Cancel Save

Known-Key Decryption: Server's Certificate with Private Key

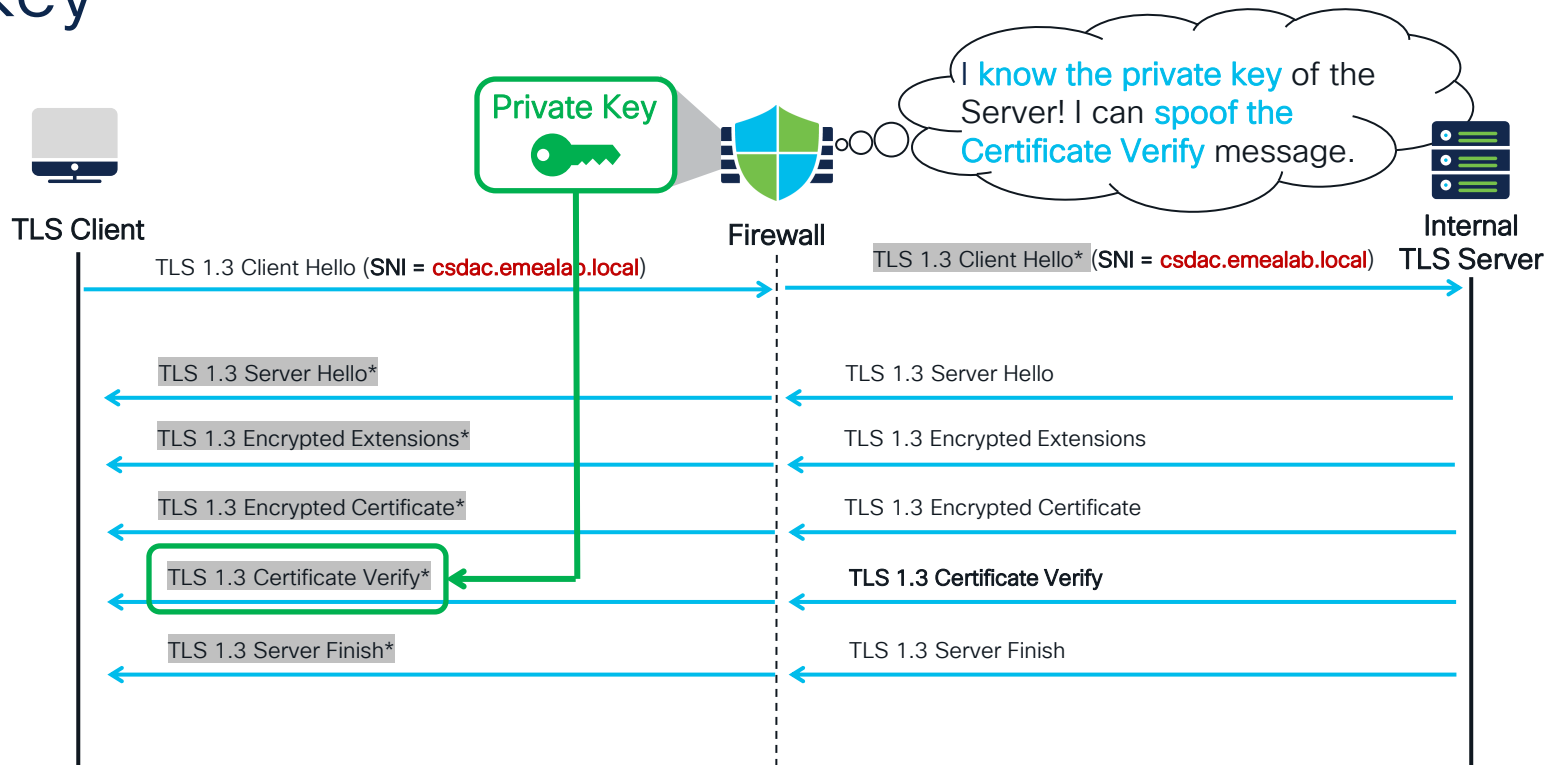


Private Key

The internal server's certificate must be imported with its **private key**.

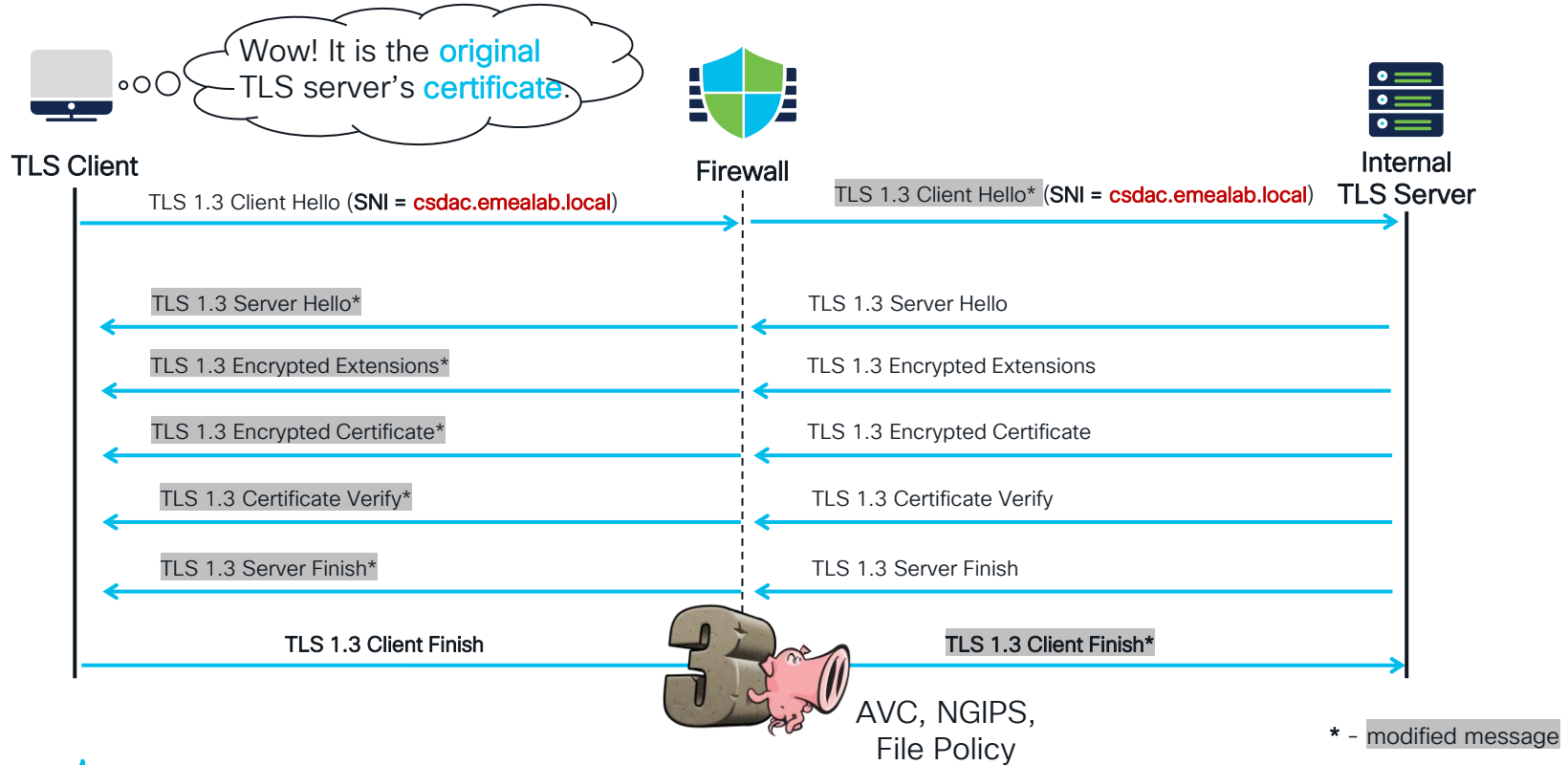
The firewall **compares Internal Certificates** set in the decryption rules **with certificate returned by the server**.

Known-Key Decryption: Server Verify with Private Key



* - modified message

Known-Key Decryption: Client Check



Now It's Time for a Quiz 😊

- You can win a **bouncy ball**
- There will be **3 quizzes** during this session
- Each quiz has **2 questions**
- You will have **30, 45 or 60 seconds** for a question



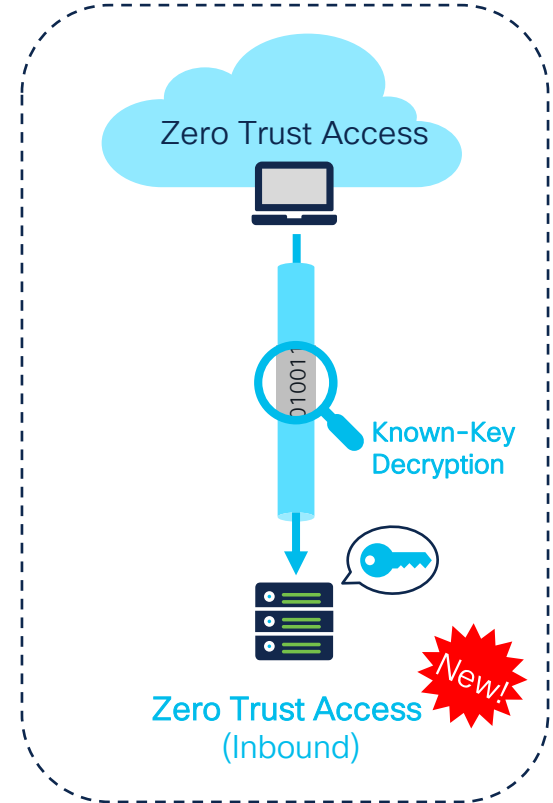
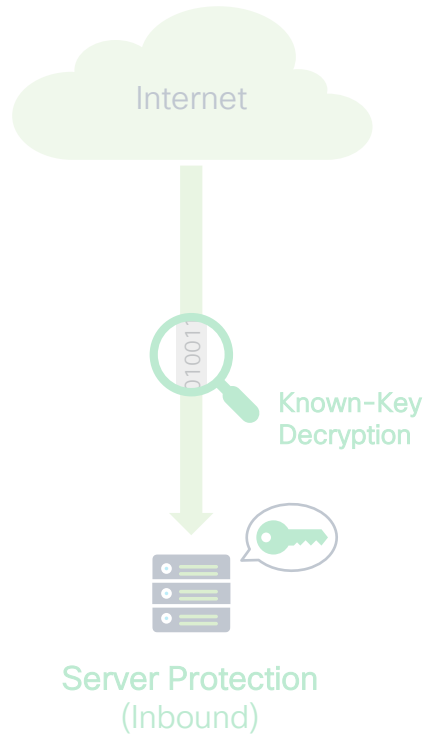
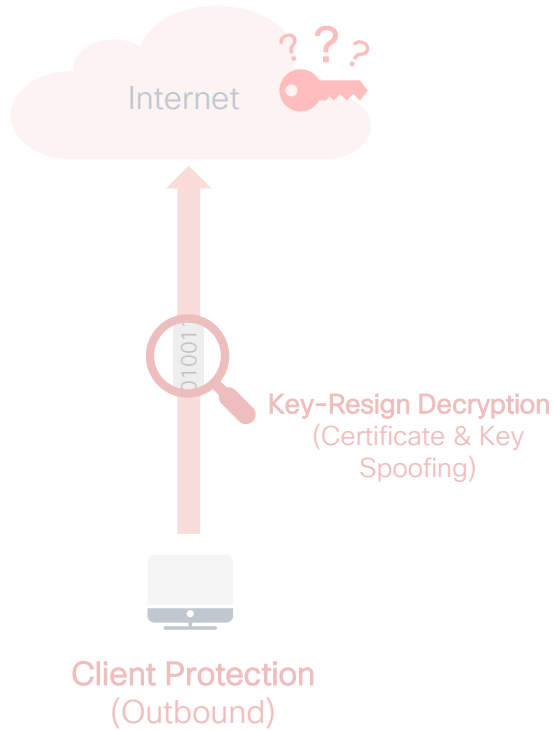
A champion of a quiz also wins a **QUIZ IMMUNITY** until the end of this session.

QUIZ 1: Decryption on Cisco Secure Firewall

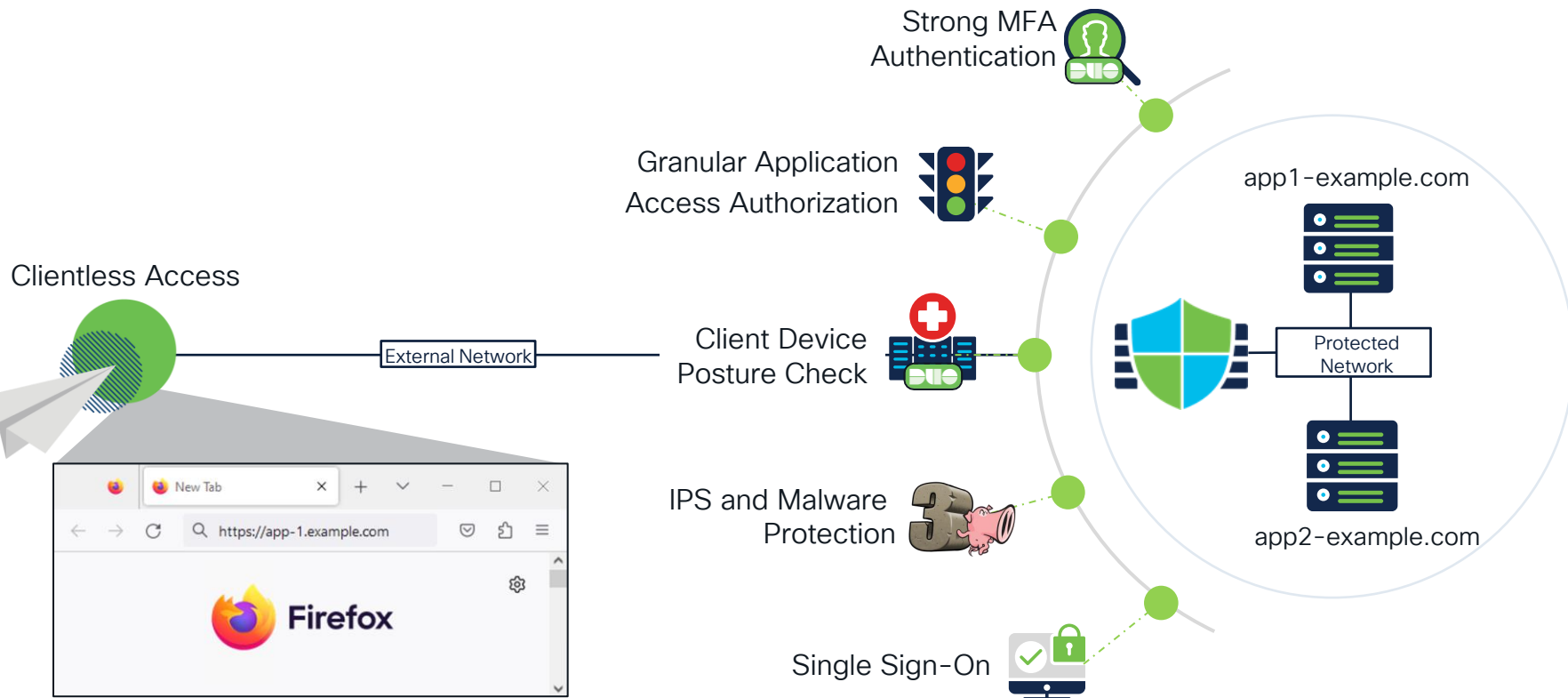
Join at
slido.com
#2592 848



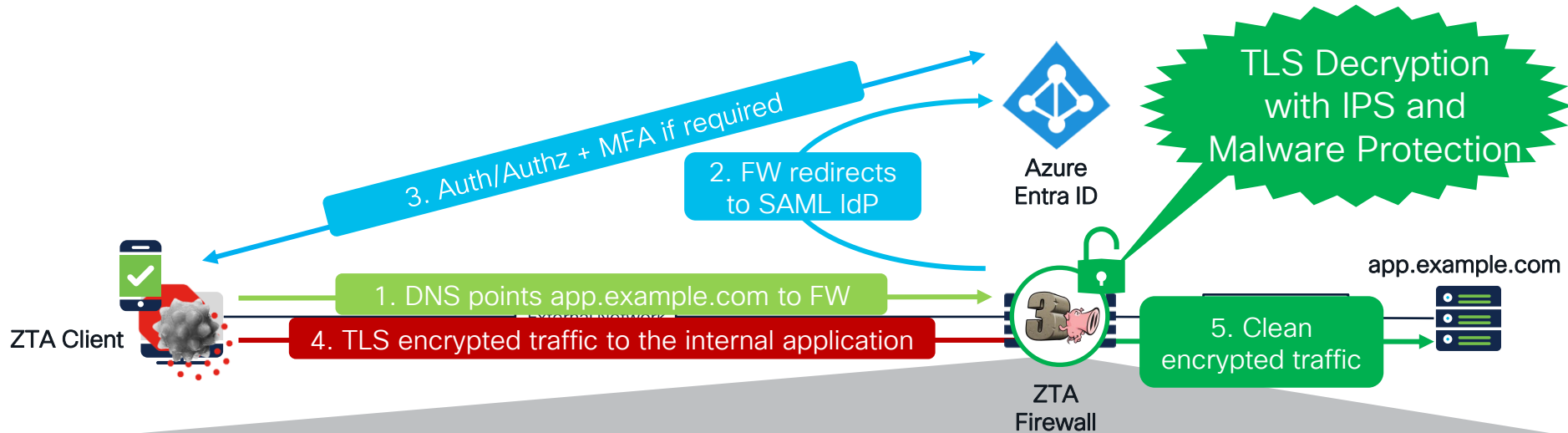
Three Flavors of TLS Decryption



Zero Trust Access (ZTA) - Overview



Zero Trust Access – Successful Auth/Authz



Firewall Management Center
Analysis / Unified Events

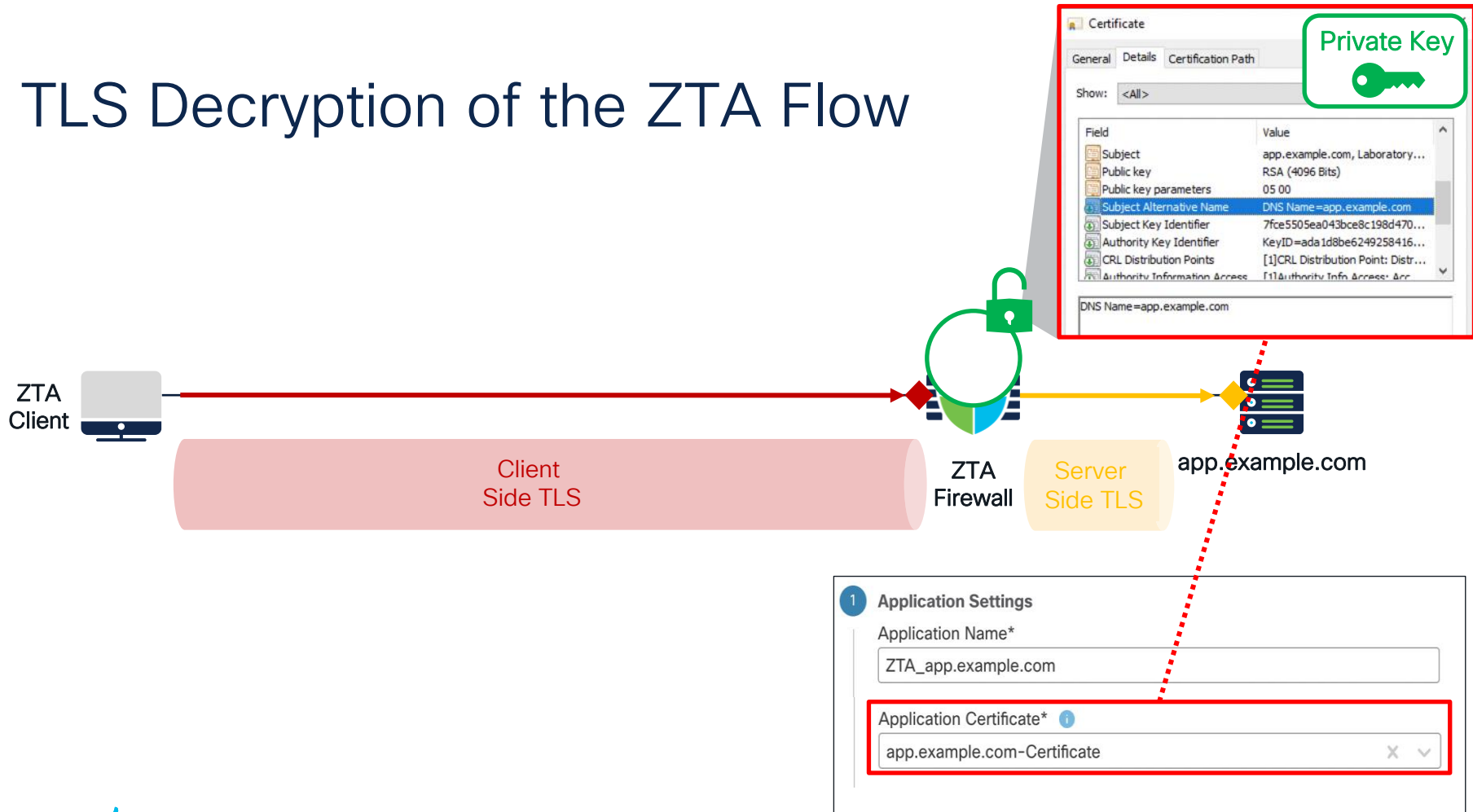
Overview Analysis Policies Devices Objects Integration Deploy

Event Type: Intrusion Malware

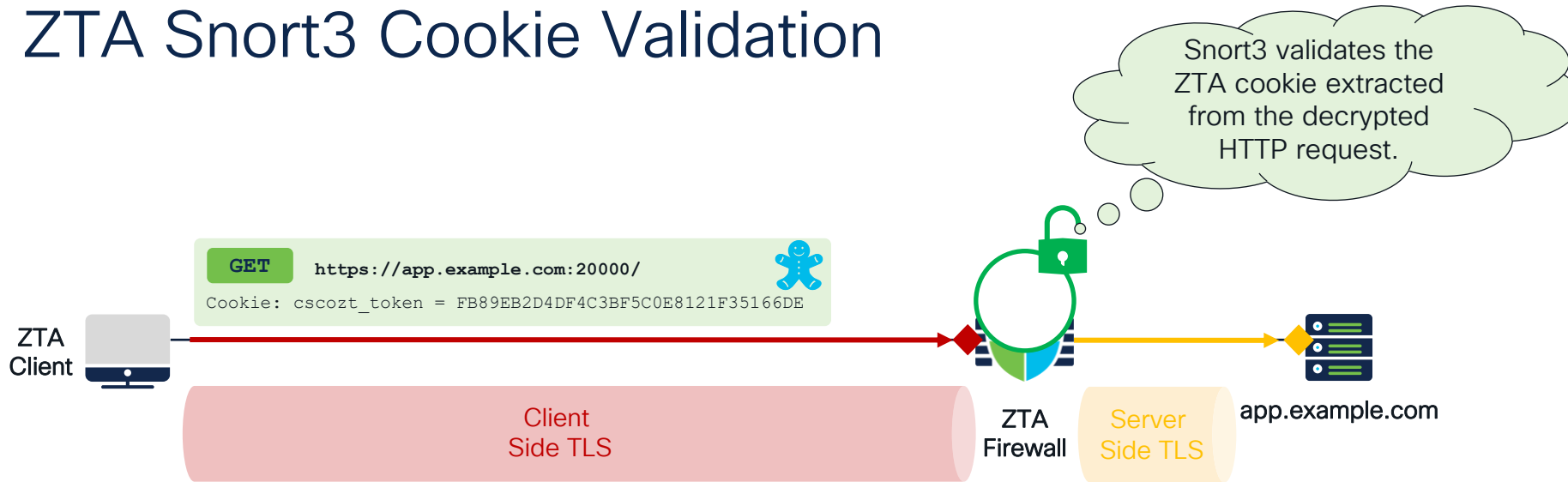
Showing all 27 events (8 19)

Time	Event Type	Action	Source IP	Destination IP	Destination Port / ICMP Code	Source User
2023-10-03 09:59:30	Malware	Malware Block	172.16.135.101	172.16.134.96	443 (https) / tcp	dclouduser@cisco.netsectmesdcloud.onmicrosoft.com
2023-10-03 09:59:28	Intrusion	Alert	172.16.135.101	172.16.134.96	443 (https) / tcp	dclouduser@cisco.netsectmesdcloud.onmicrosoft.com

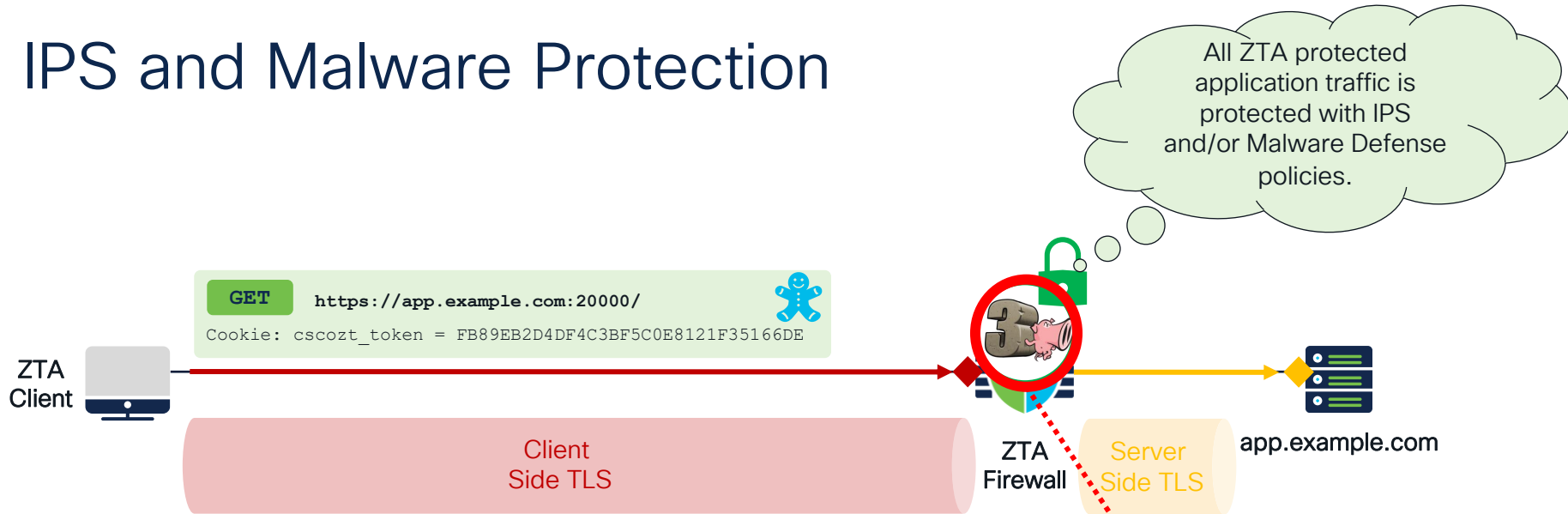
TLS Decryption of the ZTA Flow



ZTA Snort3 Cookie Validation



IPS and Malware Protection



Security Controls (Optional)

Private applications can be subject to inspection using a selected Intrusion or Malware and File policy.

Intrusion Policy

ZTAA IPS Policy

x v +

Variable Set

ZTAA-Variable-Set

x v +

Malware and File Policy

ZTAA File Policy

x v +

These are default settings for all private applications. It can be overridden at an Application or Application Group level.

DEMO: Zero Trust Access Flow with IPS and Malware Protection

It Is Not an Easy World for a Man (-in-the-Middle)

QUIC

Certificate Pinning

DNS over HTTPs

Encrypted SNI

Encrypted Client Hello

0-RTT

TLS 1.3



Challenges Posed by QUIC

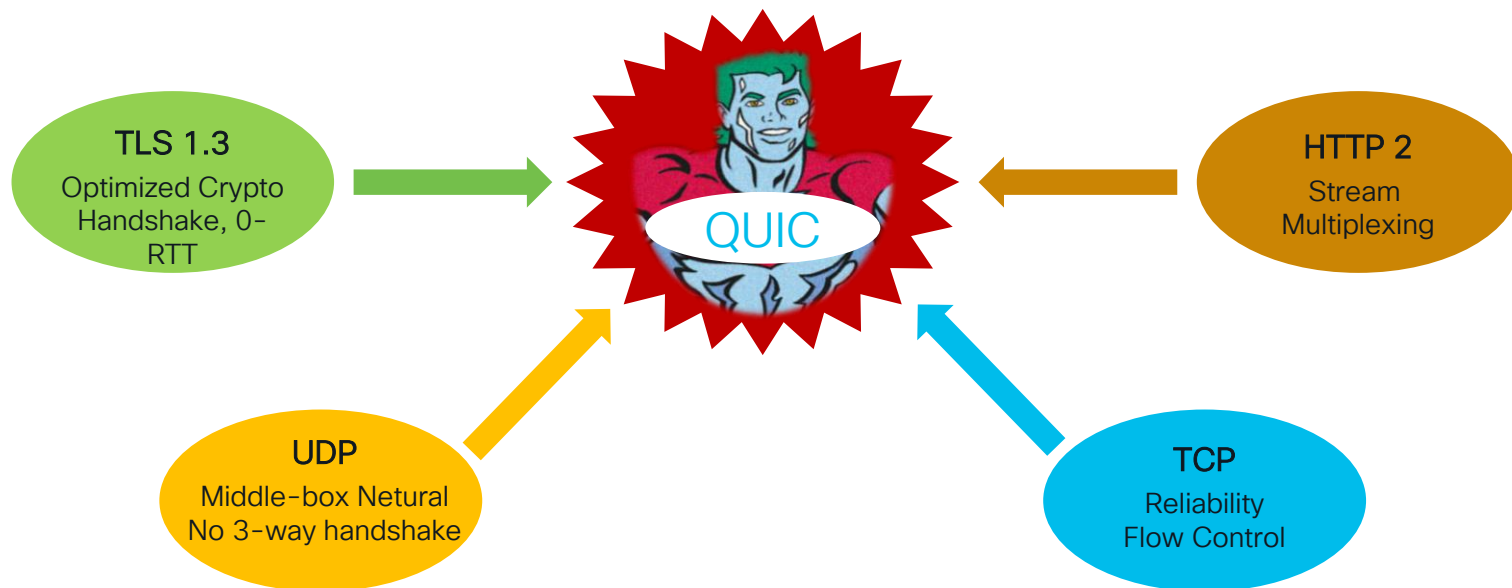
Why QUIC?

- **Take away flow-control from network and give it to applications**
 - The application is no longer reliant on TCP fair back-off algorithm
 - Applications can now aggressively compete for bandwidth (check BRKSPM-2024)
- **Low latency connection establishment and seamless roaming**
 - 0-RTT connection (based on TLS 1.3)
 - seamless migration between IPv4 / IPv6 addresses
- **Overcome head of line blocking issues with TLS over TCP**
 - HTTP payload encrypted at TLS level spread across multiple TCP segments – decrypt possible only once all segments received
 - Losing one packet may result in holding a huge blob of data
- **Counteract against network ossification slowing down new protocol adoption**
 - QUIC is completely opaque for dumb middle-boxes
 - QUIC is extremely complex to decrypt and inspect (remember TLS 1.3 adoption issues?)

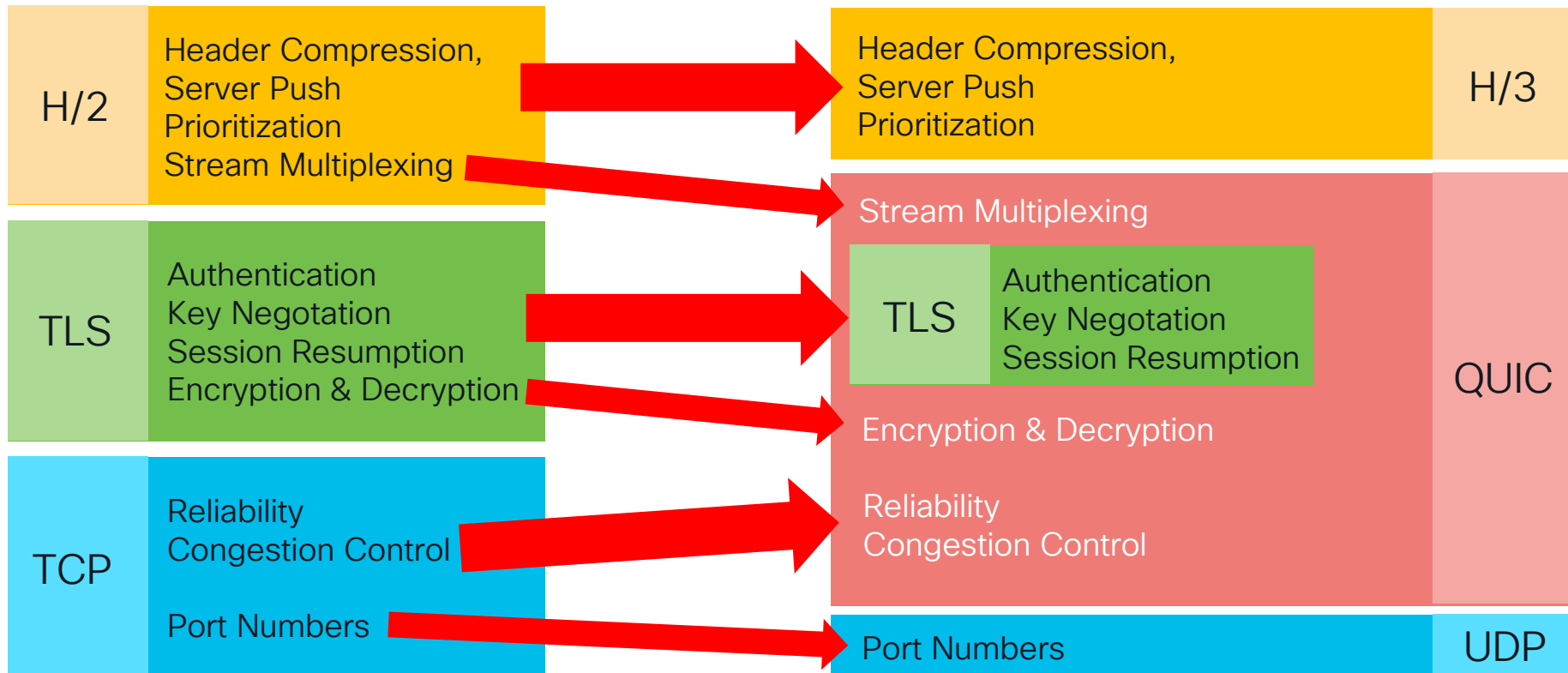
What is QUIC

- IETF proposed standard (RFC 9000) – started at Google in 2014
 - Take away flow-control from network and give it to applications
 - 0-RTT connection establishment and seamless roaming
 - Improved content delivery e.g. overcome head of line blocking issues with TLS over TCP
 - Counteract against network ossification slowing down new protocol adoption
- QUIC is a new **secure transport protocol** – an underlay for practically any application and protocol (HTTP3, SMB, BGP, SSH, DNS, etc...)

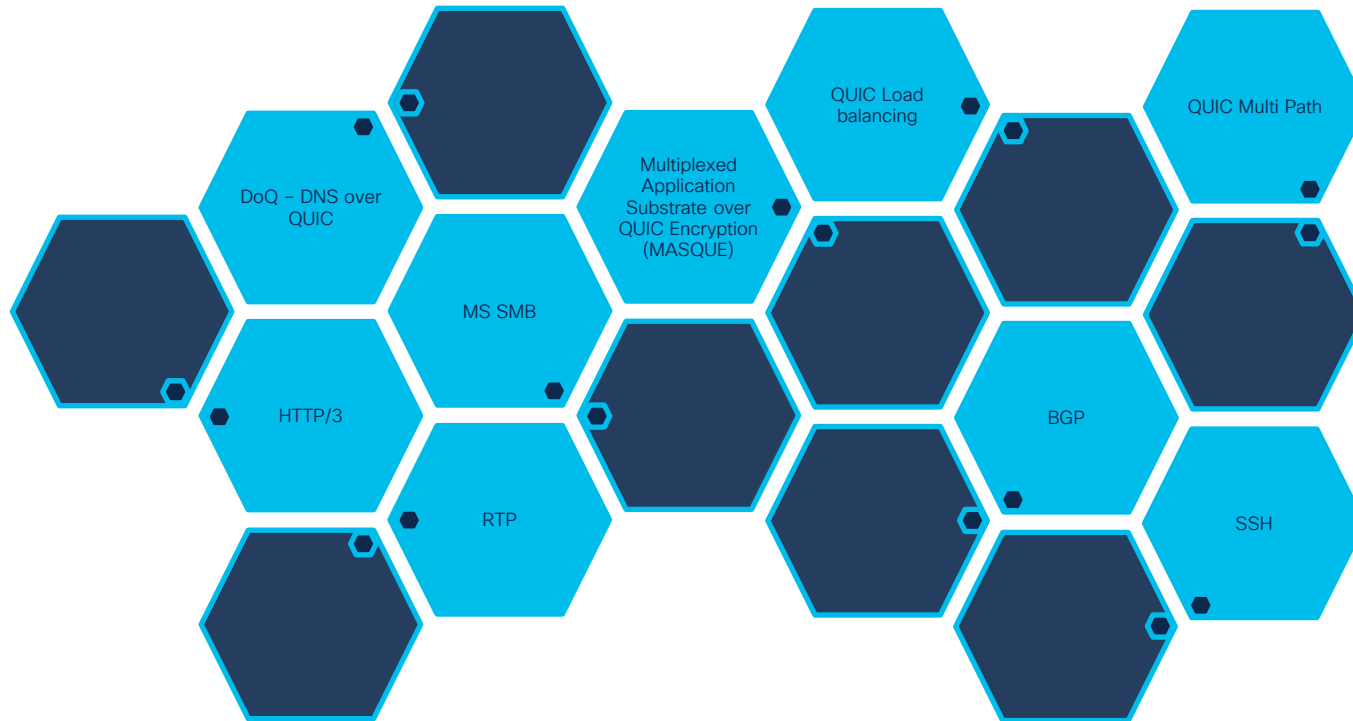
QUIC Employs Best Features of the Existing Protocols



Where Does QUIC Fit in the Protocol Stack?



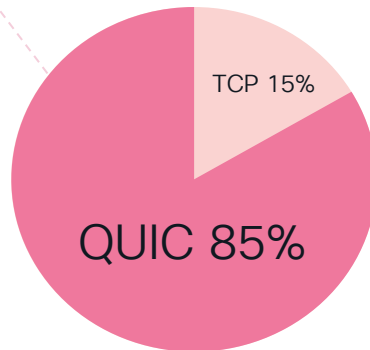
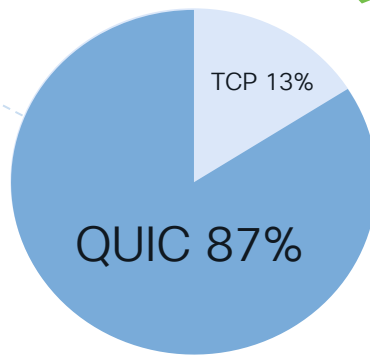
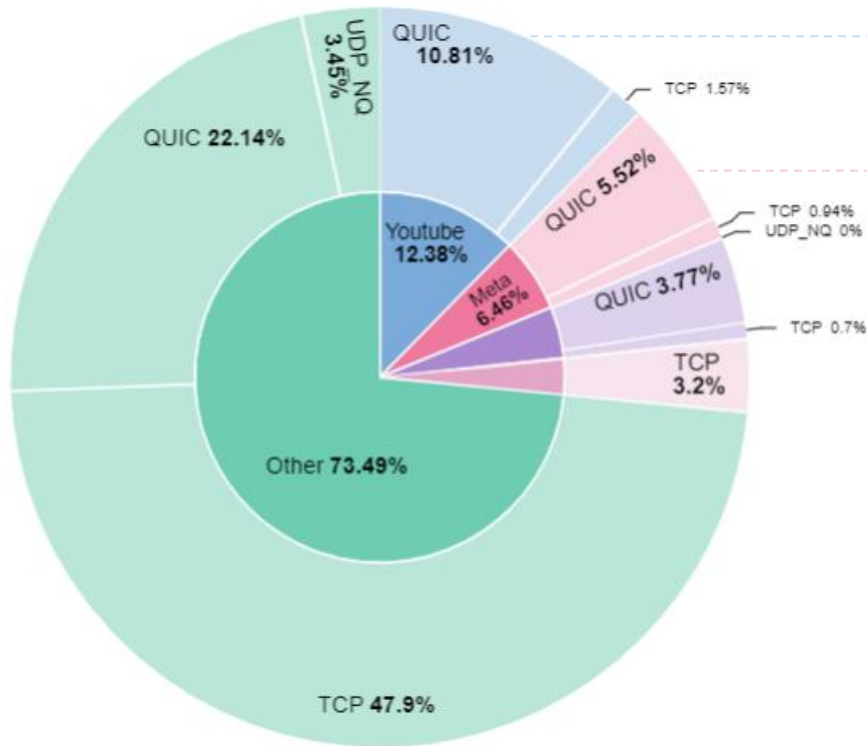
QUIC is Not Only HTTP3



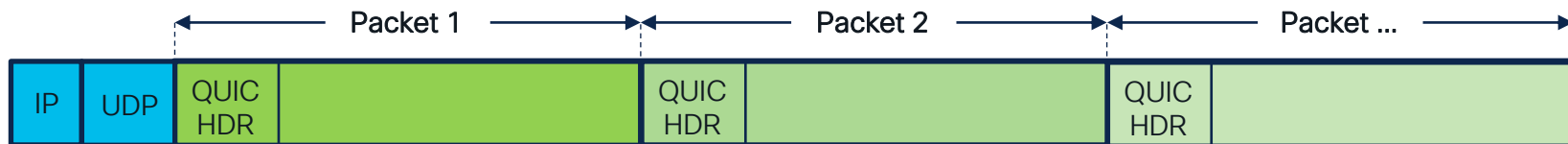
The pattern persists worldwide into 2023 (Based on EU - QUIC is 42%)

(Source: BRKSPM-2024)

For more stats on QUIC check:
BRKSPM-2024



An Anatomy of a QUIC Datagram – Packets



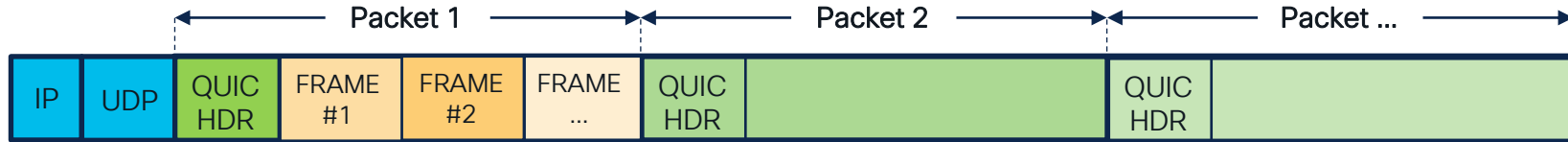
No.	Time	Source	SPORT	Destination	DPORT	Protocol	Length	Info
262	0.012218	172.217.16.10	443	192.168.10.22	49287	HTTP3	1292	Protected Payload (KP0)
<div>> Frame 262: 1292 bytes on wire (10336 bits) on interface en0, id 0</div> <div>> Ethernet II, Src: Cisco_f5:28:c5 (bc:d0:74:5c:c3:0d), Dst: Apple_5c:c3:0d (bc:d0:74:5c:c3:0d)</div> <div>> Internet Protocol Version 4, Src: 172.217.16.10, Dst: 192.168.10.22</div> <div>> User Datagram Protocol, Src Port: 443, Dst Port: 49287</div> <div>> QUIC IETF</div> <div>> QUIC IETF</div> <div>> QUIC IETF</div>								

QUIC Packet 1 – Handshake

QUIC Packet 2 – Initial Packet

QUIC Packet 3 – 1-RTT (data) with HTTP 3 Stream

An Anatomy of a QUIC Datagram – Frames



No.	Time	Source	SPORT	Destination	DPORT	Protocol	Length	Info
262	0.012218	172.217.16.10	443	192.168.10.22	49287	HTTP3	1292	Protected Payload (KP0)

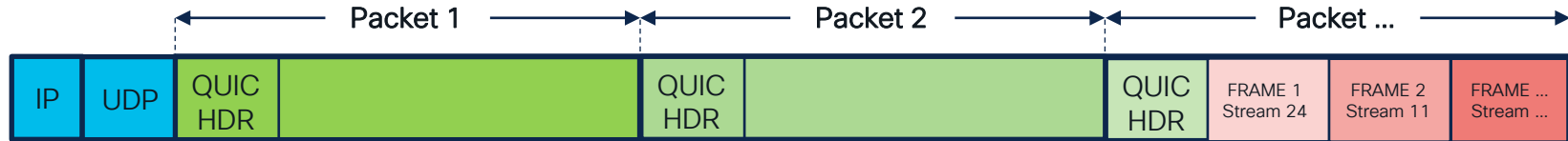
> Frame 262: 1292 bytes on wire (10336 bits), 1292 bytes captured (10336 bits) on interface en0, id 0 > Ethernet II, Src: Cisco_f5:28:c9 (08:4f:a9:f5:28:c9), Dst: Apple_5c:c3:0d (bc:d0:74:5c:c3:0d) > Internet Protocol Version 4, Src: 172.217.16.10, Dst: 192.168.10.22 > User Datagram Protocol, Src Port: 443, Dst Port: 49287								
> QUIC IETF								
> QUIC Connection information [Packet Length: 950] 1... .. = Header Form: Long Header (1) .1.. .. = Fixed Bit: True ..00 .. = Packet Type: Initial (0) 00.. = Reserved: 000 = Packet Number Length: 1 bytes (0) Version: 1 (0x00000001) Destination Connection ID Length: 0 Source Connection ID Length: 8 Source Connection ID: fda3a9295465ed94 Token Length: 0 Length: 932 Packet Number: 1 Payload: 3426ba515607ec6a65347f9191ae6a80ae59e41150b07354f31185ca8f4fdad1d2d10a16...								
> ACK > CRYPTO > PADDING Length: 809								
> QUIC IETF > QUIC IETF								

QUIC Packet 1 in this UDP datagram

Header describing packet type, length etc...

- ACK frame (flow control)
- CRYPTO frame (TLS 1.3 Server Hello)
- PADDING frame

An Anatomy of a QUIC Datagram – Streams



Four **stream frames** encapsulated in a single QUIC packet.

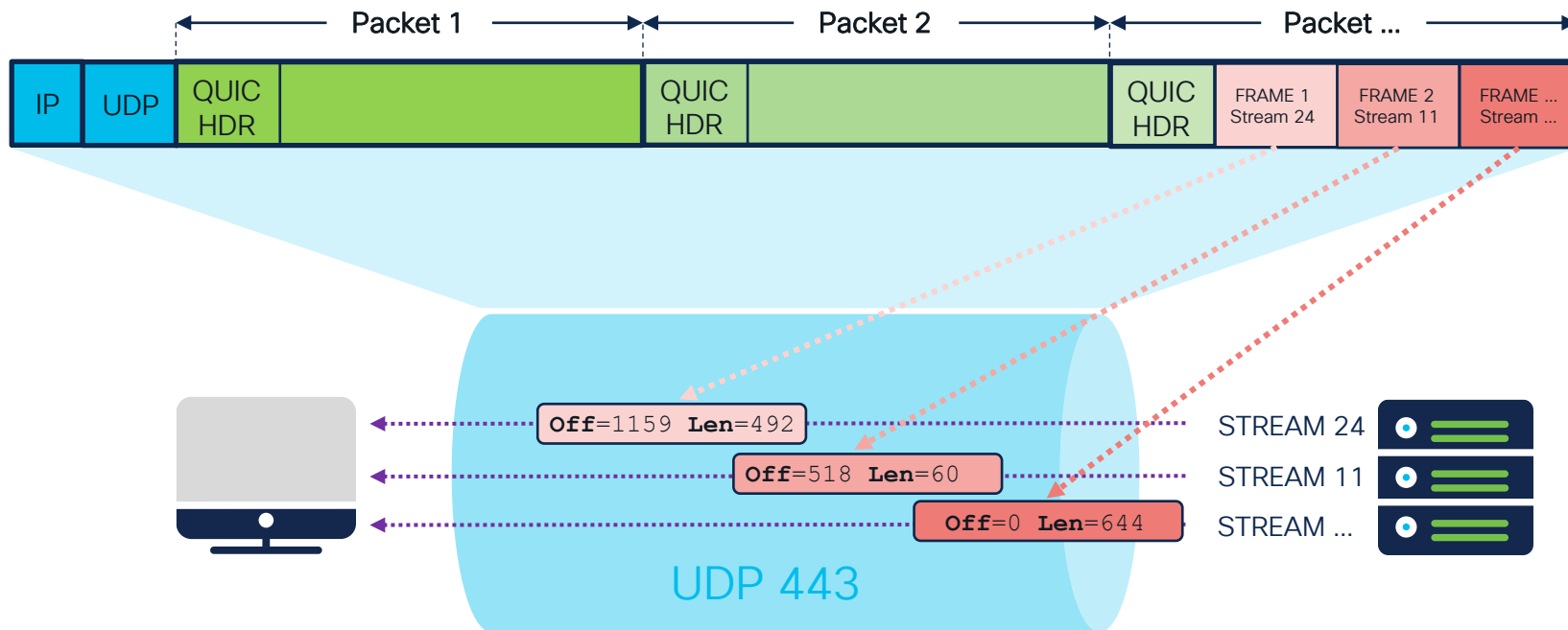
```

↓ 416 0.000177 142.250.203.195 443 192.168.10.22 63419 HTTP3 1292 Protected Payload (KP0)
1292 bytes on wire (10336 bits), 1292 bytes captured (10336 bits) on interface en0, id 0
, Src: Cisco_f5:28:c9 (08:4f:a9:f5:28:c9), Dst: Apple_5c:c3:0d (bc:d0:74:5c:c3:0d)
Protocol Version 4, Src: 142.250.203.195, Dst: 192.168.10.22
> User Datagram Protocol, Src Port: 443, Dst Port: 63419
  > QUIC IETF
    > QUIC Connection information
      [Packet Length: 1250]
    > QUIC Short Header PKN=20
      > STREAM id=24 fin=1 off=1159 len=492 dir=Bidirectional origin=Client-initiated
      > STREAM id=11 fin=0 off=518 len=60 dir=Unidirectional origin=Server-initiated
      > STREAM id=12 fin=1 off=0 len=644 dir=Bidirectional origin=Client-initiated
      > STREAM id=11 fin=0 off=578 len=17 dir=Unidirectional origin=Server-initiated
    > Hypertext Transfer Protocol Version 3
    > Hypertext Transfer Protocol Version 3
    > Hypertext Transfer Protocol Version 3
    > Hypertext Transfer Protocol Version 3
  
```

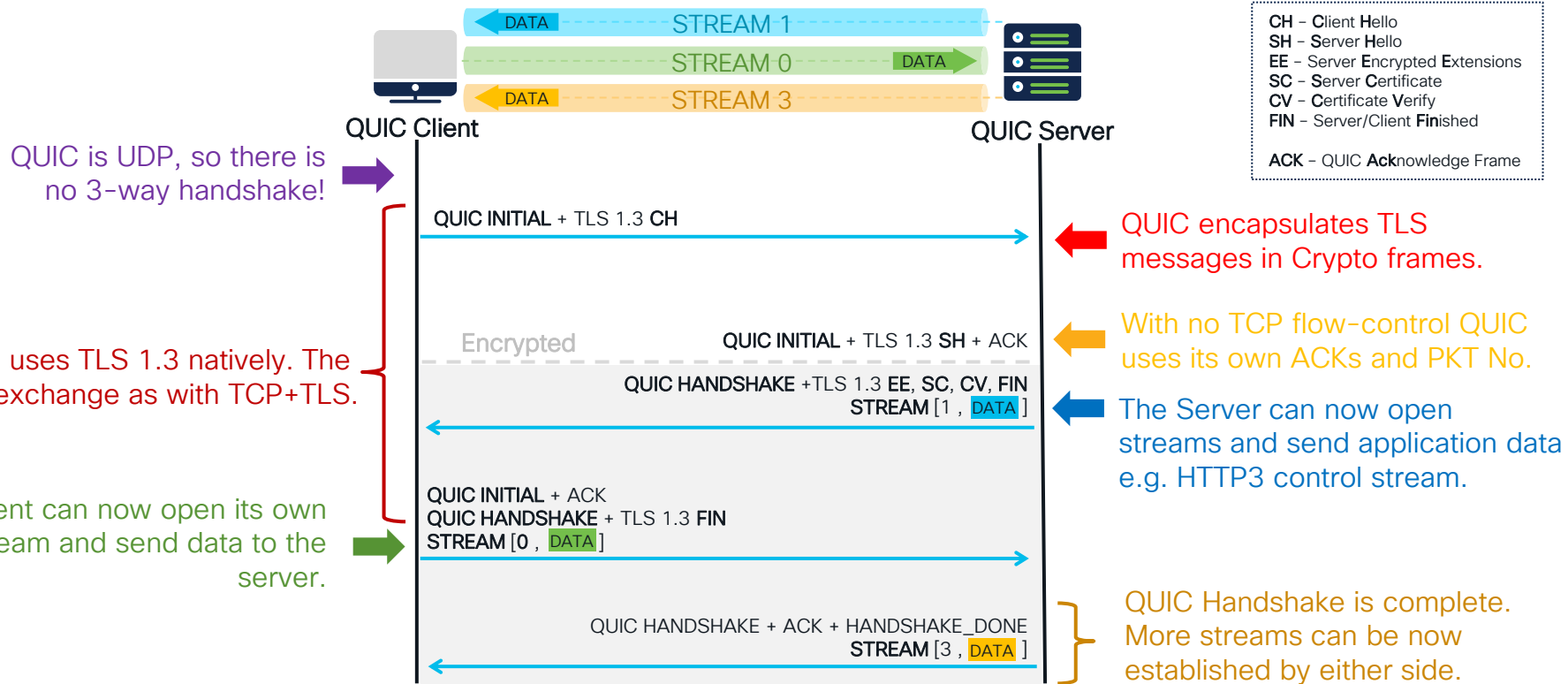
Each QUIC Stream has its **individual flow control** parameters.

All streams in this packet carry **HTTP 3 payloads**.

An Anatomy of a QUIC Datagram – Streams



Understanding a QUIC Session Handshake



The More You Invest the More You See...

Dumb middle-box:



A bit smarter device (reversing QUIC **header protection** aka. weak encryption):

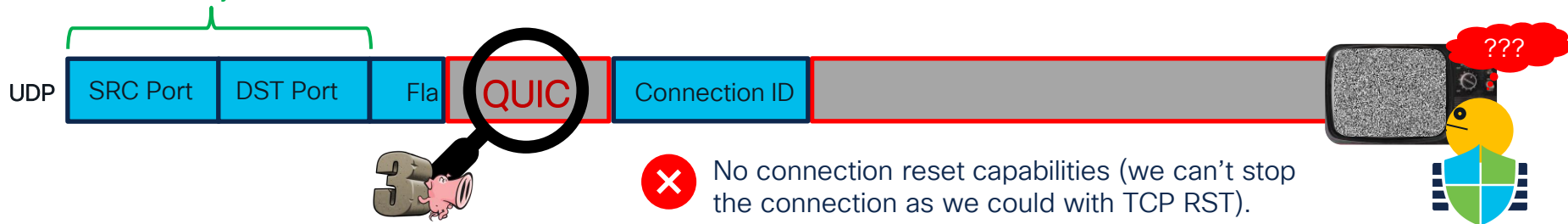


Full QUIC decryption (reversing QUIC **header protection** and **payload encryption**):



The Firewall is a “Dumb Middle-Box”... for now.

✓ We can configure firewall to block/allow by IP and/or UDP 443



✓ Snort3 detects QUIC with AppID and EVE on any UDP port.

✗ The firewall, has no visibility into the state of a connection i.e:
– Is it new? Existing? Migrated? 0-RTT resumed? Etc...

✗ We can't say if this connection is idle or was already closed by QUIC client/server.

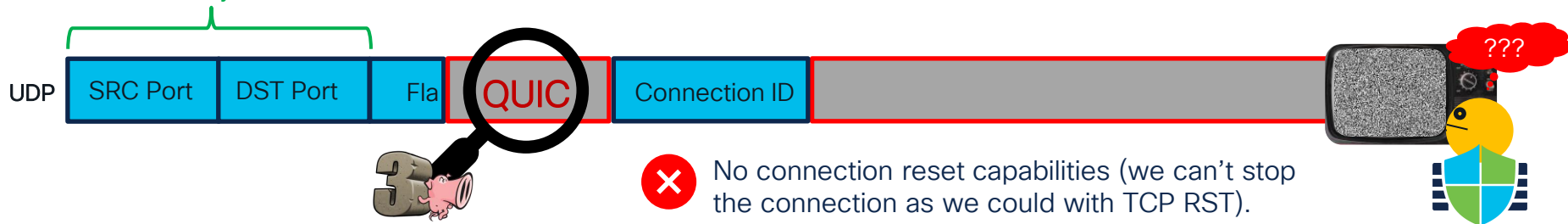
✗ No connection reset capabilities (we can't stop the connection as we could with TCP RST).

✗ Selective packet drop, won't back-off the application, quite contrary... the app may get more aggressive!

✗ No IPS/Malware protection...

The Firewall is a “Dumb Middle-Box”... for now.

✓ We can configure firewall to block/allow by IP and/or UDP 443



✓ Snort3 detects QUIC with AppID and EVE on any UDP port.

✗ The firewall, has no visibility into the state of a connection i.e:
– Is it new? Existing? Migrated? 0-RTT resumed? Etc...

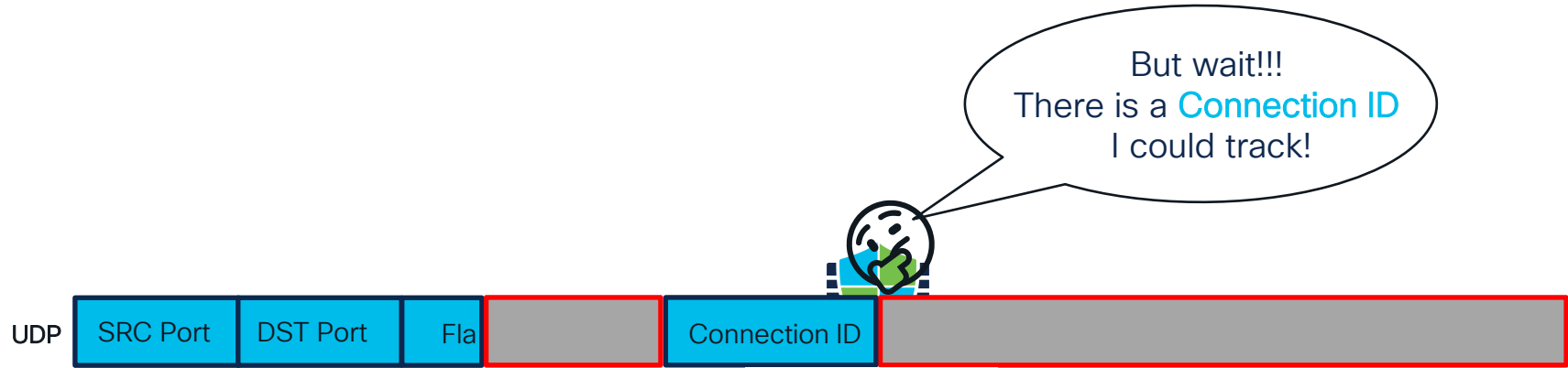
✗ We can't say if this connection is idle or was already closed by QUIC client/server.

✗ No connection reset capabilities (we can't stop the connection as we could with TCP RST).

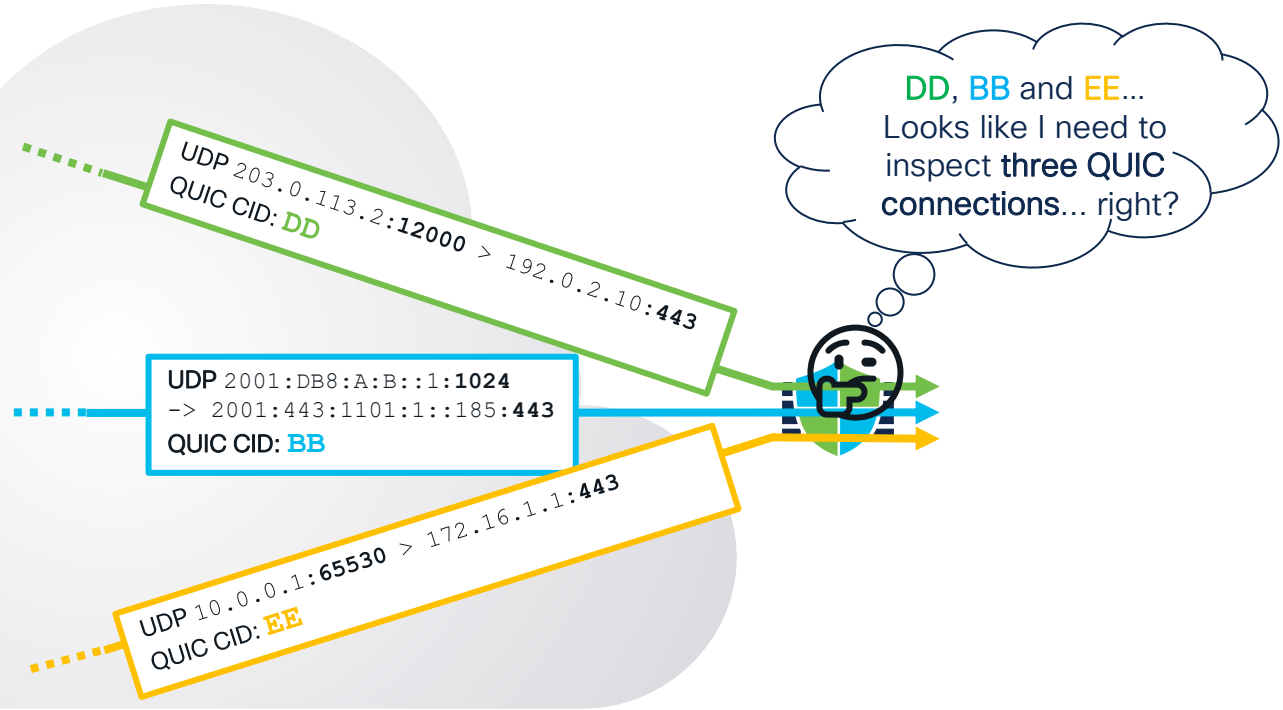
✗ Selective packet drop, won't back-off the application, quite contrary... the app may get more aggressive!

✗ No IPS/Malware protection...

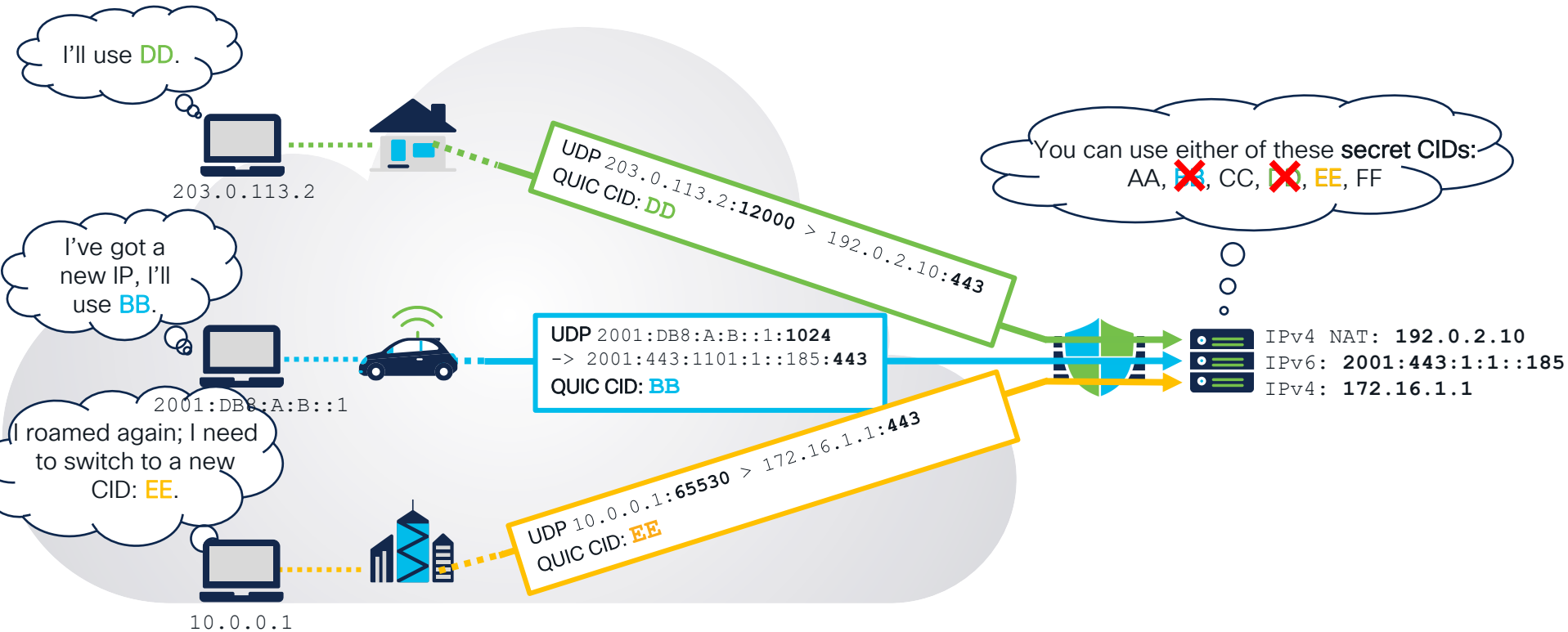
The Firewall is a “Dumb Middle-Box” ... for now.



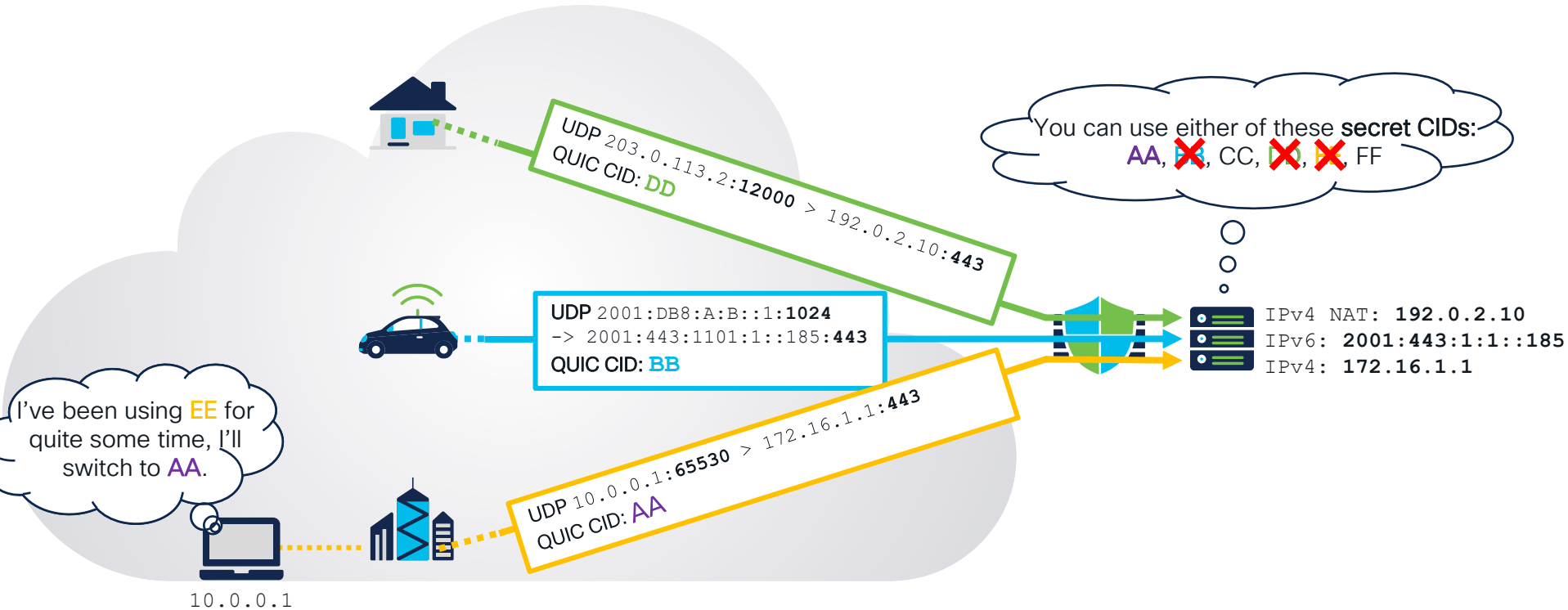
How Many QUIC Connections Do You See?



QUIC Uses Connection Identifiers (CID) for Connection Migration and Tracking

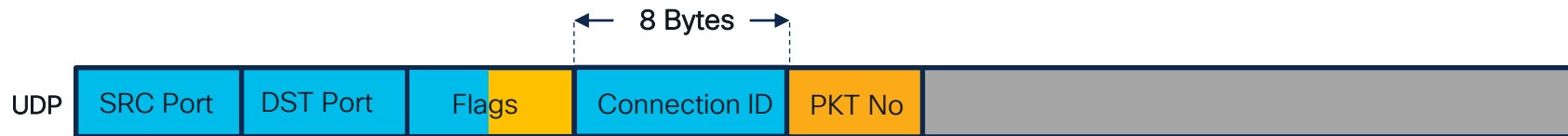


Connection ID Can Change Anytime...

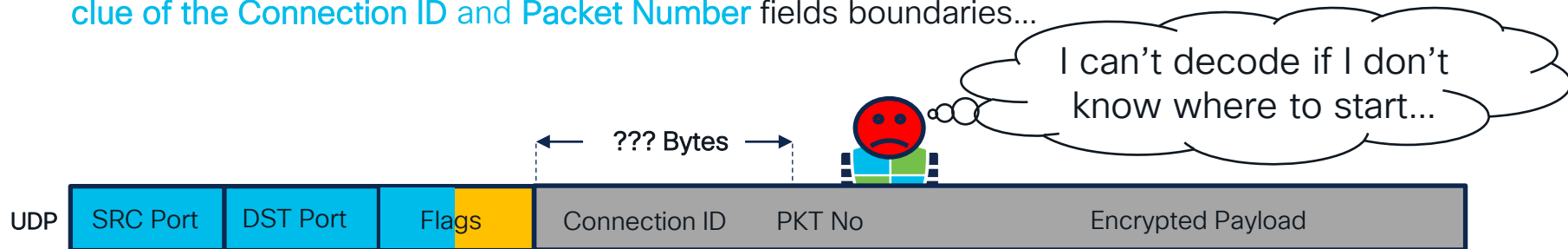


The Connection IDs and Packet Numbers May Be Hidden Too...

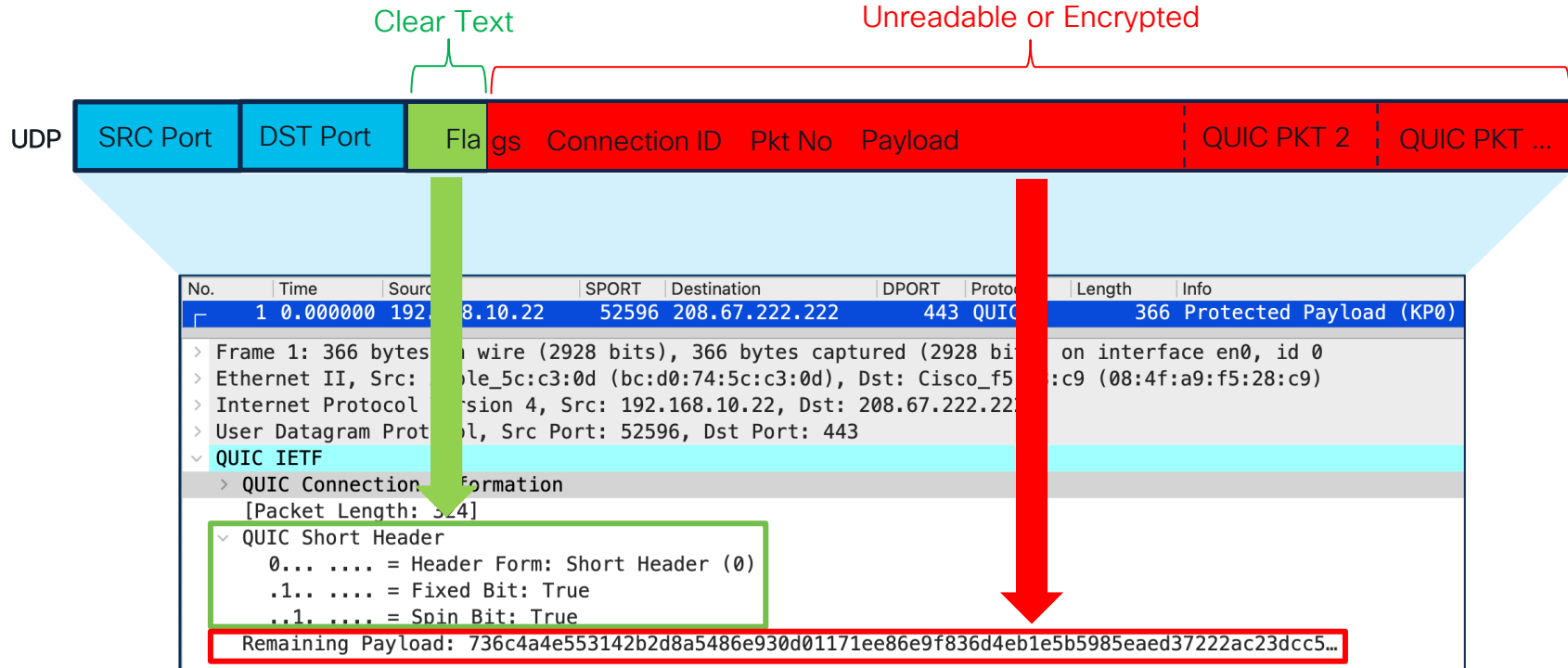
- QUIC uses **Long Header packets during handshake** with Connection ID length field (e.g. 8 bytes)
- Afterwards, the QUIC switches to **Short Header packets** without CID length indication. We can still see the Connection ID and Packet Number **only if the CID length remains unchanged**.



- If the **server changes the length** of the Connection ID (over secure channel), the firewall **has no clue of the Connection ID and Packet Number** fields boundaries...

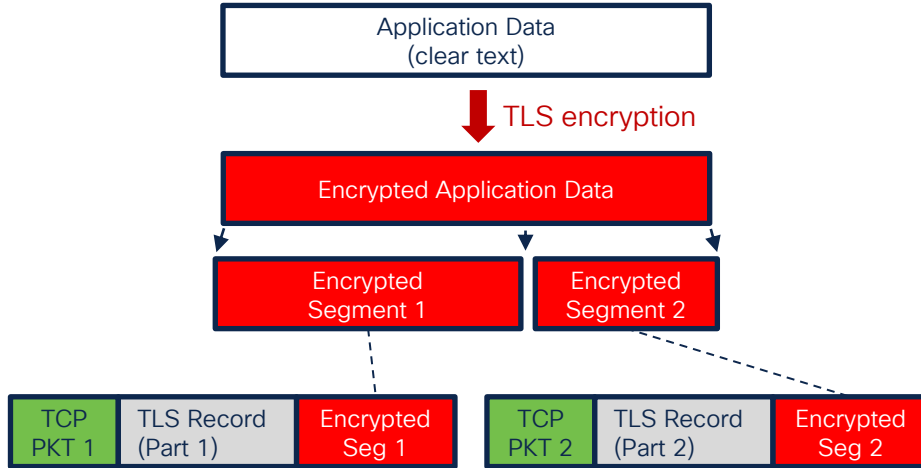


The Firewall Can Read the First 3-bits of First QUIC Packet in the UDP Datagram



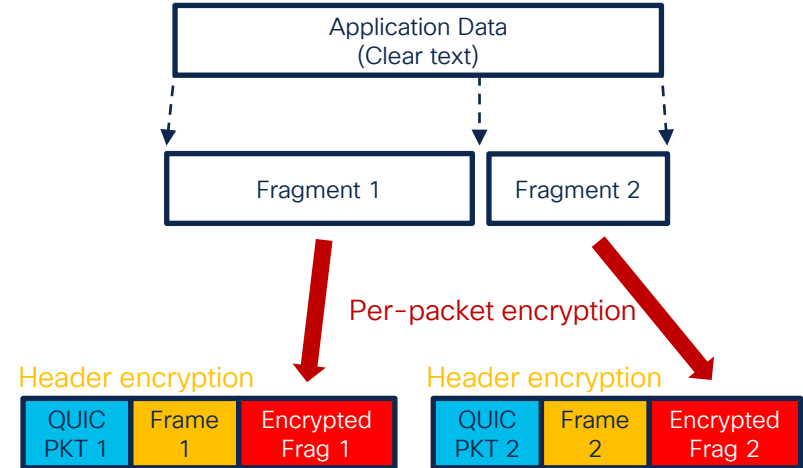
QUIC Requires More Crypto Invocations

- In **HTTP over TCP**, application data is encrypted by the TLS layer entirely, before divided into TCP segments



1x crypto invocation to decrypt

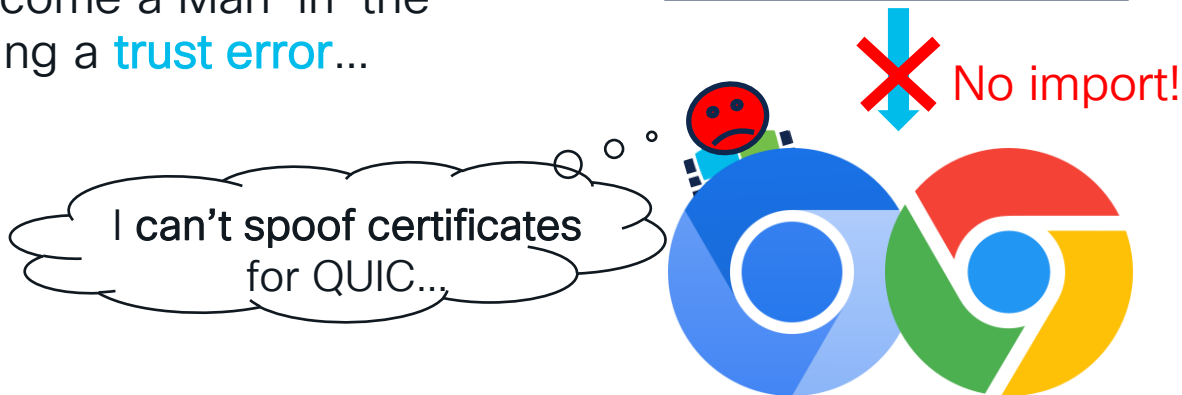
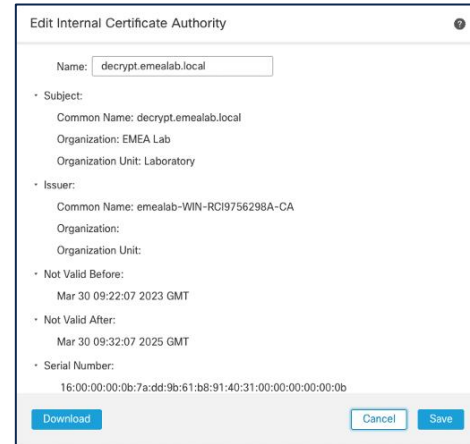
- In **HTTP3**, the data is fragmented first and then encrypted at the QUIC packet level + each packet has header protection



4x crypto invocations to decrypt

Chrome QUIC trusted CA store lock...

- Chrome and Chromium **locked the CA trusted store** for QUIC
- In Chrome we **cannot install a key-resigning certificate authority** for client protection (Internet outbound) use case
- The firewall can't become a Man-in-the-Middle without causing a **trust error**...



DEMO: A QUIC Capture

Why Is It So Hard to Inspect QUIC?

- Always **authenticated** and **encrypted**... **TWICE!!!**
 - It was **designed specifically for quick adoption**, preventing middle boxes from interfering:
 - Obscured or encrypted meta-data: **flow control**, **packet numbers**, **Connection IDs**
 - **Stateful QUIC inspection is** extremely **hard to implement** in data plane (LINA) comparing to TCP
 - **Connection ID oriented** – obsoletes 5-tuple based paradigms
 - Hardware horizontal scaling challenge – 5-tuple load sharing across CPUs no longer possible
 - Seamless IPv4/IPv6 connection migration
 - A traditional **firewall can only detect** QUIC in UDP but that's about it...
-
- The industry consensus is to **block QUIC** (Good luck with that...)
 - You can use **Encrypted Visibility Engine** for **QUIC application visibility**

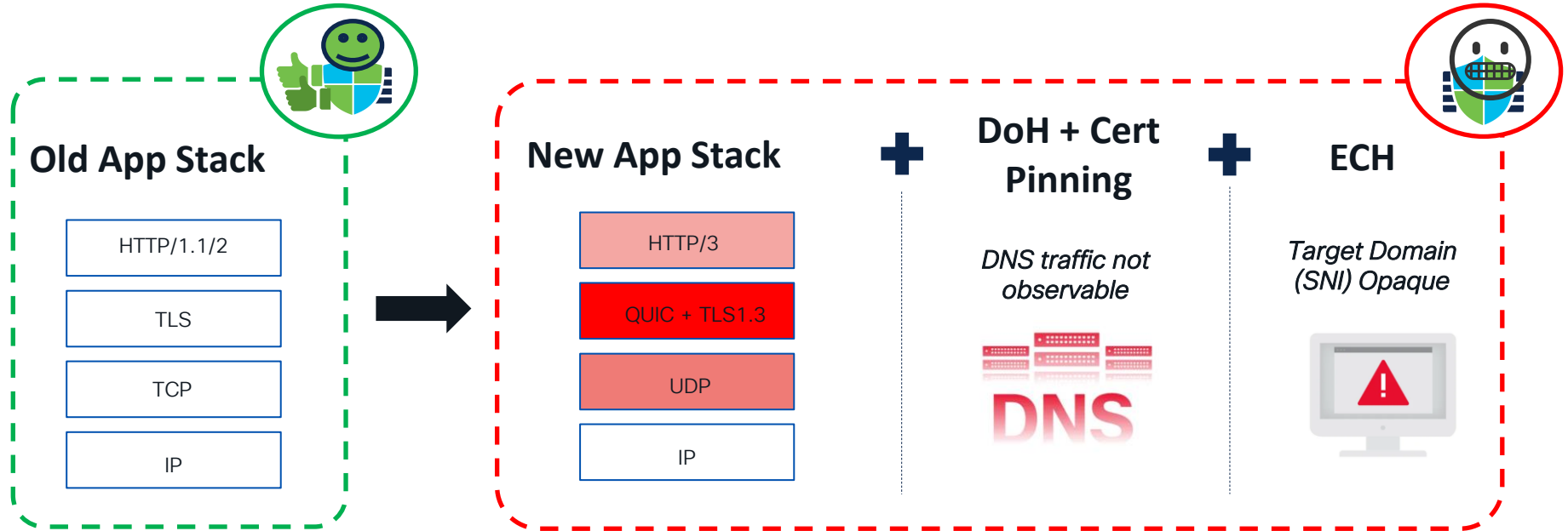
QUIZ 2: QUIC Introduction

Join at
slido.com
#2592 848



Encrypted Visibility Engine

The Nightmare Slowly Becomes a Reality...



Encrypted Visibility Engine uses TLS Fingerprinting

TLS ClientHello

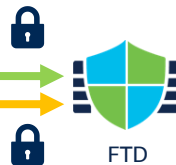
✓ Cipher Suites (18 suites)

Cipher Suite: TLS_AES_128_GCM_SHA256 (0x1301)
Cipher Suite: TLS_CHACHA20_POLY1305_SHA256 (0x1303)
Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0xcca9)
Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0xc030)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
Cipher Suite: TLS_DHE_RSA_WITH_AES_128_CBC_SHA (0x0033)
Cipher Suite: TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x0039)

Confidence: 99.94%
Process: **firefox.exe**
Version: 76.0.1
Category: **browser**
OS: Windows 10 19041.329
Typical FQDN: **cisco.com**

TCP/TLS 192.168.2.110/34624->172.16.45.200/443

TCP/TLS 192.168.2.110/21013->203.0.113.154/443



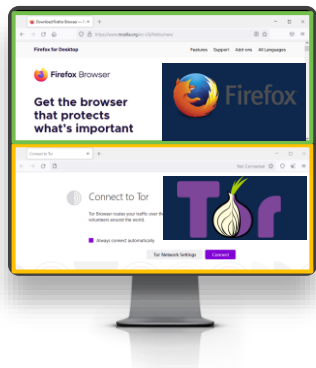
Generate unique fingerprints for client applications based on TLS, TCP and other clear text fields to use for context enrichment

TLS ClientHello

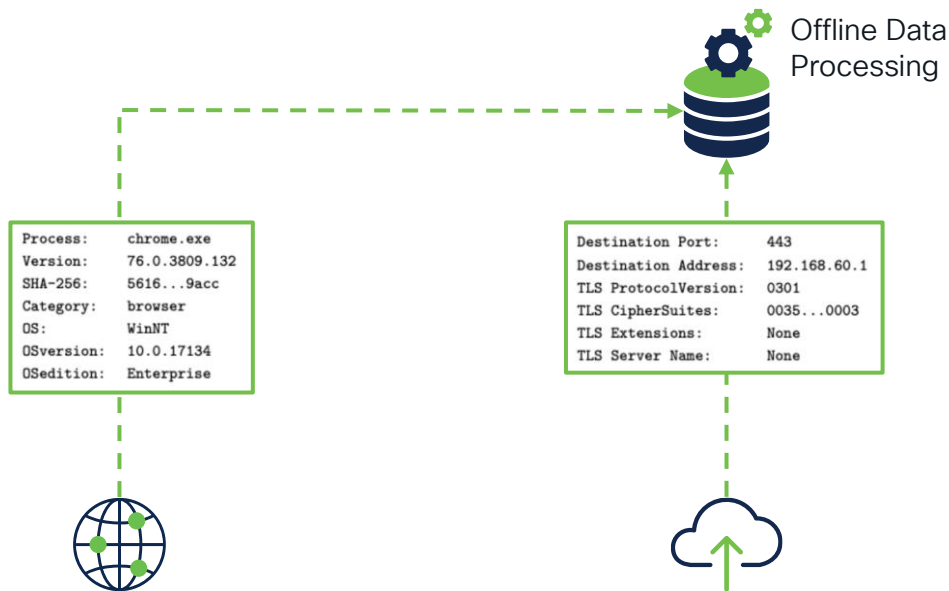
✓ Cipher Suites (19 suites)

Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384 (0xc024)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256 (0xc023)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384 (0xc028)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256 (0xc027)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
Cipher Suite: TLS_RSA_WITH_AES_256_GCM_SHA384 (0x009d)
Cipher Suite: TLS_RSA_WITH_AES_128_GCM_SHA256 (0x009c)

Confidence: 100%
Process: **tor.exe**
Version: 9.0.2
Category: **anonymizer**
OS: Windows 10 19041.329
Typical FQDN: **nsksdlkoup.me**



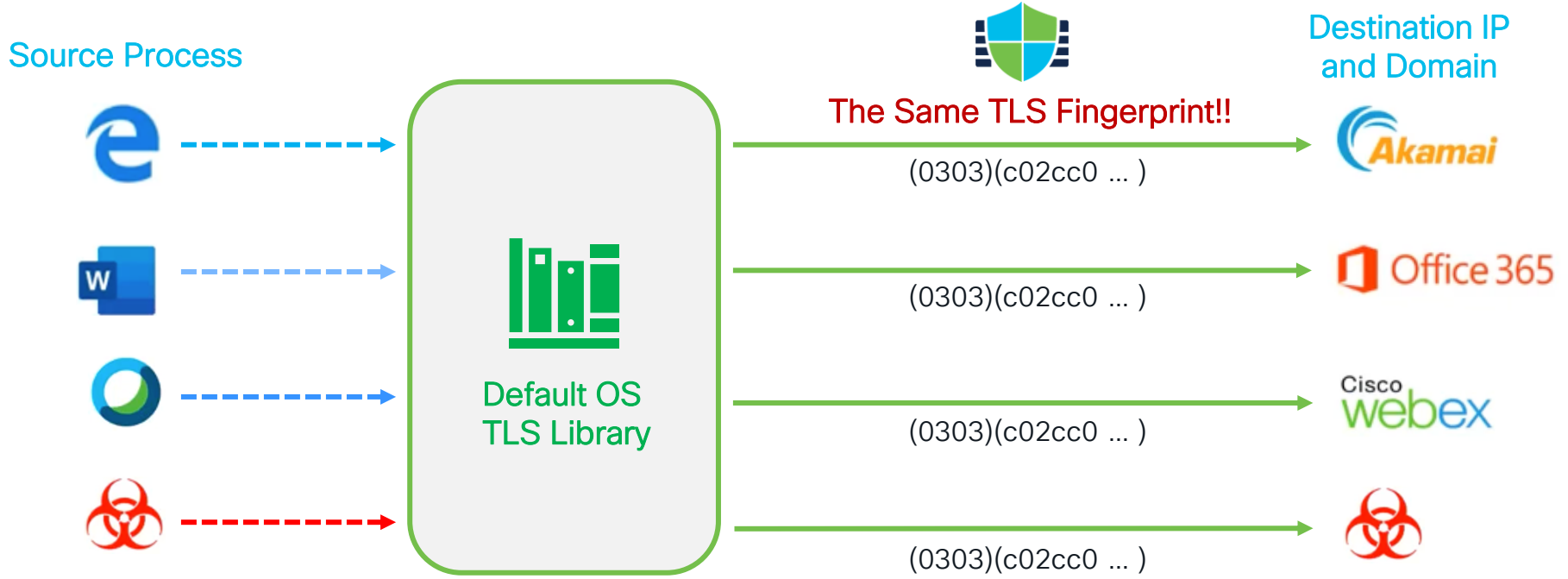
Machine Learning Requires a Comprehensive Data Set



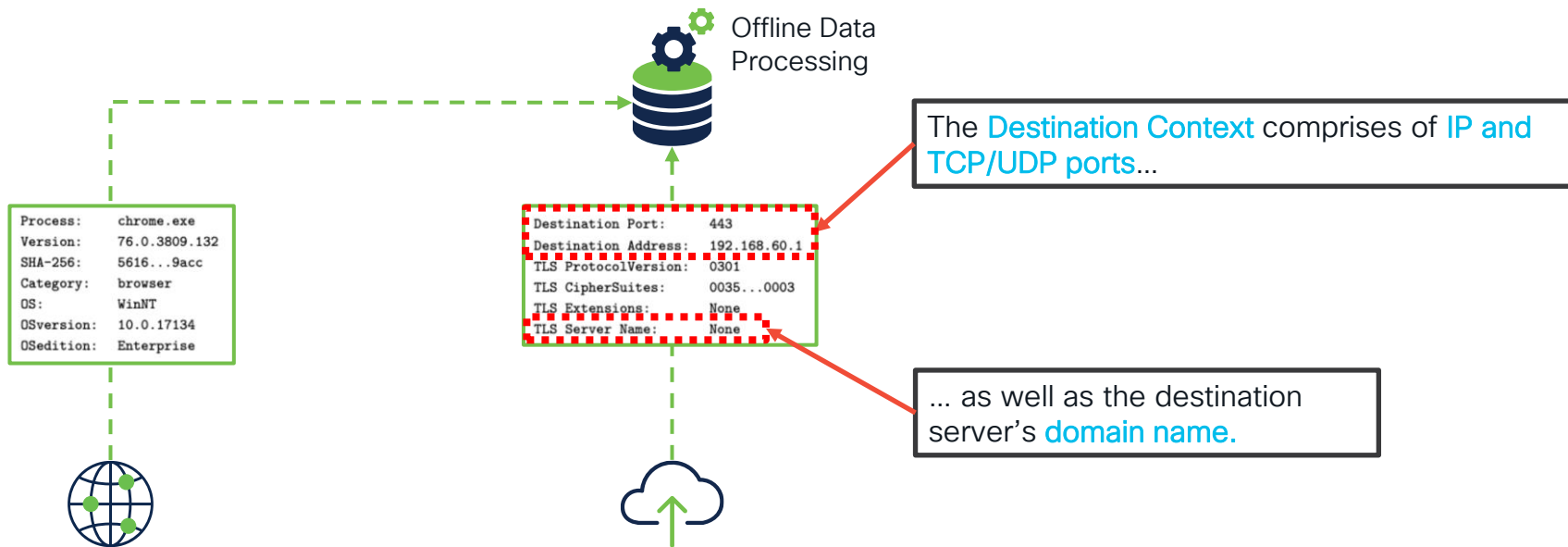
Process data from **80,000**
AnyConnect endpoints
everyday

Over 30 DCs around the globe
collecting **1B** TLS fingerprints **daily**

The Importance of the Destination Context



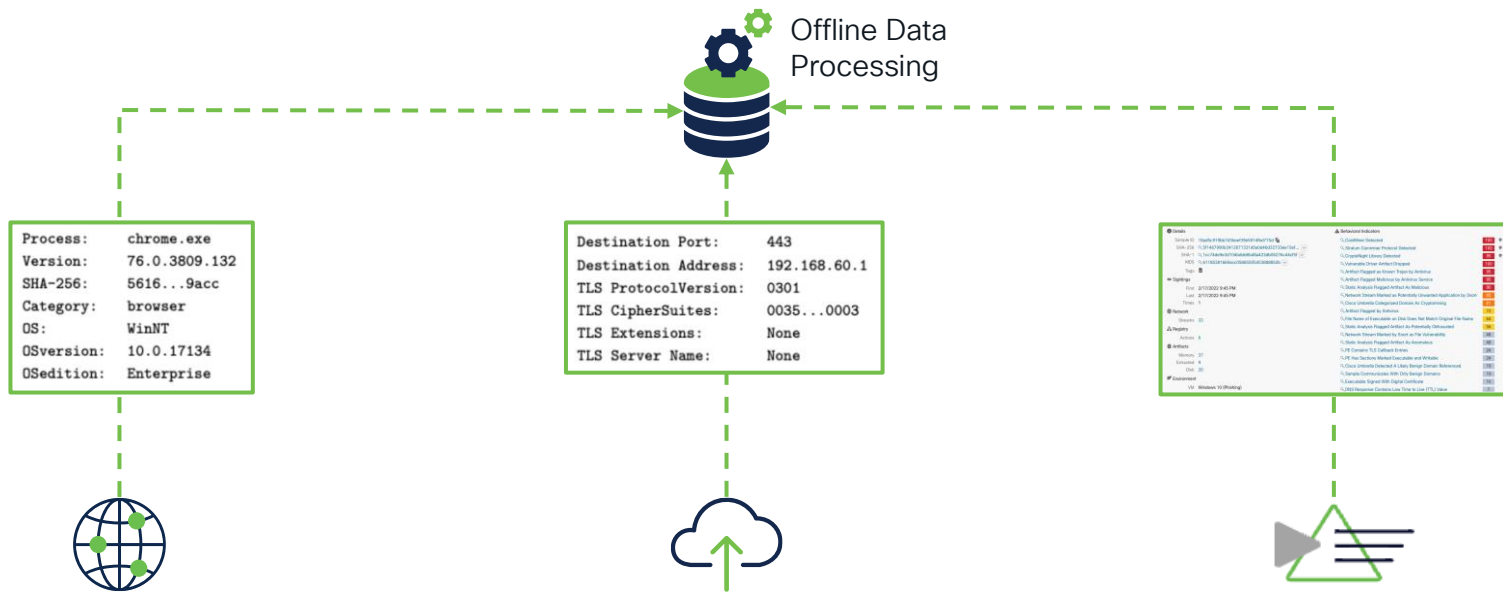
Machine Learning Requires a Comprehensive Data Set



Process data from **80,000**
AnyConnect endpoints
everyday

Over 30 DCs around the globe
collecting **1B** TLS fingerprints **daily**

Machine Learning Requires a Comprehensive Data Set

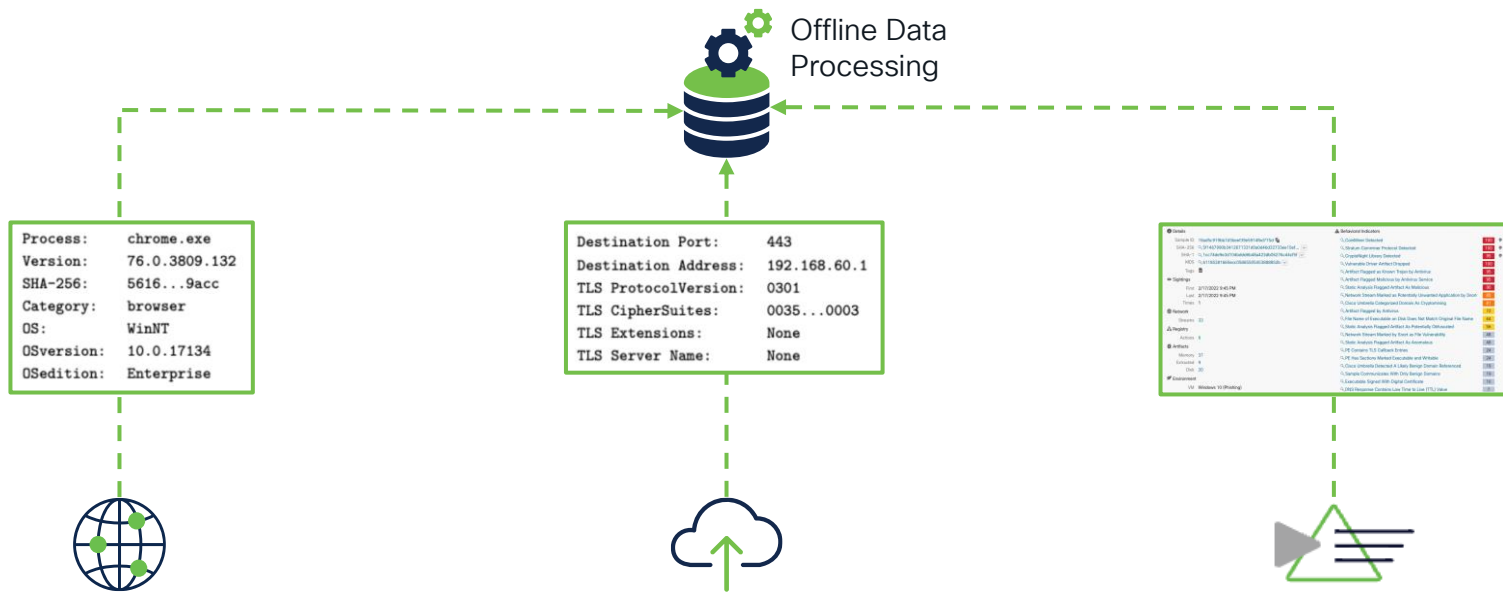


Process data from **80,000**
AnyConnect endpoints
everyday

Over 30 DCs around the globe
collecting 1B TLS fingerprints daily

10,000 samples sandboxed
by Secure Malware Analytics
everyday

Machine Learning Requires a Comprehensive Data Set

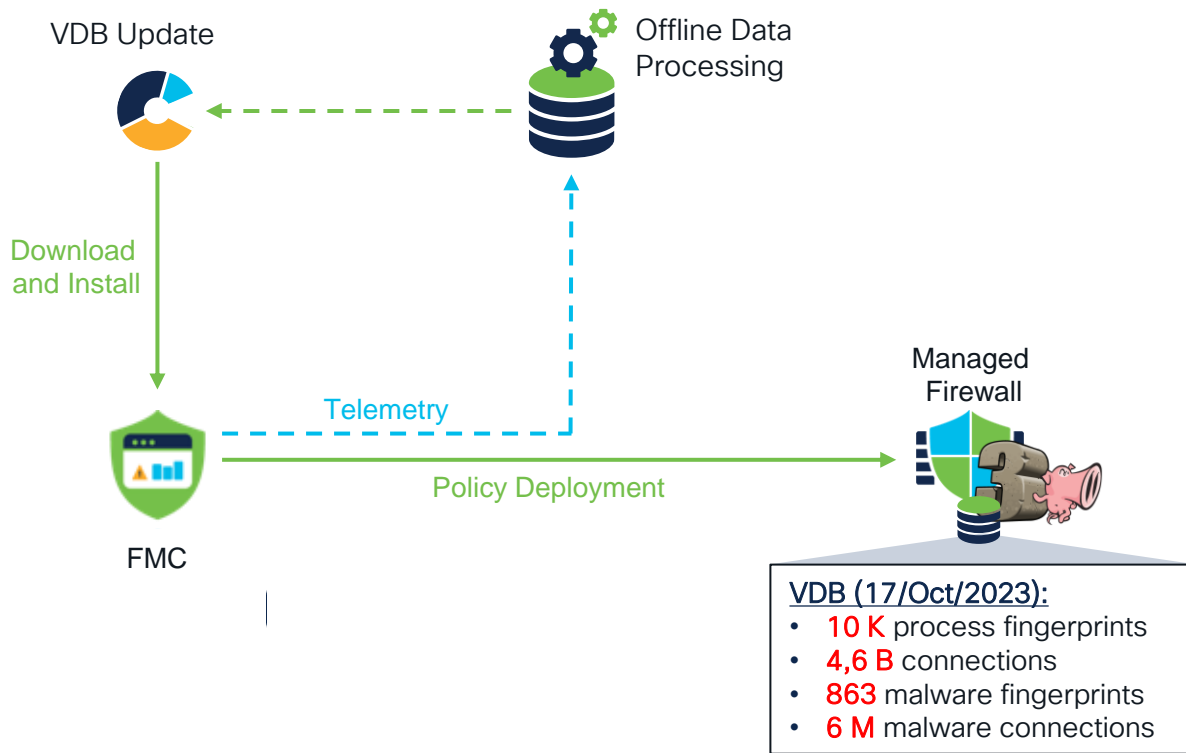


Process data from **80,000**
AnyConnect endpoints
everyday

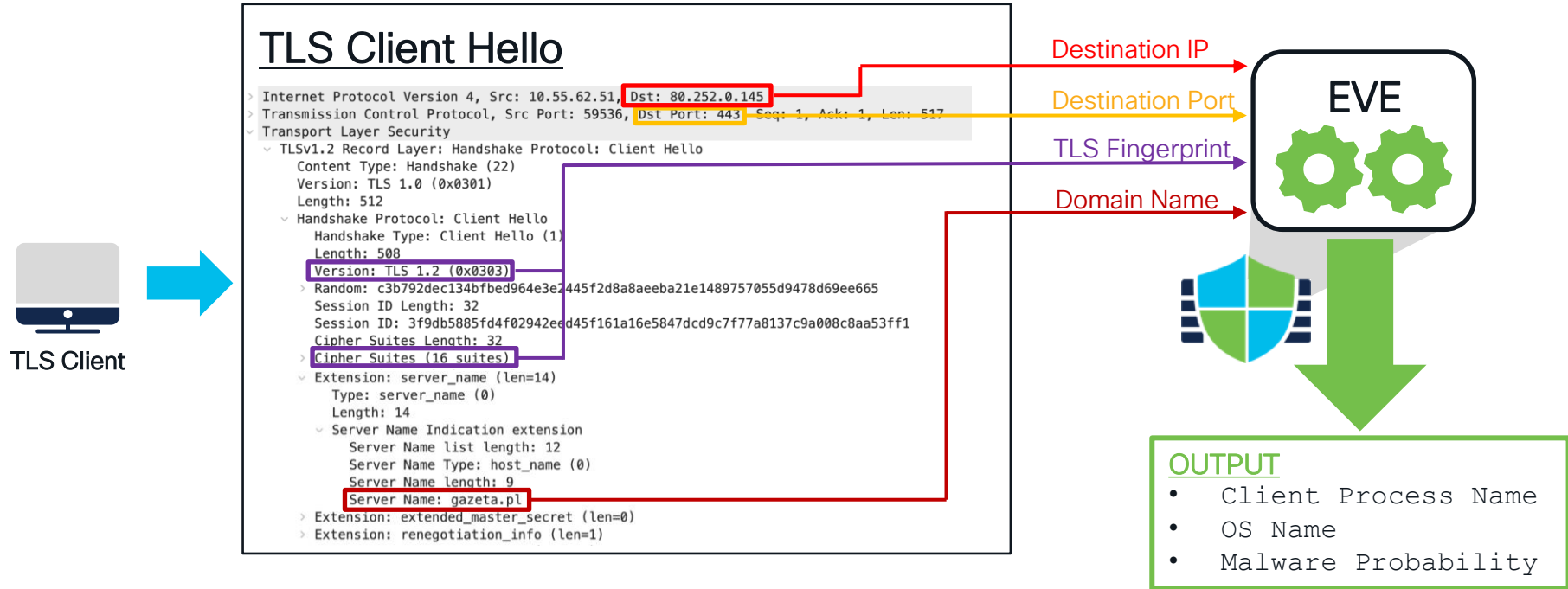
Over 30 DCs around the globe
collecting **1B** TLS fingerprints **daily**

10,000 samples sandboxed
by Secure Malware Analytics
everyday

TLS Fingerprint Database is Included in the VDB



Fingerprinting Analysis at the Firewall



EVE in action

Process Confidence denotes how certain we are with EVE judgement on the source process.

Threat Confidence and **Score** provide calculated likelihood of a flow being sourced by malware.

	Time	Action	Event Type	Destination IP	Destination Port / ICMP Code	Encrypted Visibility Process Confidence Score	Encrypted Visibility Process Name	Encrypted Visibility Threat Confidence	Encrypted Visibility Threat Confidence Score	Client Application	Detection Type
>	2022-05-04 17:14:10	➡ Allow	🔗 Connection	199.58.81.140	443 (https) / tcp	100%	tor	Very Low	0%	TOR	TLS Fingerprint
>	2022-05-04 17:13:53	➡ Allow	🔗 Connection	163.172.21.117	443 (https) / tcp	100%	tor	Very Low	0%	TOR	TLS Fingerprint
>	2022-05-04 17:13:49	➡ Allow	🔗 Connection	45.66.33.45	443 (https) / tcp	100%	tor	Very Low	0%	TOR	TLS Fingerprint
>	2022-05-04 17:13:32	➡ Allow	🔗 Connection	131.188.40.189	443 (https) / tcp	100%	tor	Very Low	0%	TOR	TLS Fingerprint
>	2022-05-04 17:13:27	➡ Allow	🔗 Connection	45.14.233.149	443 (https) / tcp	100%	tor	Very Low	0%	TOR	TLS Fingerprint

Process Name denotes the name of the process on the host that generated this flow.

Client Application detected by EVE and added to the source **Host Profile**.

Detection Type is marked as **TLS Fingerprint**.

Viewing EVE Process Analysis

...and cross-launch to appid.cisco.com to check the **Process Analysis**.

You can view **EVE fingerprints** in a new column Unified Events...

Fingerprint **prevalance** across processes in EVE's dataset.

Destination context – SNI / IP / Port distribution in EVE's dataset.

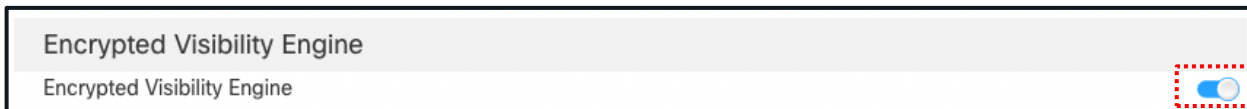
Time	Event Type	Control Rule	Access Control Policy	Device	Encrypted Visibility Fingerprint	Encrypted Visibility Process Confidence Score	Encrypted Visibility Process Name	Encrypted Visibility Threat Confidence	Encrypted Visibility Threat Confidence Score
2023-05-22 10:05:43	% Connection	any Log	ACP	ftd.emeslab.local	tls/1/(0303)(c02cc02bc030...	30%	microsoft networking	Medium	22%
2023-05-22 10:04:52	% Connection	any Log	ACP	ftd.emeslab.local	https		code42 crashplan	Very Low	0%

Process Name ↑↓	Prevalance ↑↓
<input type="checkbox"/> generic dmz process	0.4488
<input type="checkbox"/> cisco amp for endpoints	0.2037
<input type="checkbox"/> microsoft office	0.1388
<input type="checkbox"/> microsoft networking	0.1382
<input type="checkbox"/> cisco webex	0.0218
<input type="checkbox"/> cisco anyconnect	0.0188
<input type="checkbox"/> google services	0.0053

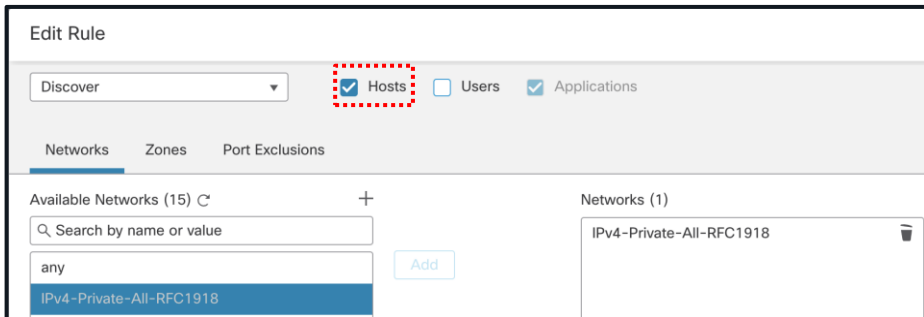
Server Name ↑↓	IP Address ↑↓	Port ↑↓
1.0000	0.8121	0.4545
0.0000	0.0000	0.2086
0.0000	0.0029	0.1420
0.0000	0.1846	0.1417
0.0000	0.0000	0.0224
0.0000	0.0000	0.0000
0.0000	0.0000	0.0054

Enabling EVE

- Enable EVE in ACP **Advanced** tab:



- Enable **Hosts** in the **Discovery Policy** to view them in the network map:



Configuring an EVE Based Application Detectors

1. **Create Custom Detector** with EVE Detection Type.

Name	Detection Type	Details	Port(s)	Type	State
EVE-AppD-ToR EVE Detected ToR	EVE	EVE-AppD-ToR		Basic	

Edit Pattern

Detection Type *
Encrypted Visibility Engine

Process Name
tor

Min Process Confidence
95

Cancel OK * Required

Action: Block Time Range: None

Zones Networks VLAN Tags Users Applications Ports URLs Dynamic Attributes

Application Filters Clear All Filters

Search by name

Available Applications (3)

Q EVE-App

All apps matching the filter

- EVE-AppD-Dropbox
- EVE-AppD-Skype
- EVE-AppD-ToR

Add to Rule

Selected Applications and Filters (1)

Applications

EVE-AppD-ToR

2. Specify **Process Name** and minimum **Confidence** level.

3. Apply your custom detector in the **Access Control Policy** rule.

DEMO: Blocking Malicious Flows with Encrypted Visibility Engine

JA3 and JA3S

Learn more:

<https://engineering.salesforce.com/tls-fingerprinting-with-ja3-and-ja3s-247362855967/>



TLS Client



TLS Client Hello

```
> Internet Protocol Version 4, Src: 172.16.12.20, Dst: 89.238.73.97
> Transmission Control Protocol, Src Port: 65526, Dst Port: 443, Seq: 1, Ack: 1, Len: 517
> Transport Layer Security
  > TLSv1.3 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: 1e27e095f72393a1b11ea00784ea49a32122da108d62fc266af2adc7c50aad7
    Session ID Length: 32
    Session ID: 48af6a694a01f8860ac00dcd928714efe7fa2c29b5e804b574c6a2bf406eeec9
    Cipher Suites Length: 32
    Cipher Suites (16 suites)
```

JA3



JA3 = e7d705a3286e19ea42f587b344ee6865

Based on:

- Version
- Accepted Ciphers
- List of Extensions
- Elliptic Curves and Formats

JA3S = a95ca7eab4d47d051a5cd4fb7b6005dc

Based on:

- Version
- Accepted Ciphers
- List of Extensions

TLS Server Hello

```
> Internet Protocol Version 4, Src: 89.238.73.97, Dst: 172.16.12.20
> Transmission Control Protocol, Src Port: 443, Dst Port: 65526, Seq: 1, Ack: 518, Len: 127
> Transport Layer Security
  > TLSv1.3 Record Layer: Handshake Protocol: Server Hello
    Content Type: Handshake (22)
    Version: TLS 1.2 (0x0303)
    Length: 122
  > Handshake Protocol: Server Hello
    Handshake Type: Server Hello (2)
    Length: 118
    Version: TLS 1.2 (0x0303)
    Random: 0541fca0983815638f85a8722e201f31f742e11b9bee912a2059f016f8615981
    Session ID Length: 32
    Session ID: 48af6a694a01f8860ac00dcd928714efe7fa2c29b5e804b574c6a2bf406eeec9
    Cipher Suite: TLS_AES_256_GCM_SHA384 (0x1302)
    Compression Method: null (0)
```

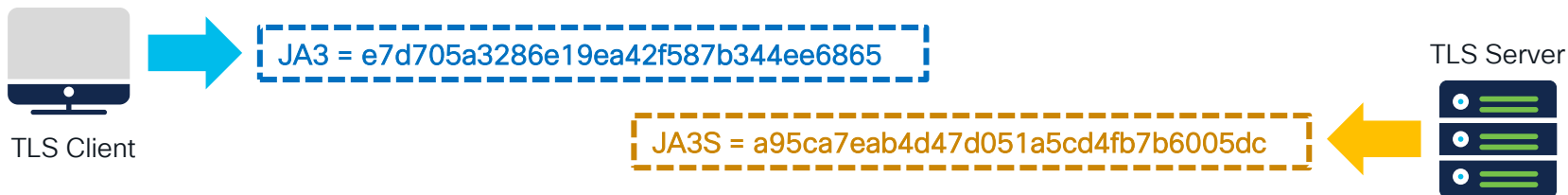
JA3S



TLS Server



JA3 and JA3S



- Neither JA3 nor JA3S produces accurate discovery on its own:
 - **JA3 does not provide destination context** – individual processes and destinations may produce the same fingerprint
 - **JA3S depends on the client hello** – there is no single fingerprint value identifying an individual TLS server
- However, when JA3 and JA3S used together:
 - **JA3/JA3S pair becomes more reliable** – usually, the server responds to the same client in exactly same way
 - **The JA3/JA3S is IP/domain agnostic** – if malware CnC relocates, the TLS client/server fingerprints remain unchanged

CISCO *Live!*



Client Hello Normalization with EVE

Learn more:

<https://hnull.org/2022/12/01/sorting-out-randomized-tls-fingerprints/>

The new **normalized TLS fingerprint** format.



TLS Client Hello 1

```
Internet Protocol Version 4, Src: 10.55.62.51, Dst: 88.252.8.145
```

TLS Client Hello 2

```
Internet Protocol Version 4, Src: 10.55.62.51, Dst: 88.252.8.145
```

```
Transmission Control Protocol, Src Port: 59536, Dst Port: 443, Seq: 1, Ack: 1, Len: 517
```

TLS Client Hello ...

```
Internet Protocol Version 4, Src: 10.55.62.51, Dst: 88.252.8.145
```

```
Transmission Control Protocol, Src Port: 59536, Dst Port: 443, Seq: 1, Ack: 1, Len: 517
```

TLS Client Hello N

```
Internet Protocol Version 4, Src: 10.55.62.51, Dst: 88.252.8.145
```

```
Transmission Control Protocol, Src Port: 59536, Dst Port: 443, Seq: 1, Ack: 1, Len: 517
```

```
Transport Layer Security
```

```
Content Type: Handshake (22)
```

```
Version: TLS 1.0 (0x0301)
```

```
Length: 512
```

```
Handshake Protocol: Client Hello
```

```
Handshake Type: Client Hello (1)
```

```
Length: 500
```

```
Version: TLS 1.2 (0x0303)
```

```
Random: c3b792dec134b7fed964e3e2445f208baeeba21e1409757855d9478d89ee665
```

```
Session ID Length: 32
```

```
Session ID: 3f9b058b5f4f82942eed45f161a565847cd9c7f77db137c9a0808caa53f1f
```

```
Cipher Suites Length: 52
```

```
Cipher Suites (16 suites)
```

tls/1

(0303)

(130213031301c02cc030009fcc...)

[

(0000)

(000a000c000a001d0017001e00190018)

(000b000403000102)

(000d0030002e040305030603080...)

(0016)

(0017)

(0023)

(0033)

]

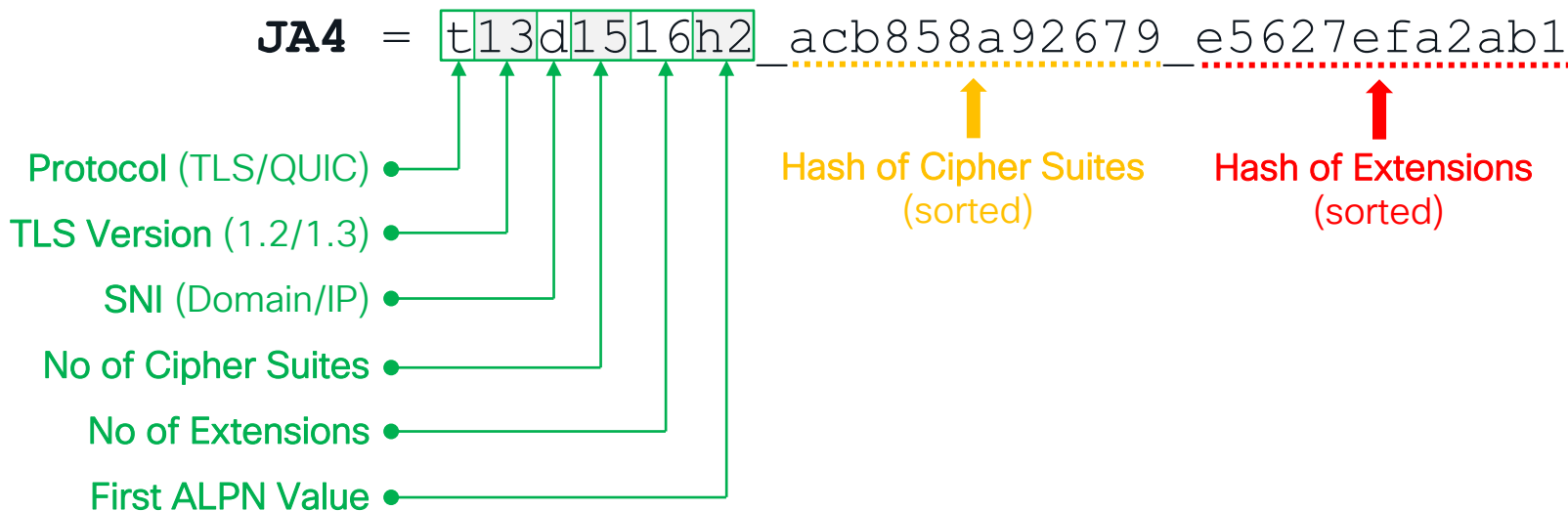


EVE provides the same fingerprint for all randomized Client Hellos

The **extensions are sorted** lexicographically (dictionary order).

New Kid on the Block - JA4/JA4+

- Next generation replacement of JA3/JA3S with added **QUIC support**
- **Human and machine-readable** format (Chrome JA4 fingerprint example)



New Kid on the Block - JA4/JA4+

- Next generation replacement of JA3/JA3S with added **QUIC support**
- **Human and machine-readable** format (Chrome JA4 fingerprint example)

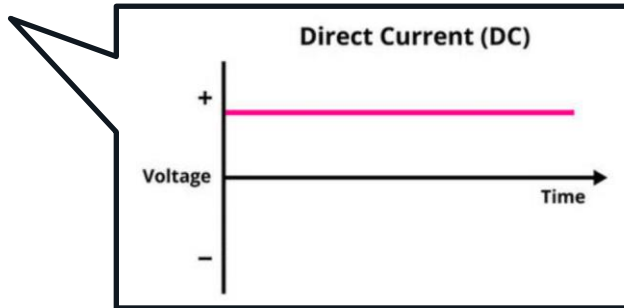
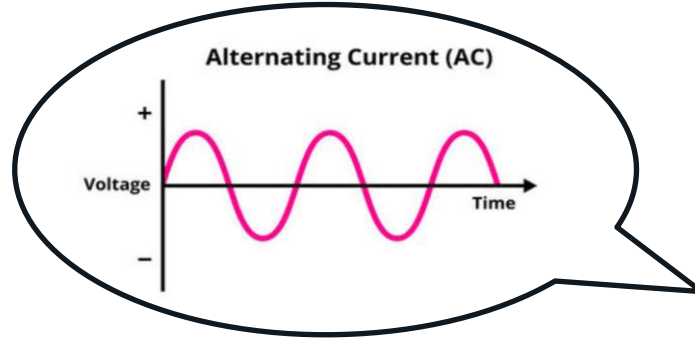
JA4 = $\overbrace{\text{t13d1516h2}}^{\text{JA4_a}} _ \overbrace{\text{acb858a92679}}^{\text{JA4_b}} _ \overbrace{\text{e5627efa2ab1}}^{\text{JA4_c}}$

- Modular fingerprint nature, e.g. JA4_**a**_b_c allows selective analysis on **a**_b, **a**_c, **a** only, etc...
- Goes way beyond **TLS Client (JA4)** and **Server Hello (JA4S)**:
 - **JA4H** – HTTP Client (HTTP request fingerprint)
 - **JA4L** – Light Distance/Location (client/server latency based and TTL)
 - **JA4X** – X509 TLS Certificate (patterns describing how certificates are generated)
 - **JA4SSH** – SSH session state (inactive, reverse shell, file transfer)

EVE vs. JA3/4





Thomas Edison
(1847-1931)

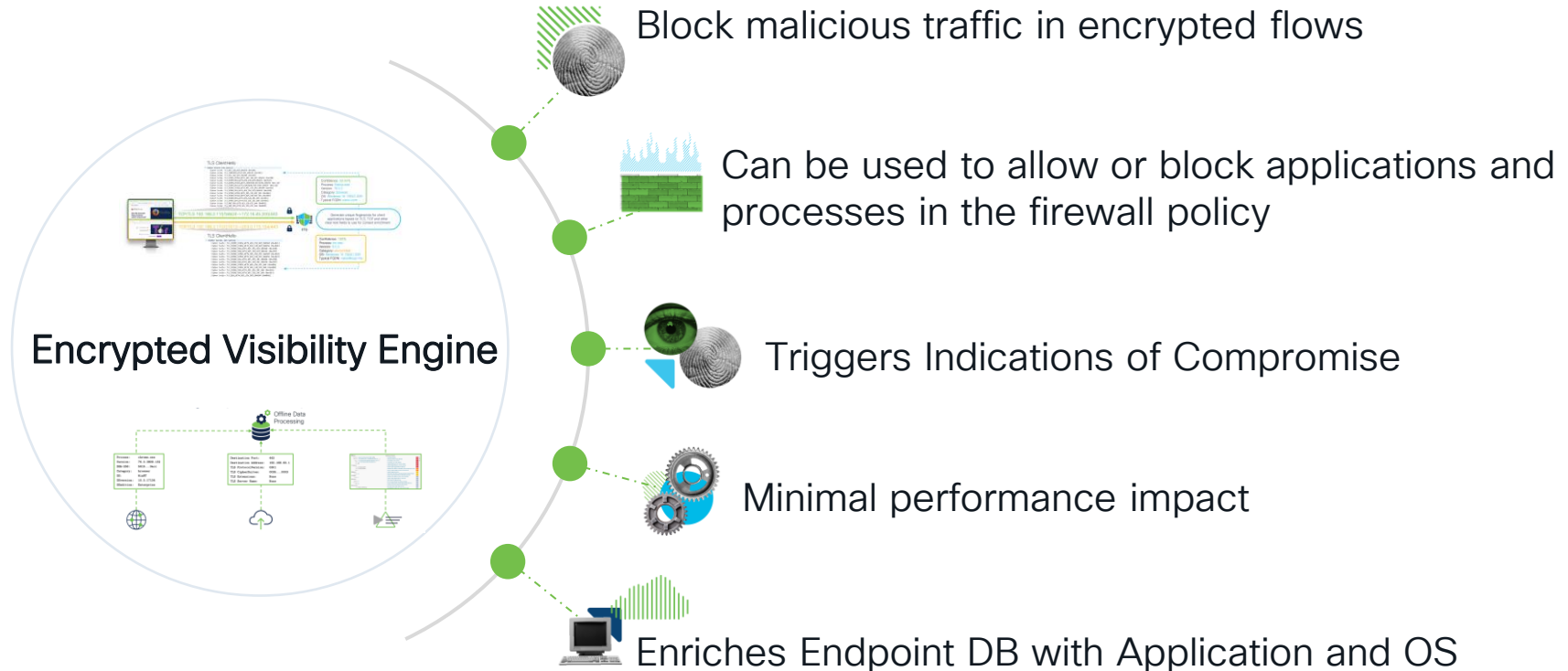


Nicola Tesla
(1856-1943)

EVE vs. JA3 and JA4

Feature		EVE	JA3/JA3S	JA4/JA4S
Fingerprint string computed from packet fields		✓	✓	✓
Data-driven, continuously trained		✓	✗	✗
Destination context		✓	✓	✓
TLS support		✓	✓	✓
QUIC support		✓	✗	✓
Fingerprint Normalization		✓	✗	✓
Fingerprinted TLS Messages		Client Hello	Client/Server Hello	Client/Server Hello

Offload Your Firewall with ML-Based Technology



QUIZ 3: Encrypted Visibility Engine

Join at
slido.com
#2592 848



Conclusions

Cisco Secure Firewall Provides **Unmatched Performance** and **Smart Decryption Services**

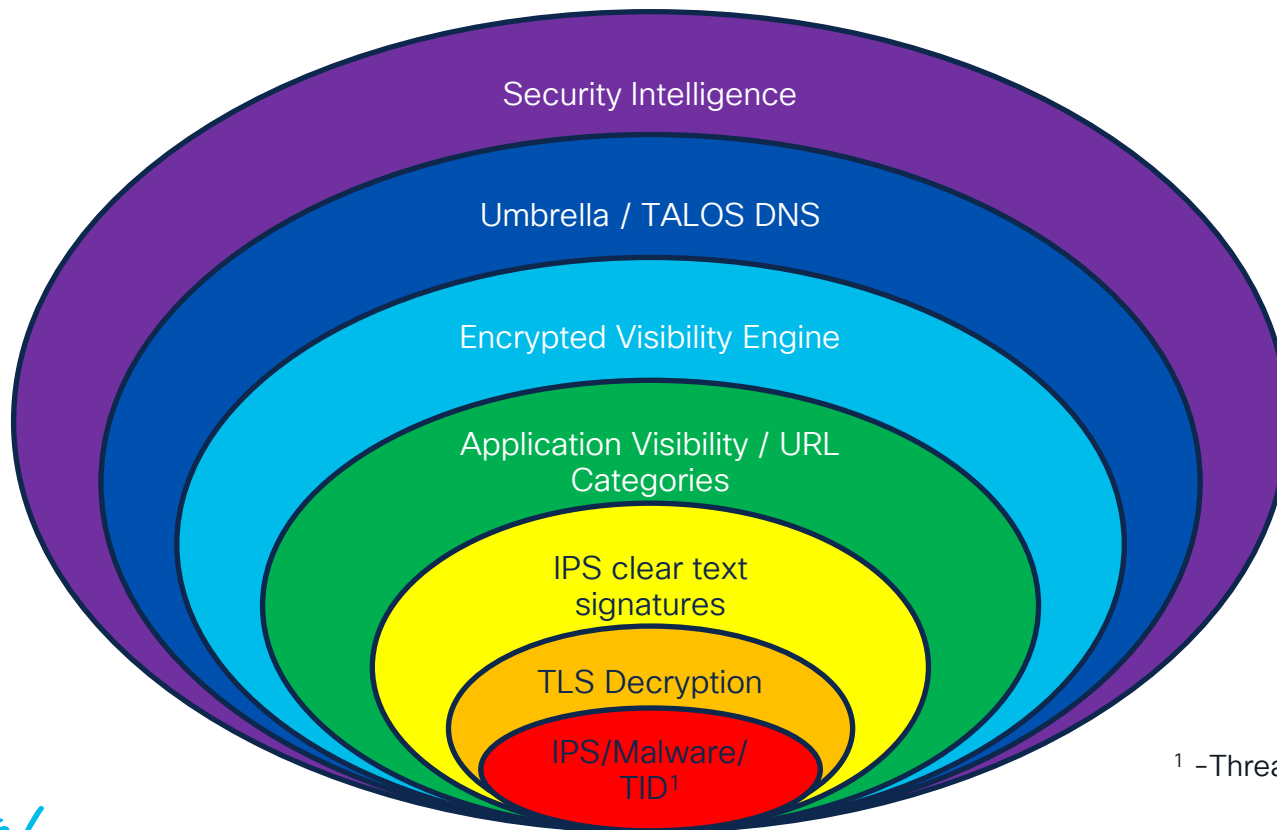
- **Learn protocols!** Better protocol understanding = easier configuration & troubleshooting.
- **Over 80% of the attacks use encrypted channels** – what is the security posture of your firewall, malware and IPS protection?
- Have a look at your decryption policy and think **how to optimize**.
- Check if the **Encrypted Visibility Engine** is enabled in your policy.



“No one can whistle a symphony. It takes a whole orchestra to play it.”

H.E. Luccock

No Single Line of Defense is Enough – Use Them All Together!



¹ -Threat Intelligence Director



The bridge to possible

Thank you

CISCO *Live!*

The background is a vibrant, abstract graphic. On the left, there are overlapping, wavy shapes in shades of red, orange, and yellow, resembling a stylized cloud or a series of overlapping circles. On the right, a bright white light source emits a series of colorful rays in shades of blue, green, and yellow, creating a sunburst effect. The overall color palette is a rainbow spectrum.

cisco *Live!*

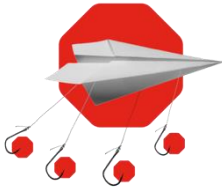
Let's go

BONUS SECTION 😊

A TLS 1.3 Handshake Walkthrough

The Primary Goals of the TLS Handshake

- Negotiate **encryption scheme** and parameters
- **Authenticate the server** (and optionally the client)
- Calculate **shared keying material**

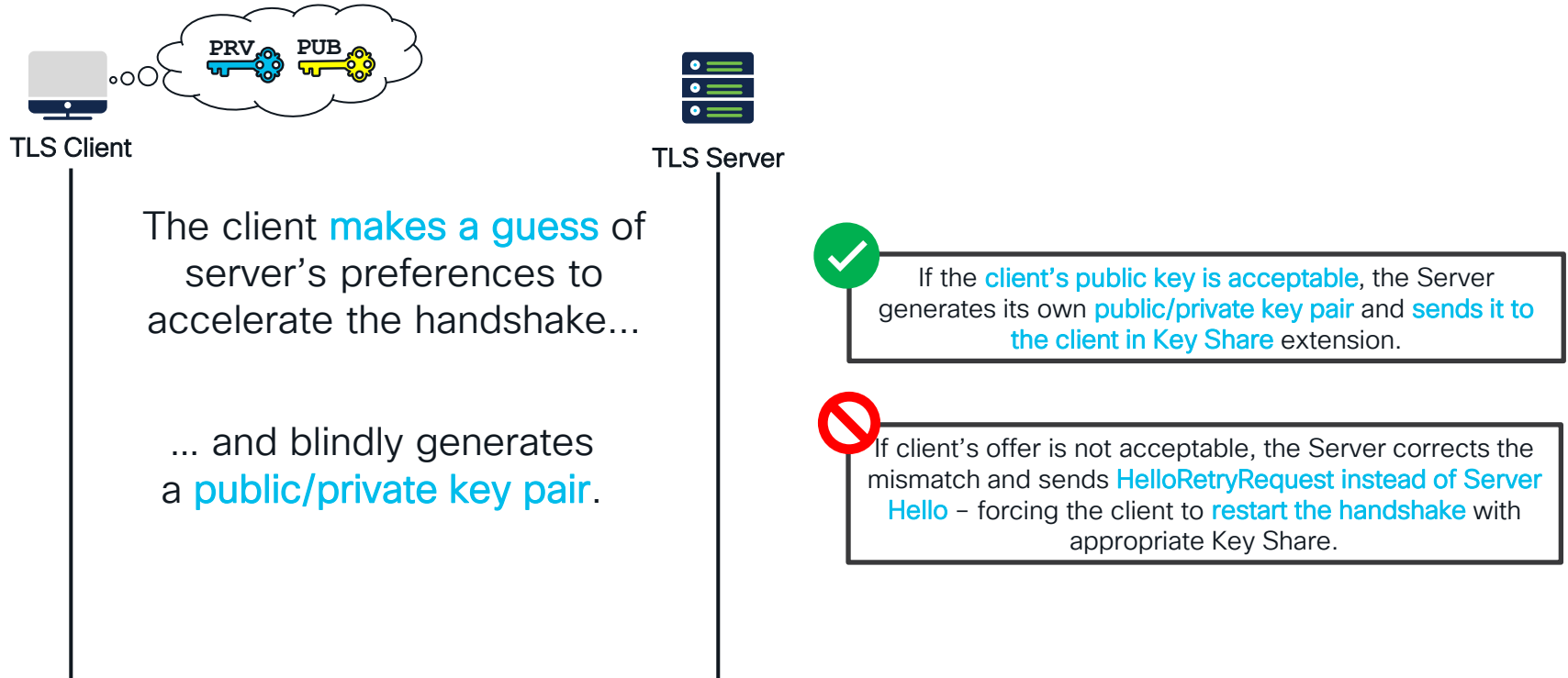


Assume handshake runs
over an unsecure channel

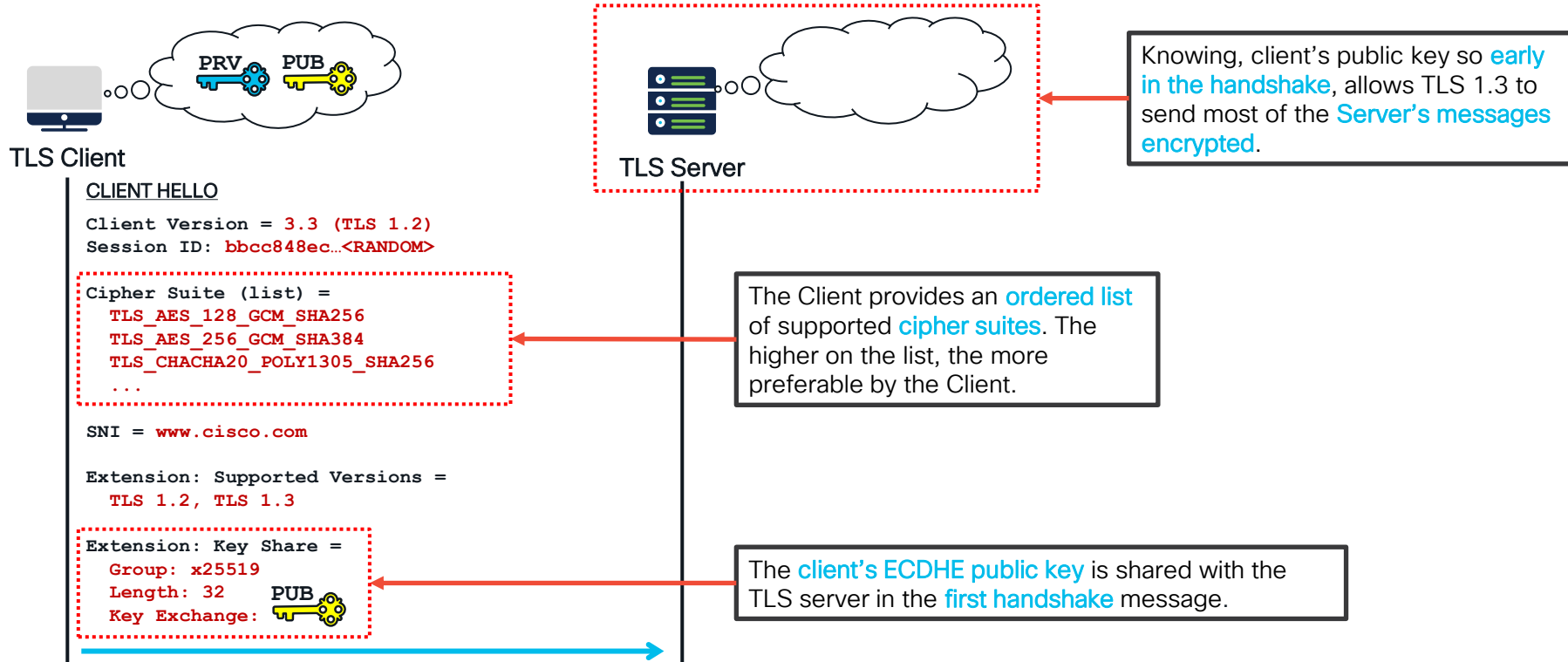


Prevent Man-in-the-Middle
and eavesdropping

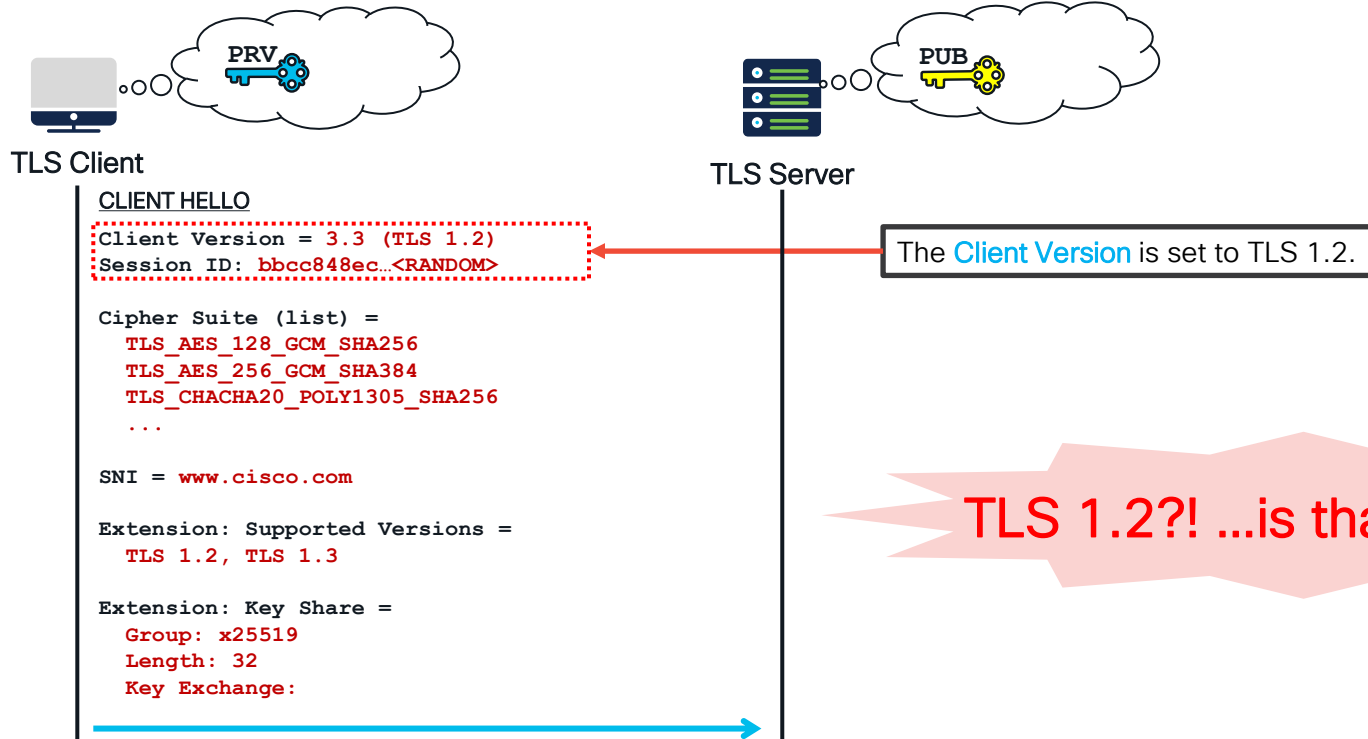
Understanding a TLS 1.3 Session Flow – Client Hello



Understanding a TLS 1.3 Session Flow – Client Hello

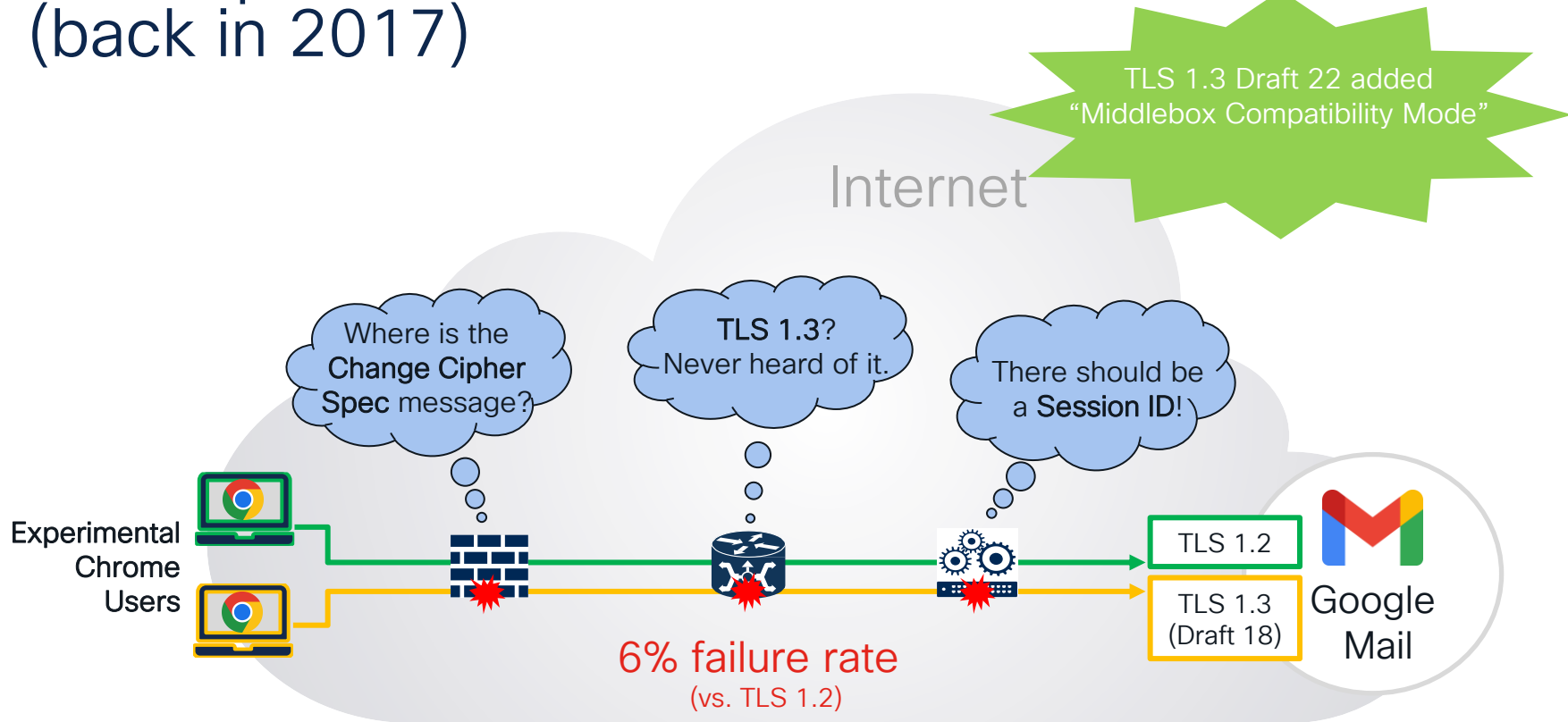


Understanding a TLS Session Flow – TLS Version Negotiation



TLS 1.2?! ...is that correct?

The Experimental Launch of TLS 1.3 Draft (back in 2017)



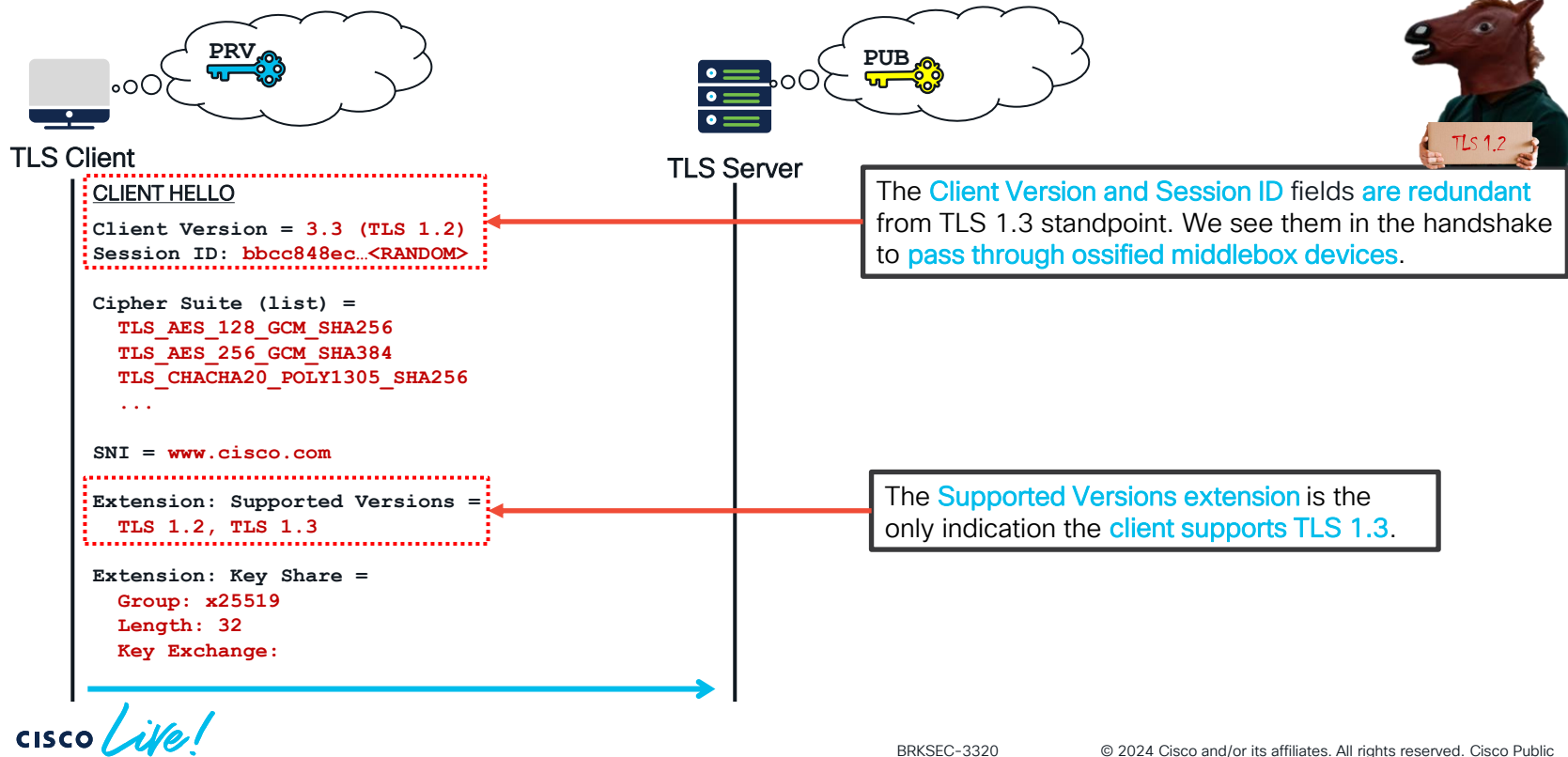
The solution being... a **convincing** disguise

Middlebox Compatibility Mode:

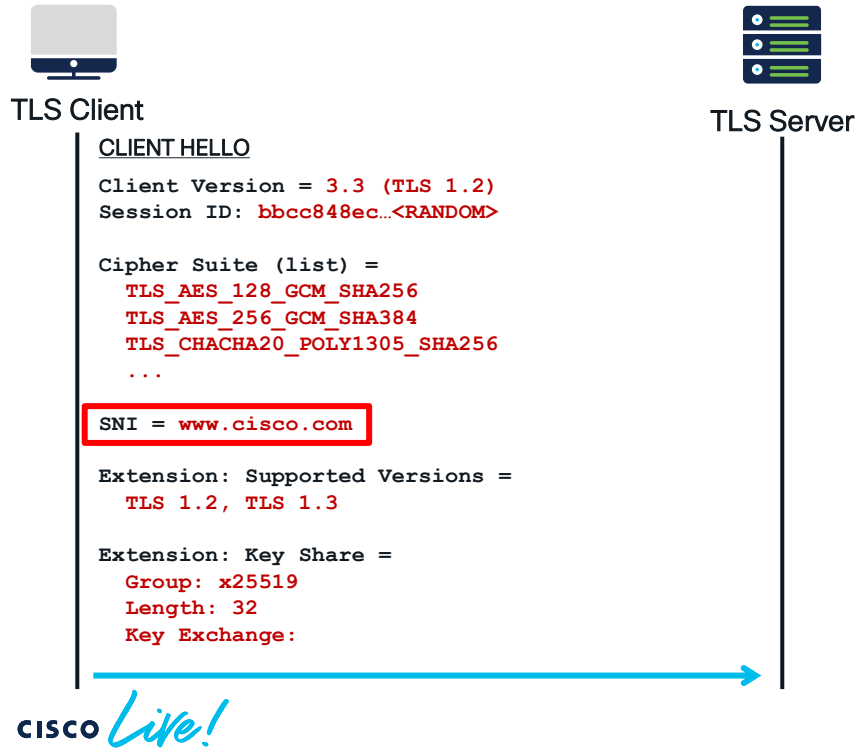
- Make the TLS 1.3 handshake look like **TLS 1.2 session resumption**
- Include a **non-empty Session ID**
- Send a dummy **ChangeCipherSpec** record



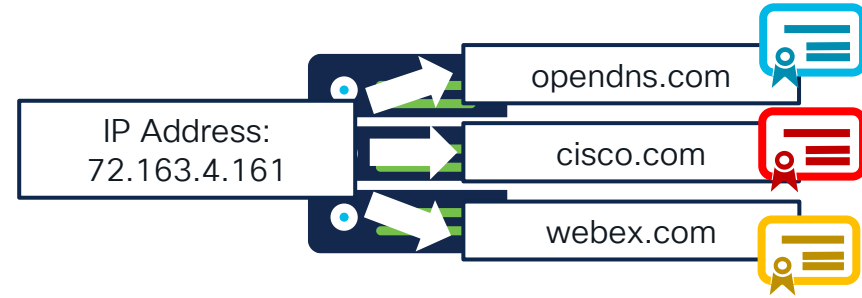
Understanding a TLS Session Flow – Middlebox Compatibility



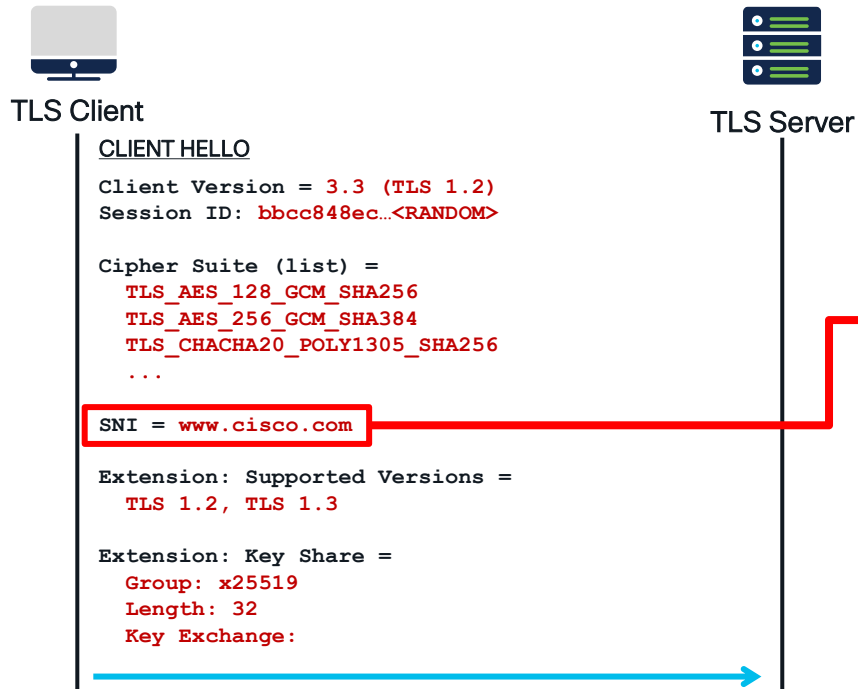
Understanding a TLS Session Flow – Server Name Indication (SNI)



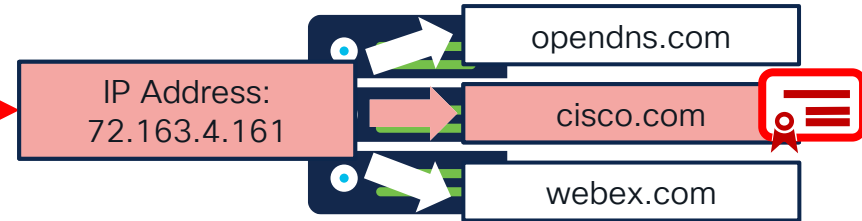
Why do we need the clear text **Server Name Indication (SNI)** extension?



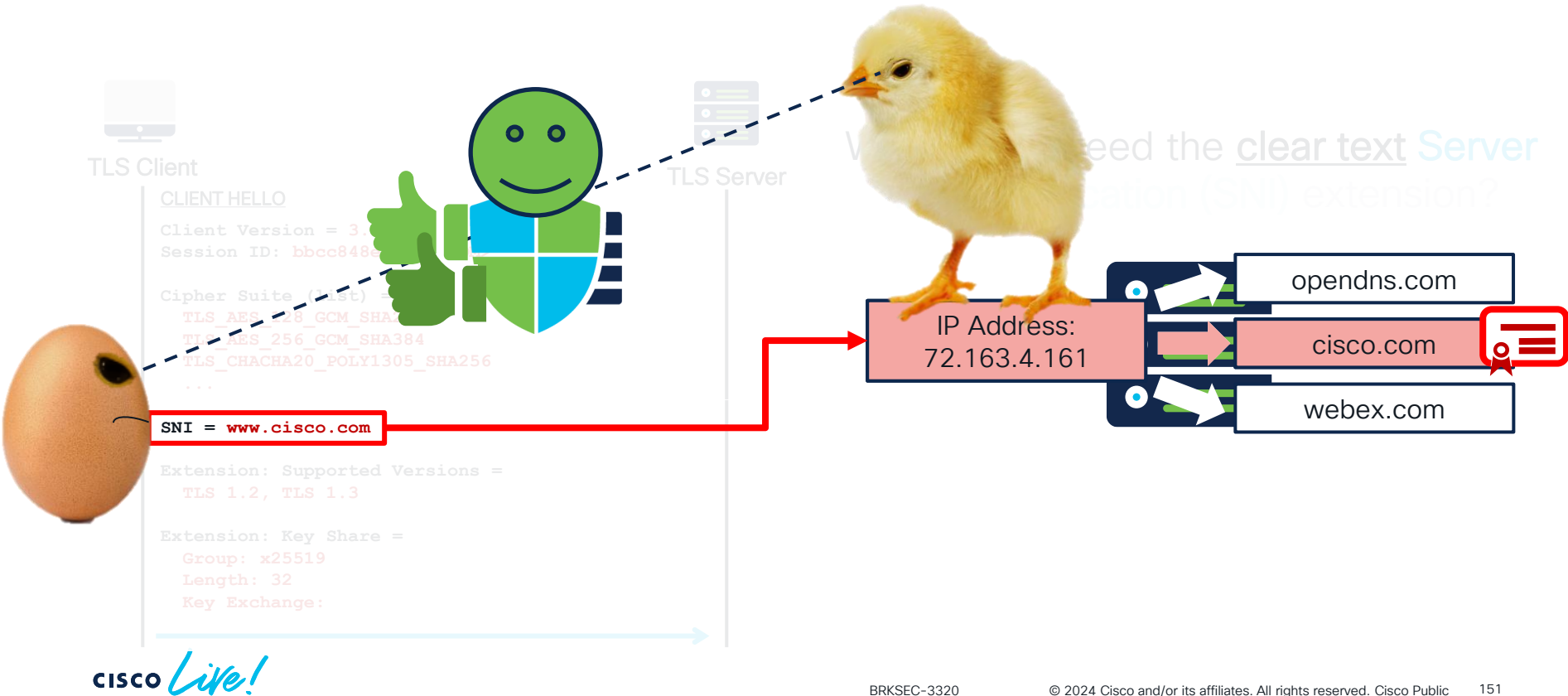
Understanding a TLS Session Flow – Server Name Indication (SNI)



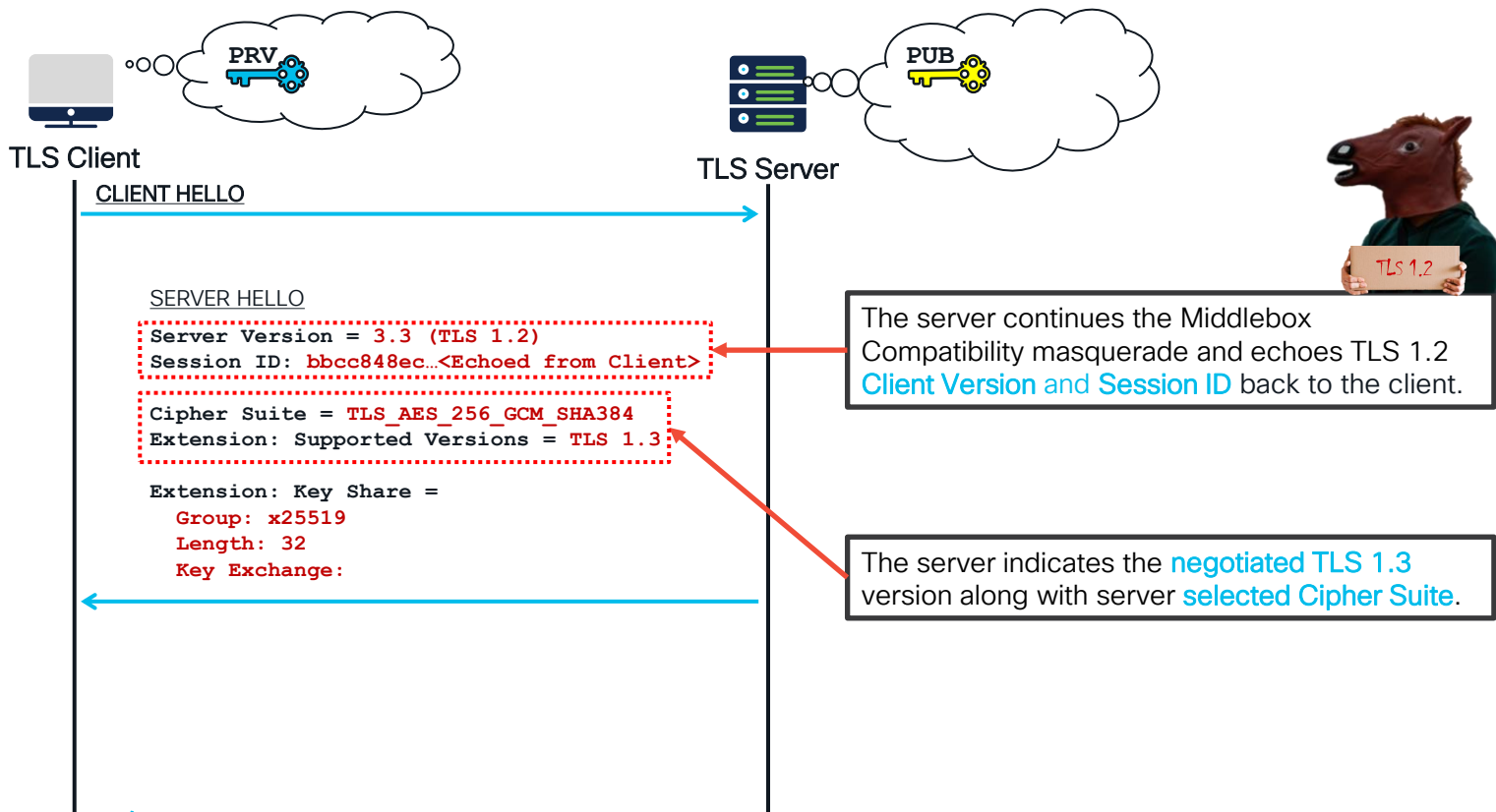
Why do we need the clear text **Server Name Indication (SNI)** extension?



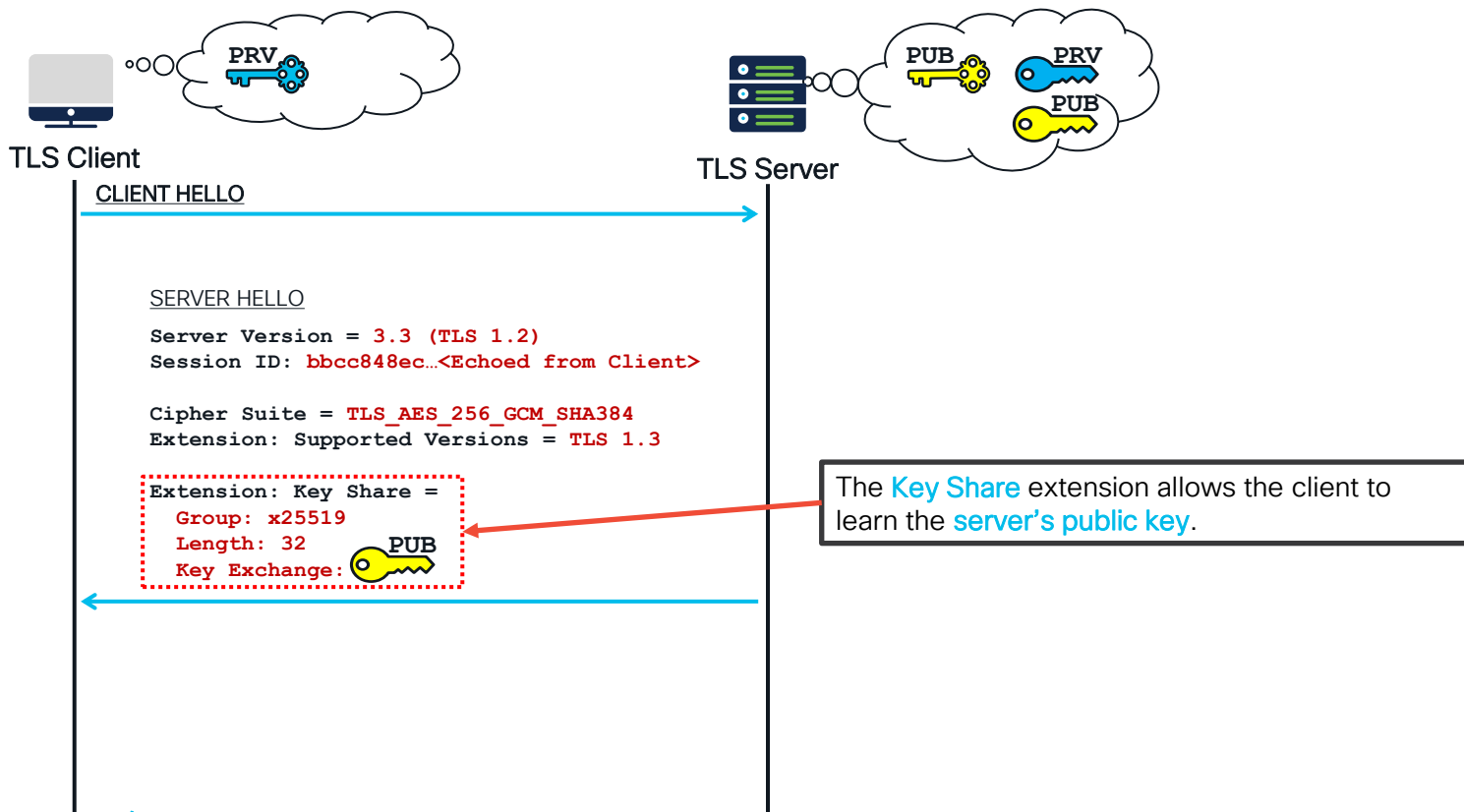
Understanding a TLS Session Flow – Server Name Indication (SNI)



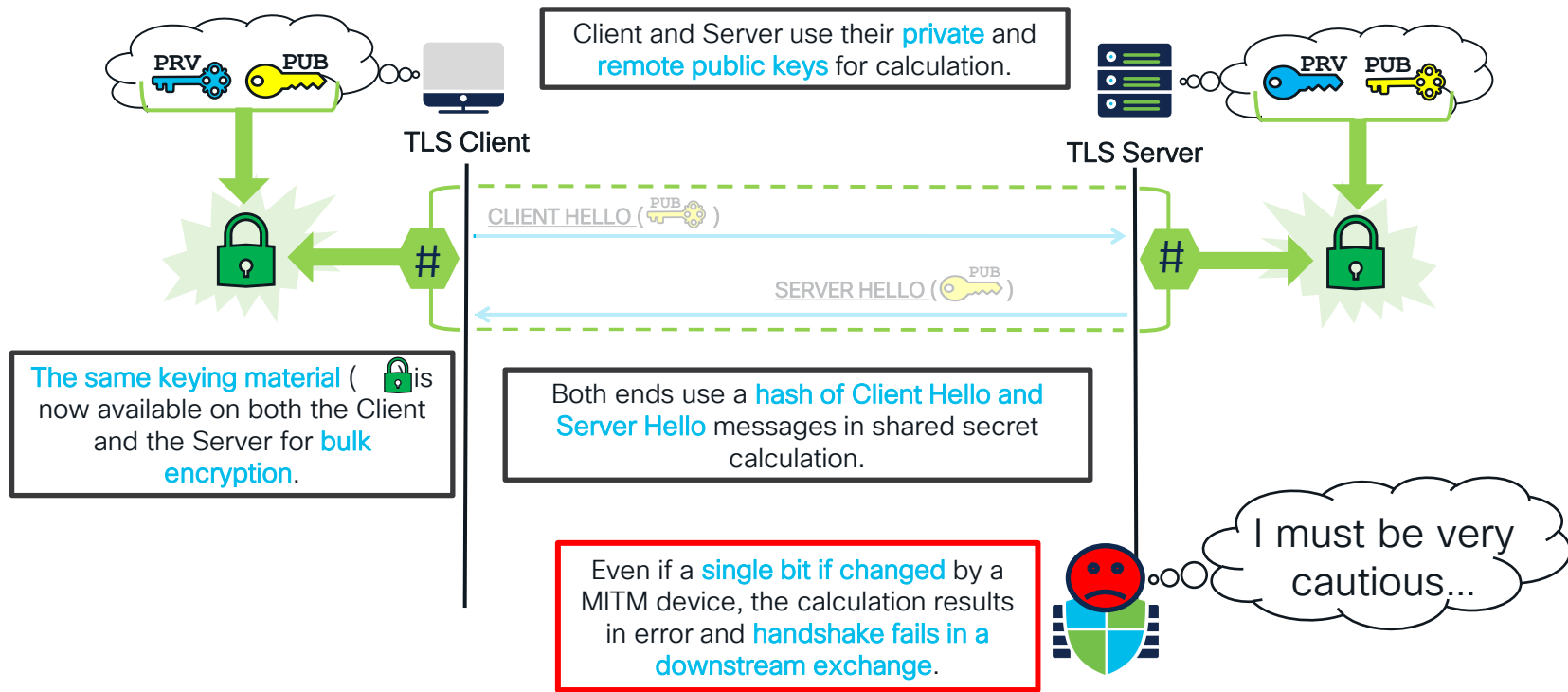
Understanding a TLS Session Flow – Server Hello



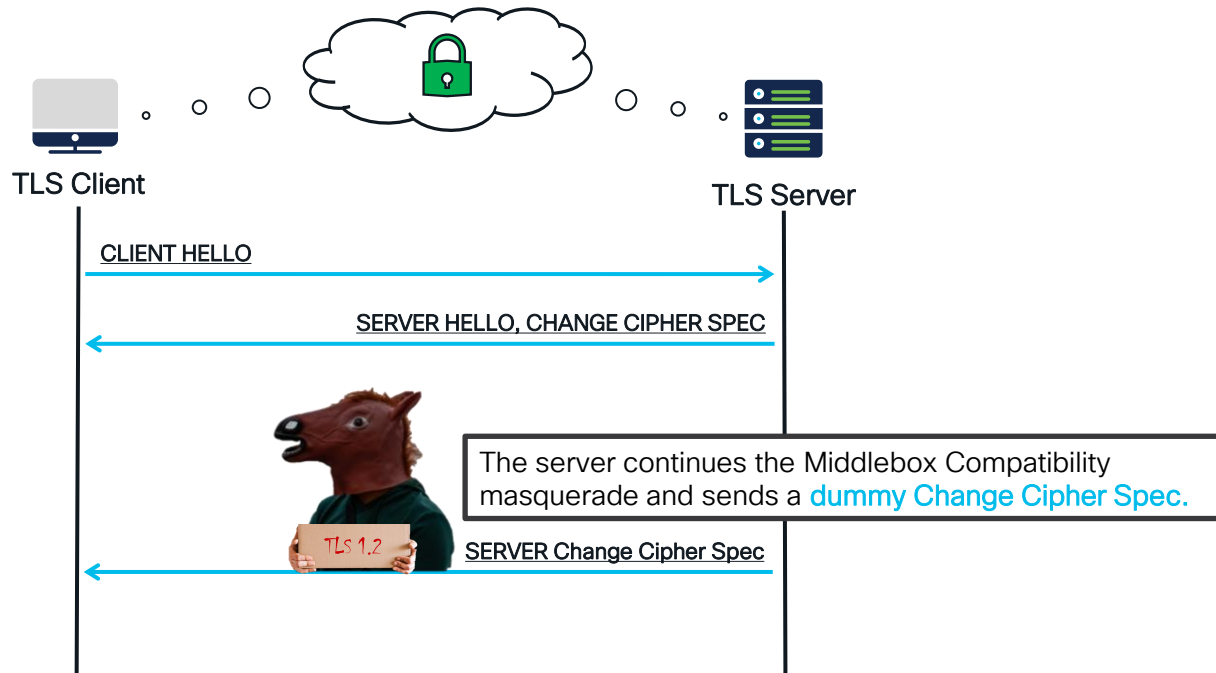
Understanding a TLS Session Flow – Server Hello



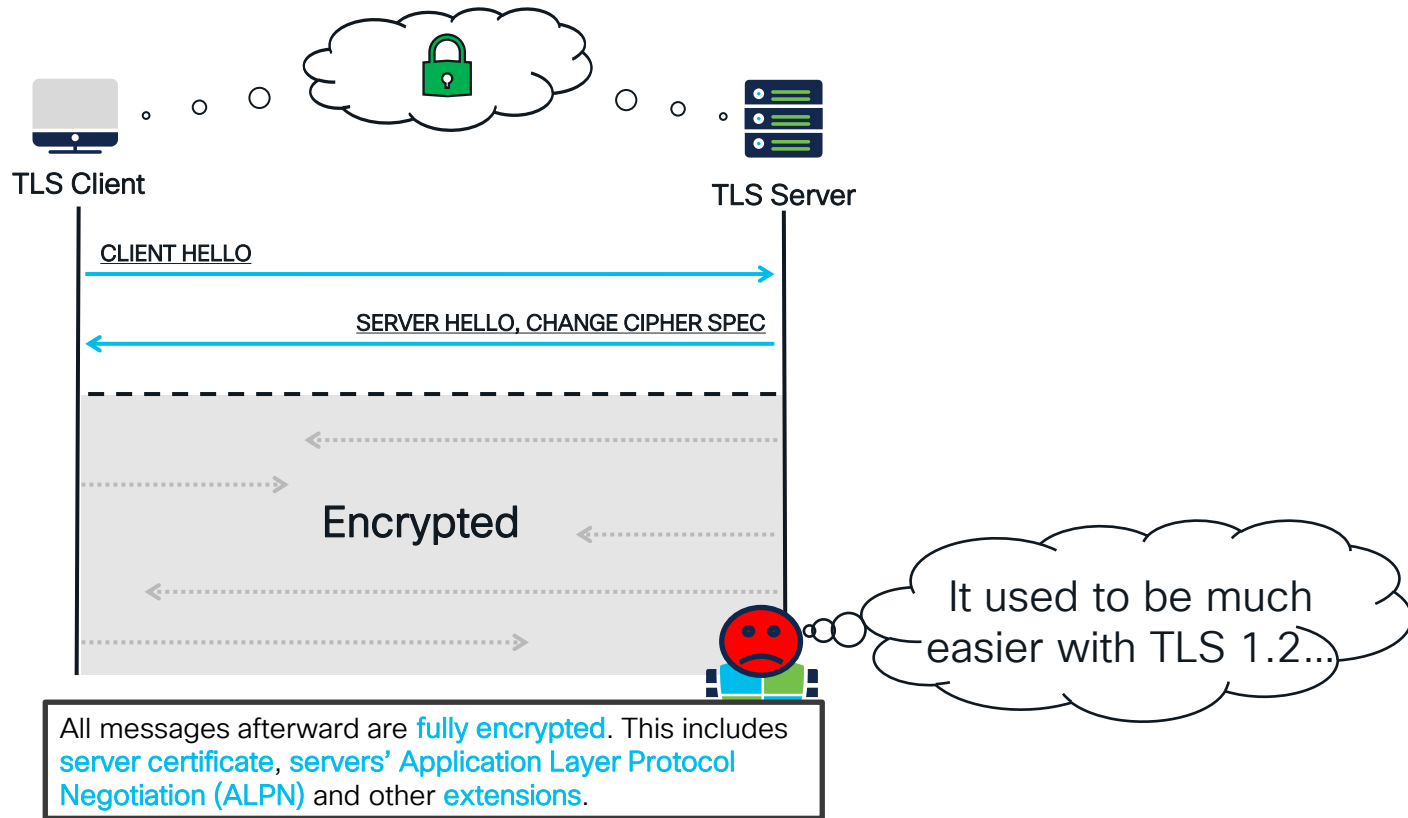
Understanding a TLS Session Flow – Calculating the Shared Keying Material



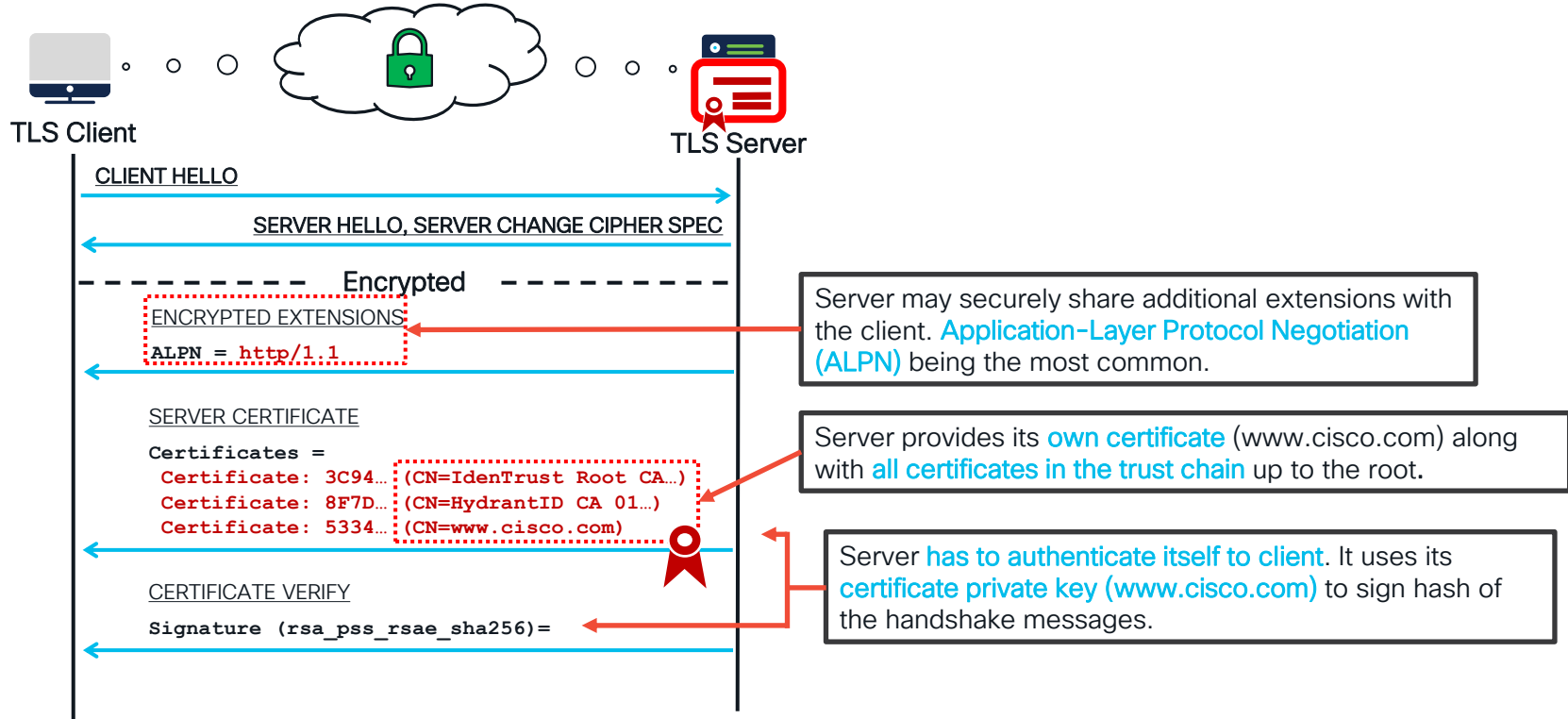
Understanding a TLS Session Flow – Encrypted Handshake



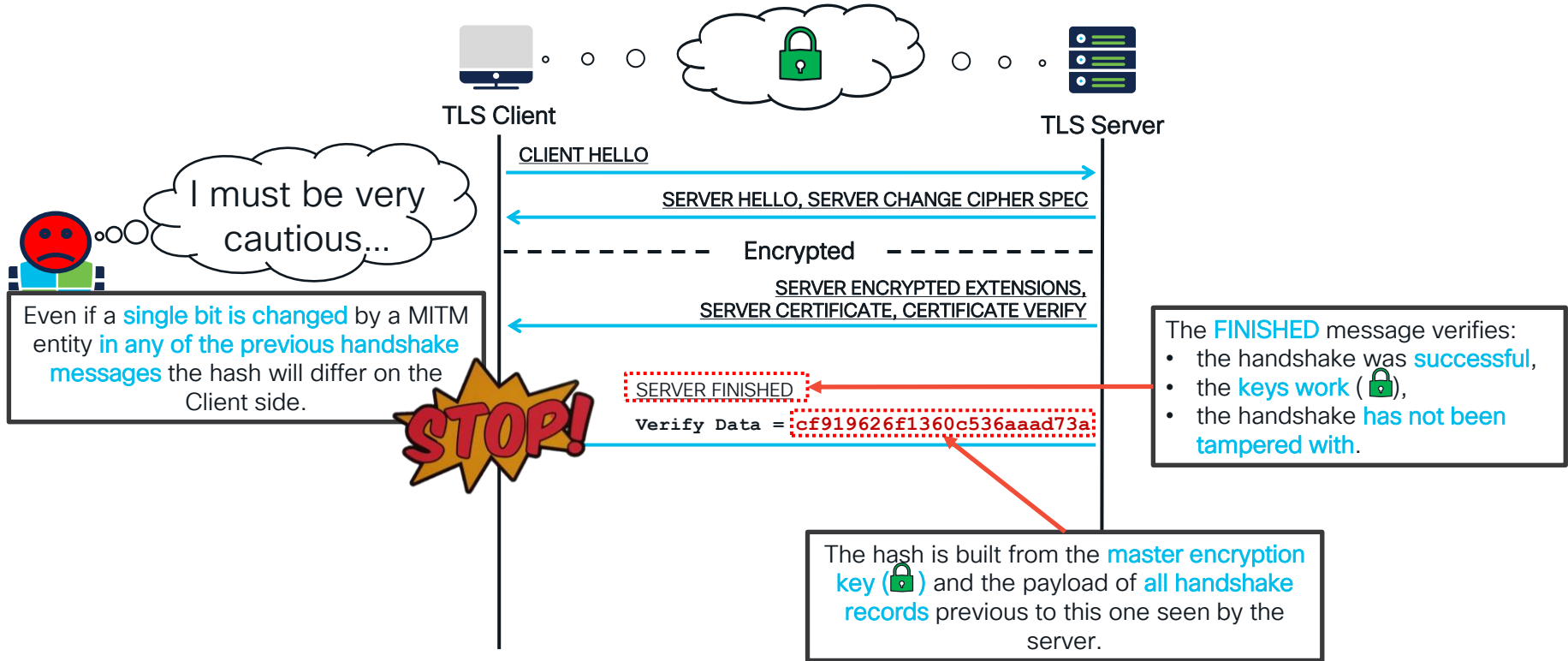
Understanding a TLS Session Flow – Encrypted Handshake



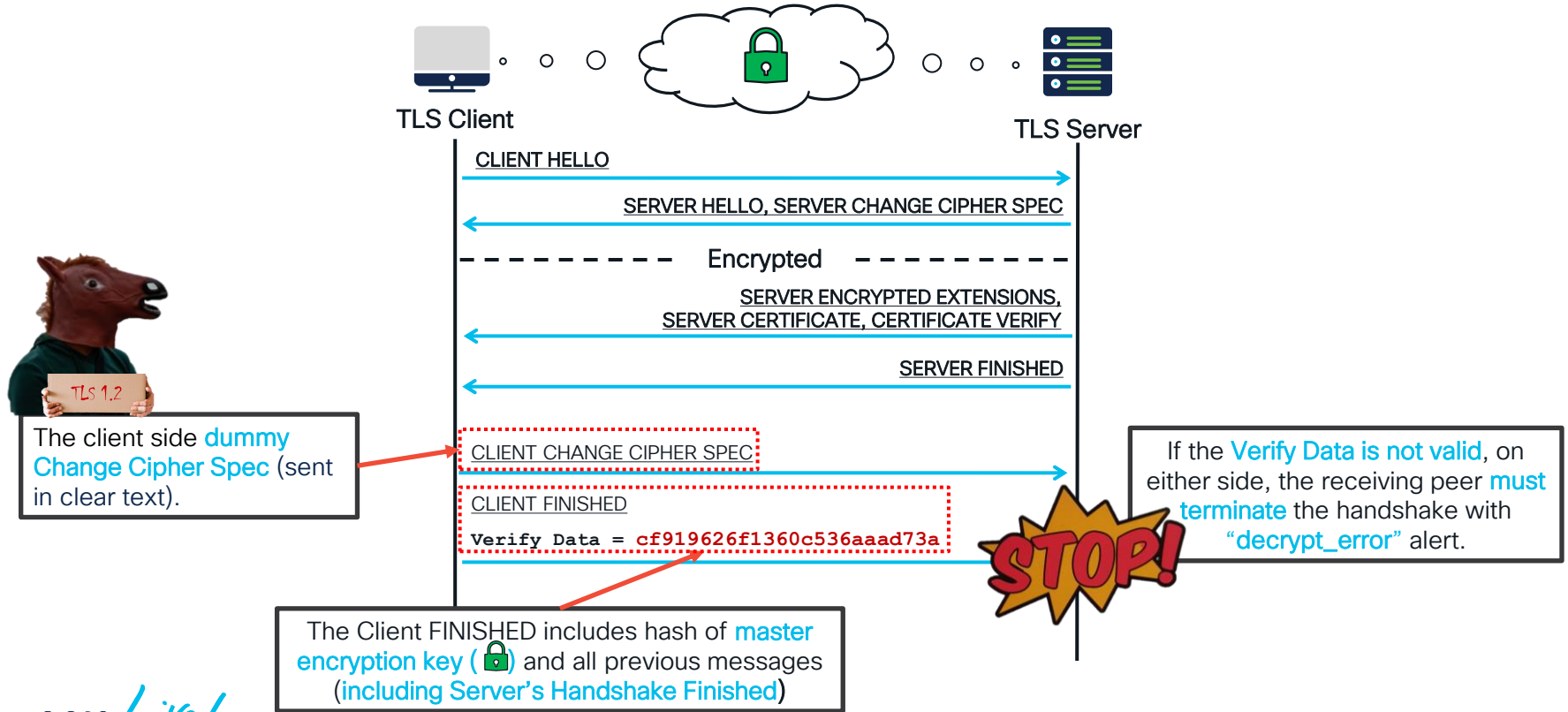
Understanding a TLS Session Flow – Encrypted Extensions and Certificate



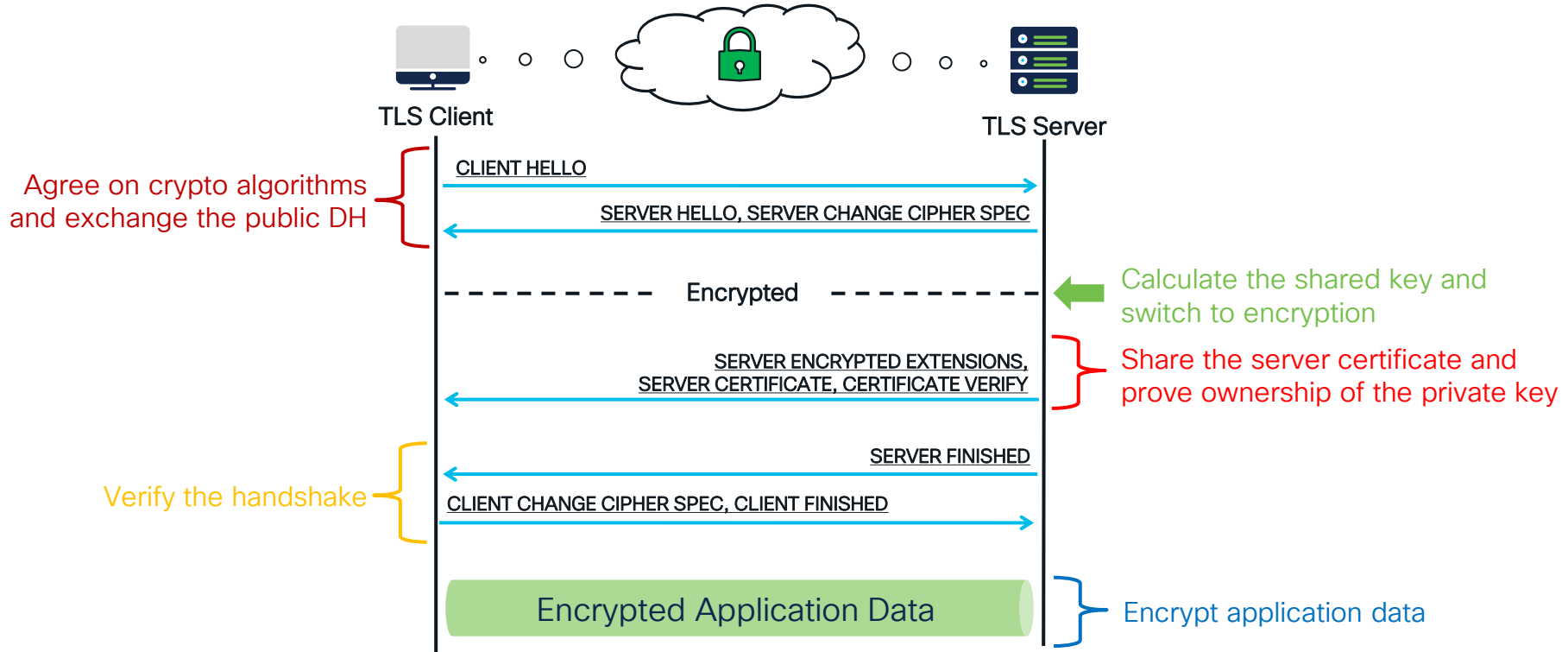
Understanding a TLS Session Flow – Server Finished



Understanding a TLS Session Flow – Client Side Finish



Understanding a TLS Session Flow – Client Side Finish



TLS Decryption Debugging

Enable the TLS debugs

```
# system support ssl-debug / ssl-debug-reset
```

The debug outputs are dumped to a local file:

```
/ngfw/var/common/xtls_log_NNNNN_NNNNN.log
```

```
22/06/29 14:56:20.475 [4054:1330]xtls_rules.cc:get_client_hello_modify_verdict:293: [DEBUG]: 172.16.12.20 63579 --
89.238.73.97 443 Processing ACTION_DECRYPT_AND_RESIGN rule 2 (Decrypt Issued by R3) (1100001000000000000000001000000000)

22/06/29 14:56:20.475 [4054:1330]xtls_rules.cc:get_client_hello_modify_verdict:343: [DEBUG]: 172.16.12.20 63579 --
89.238.73.97 443 Rule 2 (Decrypt Issued by R3) has issuer DN conditions, continuing search

22/06/29 14:56:20.475 [4054:1330]xtls_rules.cc:get_client_hello_modify_verdict:616: [INFO]: 172.16.12.20 63579 --
89.238.73.97 443 No rules matched, decision is don't modify

22/06/29 14:56:20.475 [4054:1330]xtls_rules.cc:get_client_hello_modify_verdict:622: [INFO]: 172.16.12.20 63579 --
89.238.73.97 443 Default rule determined definitive DND
```

Always set IP/Port filters:

```
> system support ssl-debug
Please specify a logging level (1-7) [6]: 7
Please specify modules [all]:
Please specify a client IP address: 172.16.136.96
Please specify a client port:
Please specify a server IP address: 89.238.73.97
Please specify a server port: 443
```



How to Survive Reading TLS Debugs with Notepad++

```

23/10/30 08:53:36.656 [5536:35799]xtls_flow.cc:process_packet:165: [DEBUG]: 172.16.136.96 58479
23/10/30 08:53:36.657 [5536:35799]length [517]
[5536:35799]16 03 01 02 00 01 00 01 fc 03 03 df 9f 1f 49 5a .....IZ
[5536:35799]a6 22 39 d2 6e 2e 0a c4 88 cc ff 3a 9a 75 ed f1 .."9.n.....u..
[5536:35799]47 db c5 10 a3 76 bb aa 8e 21 a0 20 96 4f 16 13 G....v....!..O..
[5536:35799]8c 0d 22 54 a6 de d9 f8 23 1a ef cf 1f e4 ff b3 .."T....#.....
23/10/30 08:53:36.657 [5536:35799]xtls_record_manager.cc:process_packet:82: [TRACE]: 172.16.136.96 58479
23/10/30 08:53:36.657 [5536:35799]xtls_record_queue.cc:add_record:55: [VERBOSE]: 172.16.136.96 58479
available 512
23/10/30 08:53:36.657 [5536:35799]xtls_record_queue.cc:get_next_message:404: [VERBOSE]: 172.16.136.96 58479
23/10/30 08:53:36.661 [5536:35801]xtls_flow.cc:process_packet:165: [DEBUG]: 172.16.136.96 15844 -- 72.163.4.161 443 72.163.4.161
23/10/30 08:53:36.661 [5536:35801]xtls_record_manager.cc:process_packet:82: [TRACE]: 172.16.136.96 15844 -- 72.163.4.161 443 F:s->c tracking 5429 proc
23/10/30 08:53:36.661 [5536:35801]xtls_record_queue.cc:add_record:55: [VERBOSE]: 172.16.136.96 15844 -- 72.163.4.161 443 F:s->c adding a record of size 5424
available 512
23/10/30 08:53:36.661 [5536:35801]xtls_record_queue.cc:get_next_message:404: [VERBOSE]: 172.16.136.96 15844 -- 72.163.4.161 443 72.163.4.161
23/10/30 08:53:37.071 [5536:35801]xtls_flow.cc:process_packet:165: [DEBUG]: 172.16.136.96 15844 -- 72.163.4.161 443 72.163.4.161
23/10/30 08:53:37.073 [5536:35801]xtls_record_manager.cc:process_packet:82: [TRACE]: 172.16.136.96 15844 -- 72.163.4.161 443 F:s->c tracking 5429 proc
23/10/30 08:53:37.073 [5536:35801]xtls_record_queue.cc:add_record:55: [VERBOSE]: 172.16.136.96 15844 -- 72.163.4.161 443 F:s->c adding a record of size 5424
available 5424
23/10/30 08:53:37.075 [5536:35801]xtls_server_hello_processor.cc:process:29: [TRACE]: 172.16.136.96 15844 -- 72.163.4.161 443 server_hello: len [87]
[5536:35801]172.16.136.96 15844 -- 72.163.4.161 443 version: 3.3
[5536:35801]172.16.136.96 15844 -- 72.163.4.161 443 random: 653f6f1025a32da0493d55a4bffa7cc43459553b4129b2ea8720b1da37d06eb2
[5536:35801]172.16.136.96 15844 -- 72.163.4.161 443 session id [32]: 035c491015a9b70203363469921914c7fe11c79a77f9d811ad7040a43190a66
[5536:35801]172.16.136.96 15844 -- 72.163.4.161 443 cipher_suite: [c02f] TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
[5536:35801]172.16.136.96 15844 -- 72.163.4.161 443
[5536:35801]172.16.136.96 15844 -- 72.163.4.161 443 ---extensions---
[5536:35801]172.16.136.96 15844 -- 72.163.4.161 443 alpn_extension[16]: len[5] alpn_list_len[3] ALPN list Entries: h2
[5536:35801]172.16.136.96 15844 -- 72.163.4.161 443 ec_point_formats[11]: len[2] 00

```

TLS debug dumps logs
on a **per packet basis**.

TIP: Start with finding the **beginning**
of individual TLS packets.

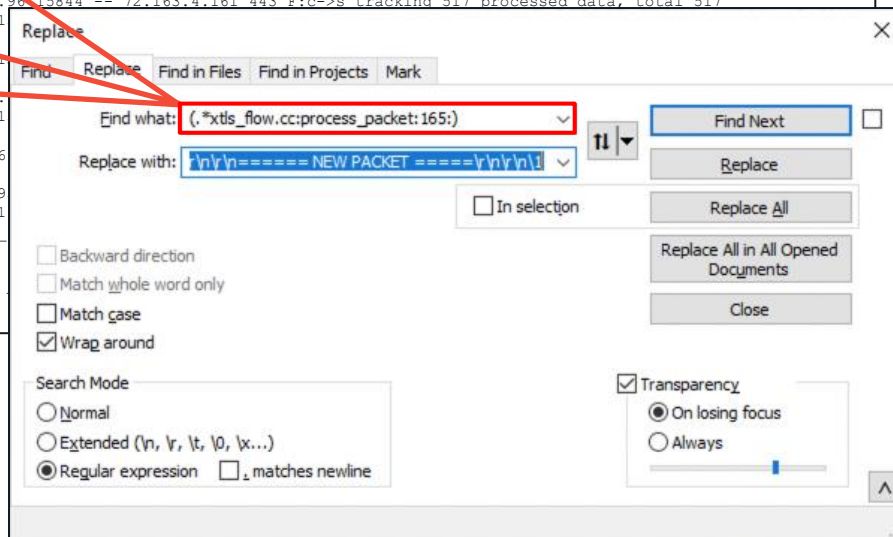


Use the Find & Replace with Regular Expressions

```

23/10/30 08:53:36.656 [5536:35799]xtls_flow.cc:process_packet:165: [DEBUG]: 172.16.136.96 58479 -- 72.163.4.161 443 172.16.136.96 58479 -> 72.163.4.161 443 length [517]
23/10/30 08:53:36.657 [5536:35799]length [517]
[5536:35799]16 03 01 02 00 01 00 01 fc 03 03 df 9f 1f 49 5a .....IZ
[5536:35799]a6 22 39 d2 6e 2e 0a c4 88 cc ff 3a 9a 75 ed f1 .....u..
[5536:35799]47 db c5 10 a3 76 bb aa 8e 21 a0 20 96 4f 16 13 G....v...!. .8.
[5536:35799]8c 0d 22 54 a6 de d9 f8 23 1a ef cf 1f e4 ff b3 .."T....#.....
23/10/30 08:53:36.657 [5536:35799]xtls_record_manager.cc:process_packet:82: [TRACE]: 172.16.136.96 58479 -- 72.163.4.161 443 F:c->s tracking 517 processed data, total 517
23/10/30 08:53:36.657 [5536:35799]xtls_record_queue.cc:add_record:55: [VERBOSE]: 172.16.136.96 58479 -- 72.163.4.161 443 F:c->s adding a record of size 517 payload size 512 total
available 512
23/10/30 08:53:36.657 [5536:35799]xtls_record_queue.cc:get_next_message:404: [VERBOSE]: 172.16.136.96 58479 -- 72.163.4.161 443 F:c->s yielding a handshake message of size 512
23/10/30 08:53:36.661 [5536:35801]xtls_flow.cc:process_packet:165: [DEBUG]: 172.16.136.96 15844 -- 72.163.4.161 443 172.16.136.96 15844 -> 72.163.4.161 443 length [517]
23/10/30 08:53:36.661 [5536:35801]xtls_record_manager.cc:process_packet:82: [TRACE]: 172.16.136.96 15844 -- 72.163.4.161 443 F:c->s tracking 517 processed data, total 517
23/10/30 08:53:36.661 [5536:35801]xtls_record_queue.cc:add_record:55: [VERBOSE]: 172.16.136.96 1
available 512
23/10/30 08:53:36.661 [5536:35801]xtls_record_queue.cc:get_next_message:404: [VERBOSE]: 172.16.1
23/10/30 08:53:37.071 [5536:35801]xtls_flow.cc:process_packet:165: [DEBUG]: 172.16.136.96 15844
23/10/30 08:53:37.073 [5536:35801]xtls_record_manager.cc:process_packet:82: [TRACE]: 172.16.136.
23/10/30 08:53:37.073 [5536:35801]xtls_record_queue.cc:add_record:55: [VERBOSE]: 172.16.136.96 1
available 5424
23/10/30 08:53:37.075 [5536:35801]xtls_server_hello_processor.cc:process:29: [TRACE]: 172.16.136
[5536:35801]172.16.136.96 15844 -- 72.163.4.161 443 version: 3.3
[5536:35801]172.16.136.96 15844 -- 72.163.4.161 443 random: 653f6f1025a32da0493d55a4bffa7cc43459
[5536:35801]172.16.136.96 15844 -- 72.163.4.161 443 session id [32]: 035c491015a9b70203363469921
[5536:35801]172.16.136.96 15844 -- 72.163.4.161 443 cipher_suite: [c02f] TLS_ECDHE_RSA_WITH_AES_
[5536:35801]172.16.136.96 15844 -- 72.163.4.161 443
[5536:35801]172.16.136.96 15844 -- 72.163.4.161 443 ---extensions---
[5536:35801]172.16.136.96 15844 -- 72.163.4.161 443 alpn_extension[16]: len[5] alpn_list_len[3]
[5536:35801]172.16.136.96 15844 -- 72.163.4.161 443 ec_point_formats[11]: len[2] 00

```



Now It Is Easier to Read

===== NEW PACKET =====

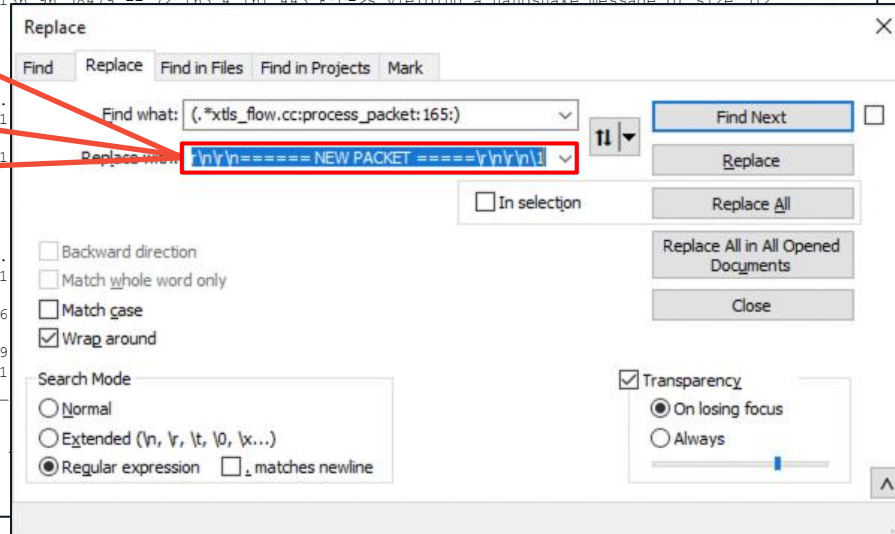
```
23/10/30 08:53:36.656 [5536:35799]xtls_flow.cc:process_packet:165: [DEBUG]: 172.16.136.96 58479 -- 72.163.4.161 443 172.16.136.96 58479 -> 72.163.4.161 443 length [517]
23/10/30 08:53:36.657 [5536:35799]length [517]
[5536:35799]16 03 01 02 00 01 00 01 fc 03 03 9f 9f 1f 49 5a .....IZ
[5536:35799]a6 22 39 d2 6e 2e 0a c4 88 cc ff 3a 9c 75 ed f1 .."9.n.....u..
[5536:35799]47 db c5 10 a3 76 bb aa 8e 21 a0 20 96 41 16 13 G...v...!. .O..
[5536:35799]8c 0d 22 54 a6 de d9 f8 23 1a ef cf 1f e4 ff d3 .."T....#.....
23/10/30 08:53:36.657 [5536:35799]xtls_record_manager.cc:process_packet:82: [TRACE]: 172.16.136.96 58479 -- 72.163.4.161 443 F:c->s tracking 517 processed data, total 517
23/10/30 08:53:36.657 [5536:35799]xtls_record_queue.cc:add_record:55: [VERBOSE]: 172.16.136.96 58479 -- 72.163.4.161 443 F:c->s adding a record of size 517 payload size 512 total
available 512
23/10/30 08:53:36.657 [5536:35799]xtls_record_queue.cc:get_next_message:404: [VERBOSE]: 172.16.136.96 58479 -- 72.163.4.161 443 F:c->s yielding a handshake message of size 512
```

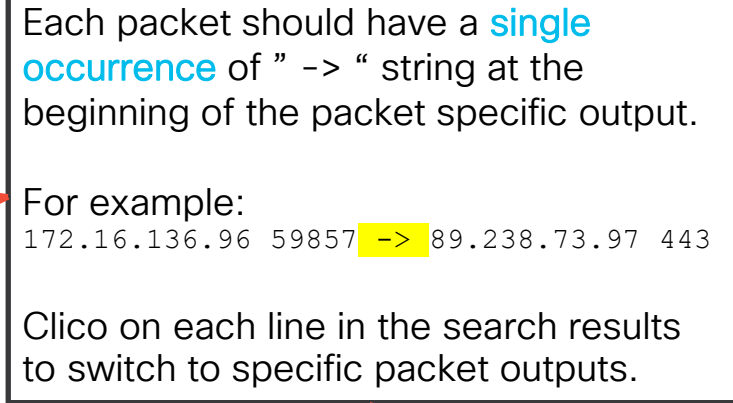
===== NEW PACKET =====

```
23/10/30 08:53:36.661 [5536:35801]xtls_flow.cc:process_packet:165: [DEBUG]: 172.16.136.96 15844
23/10/30 08:53:36.661 [5536:35801]xtls_record_manager.cc:process_packet:82: [TRACE]: 172.16.136.
23/10/30 08:53:36.661 [5536:35801]xtls_record_queue.cc:add_record:55: [VERBOSE]: 172.16.136.96 1
available 512
23/10/30 08:53:36.661 [5536:35801]xtls_record_queue.cc:get_next_message:404: [VERBOSE]: 172.16.1
```

===== NEW PACKET =====

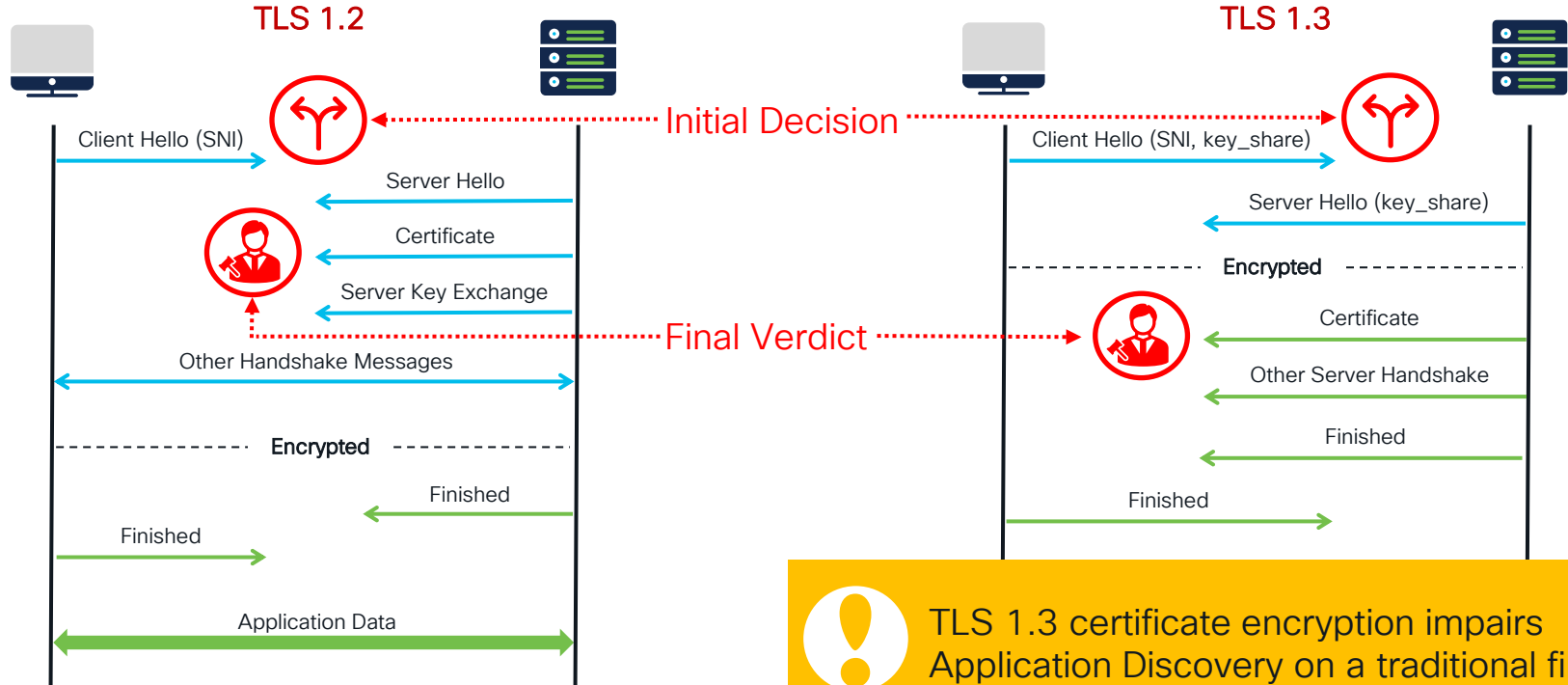
```
23/10/30 08:53:37.071 [5536:35801]xtls_flow.cc:process_packet:165: [DEBUG]: 172.16.136.96 15844
23/10/30 08:53:37.073 [5536:35801]xtls_record_manager.cc:process_packet:82: [TRACE]: 172.16.136.
23/10/30 08:53:37.073 [5536:35801]xtls_record_queue.cc:add_record:55: [VERBOSE]: 172.16.136.96 1
available 5424
23/10/30 08:53:37.075 [5536:35801]xtls_server_hello_processor.cc:process:29: [TRACE]: 172.16.136
[5536:35801]172.16.136.96 15844 -- 72.163.4.161 443 version: 3.3
[5536:35801]172.16.136.96 15844 -- 72.163.4.161 443 random: 653f6f1025a32da0493d55a4bffa7cc43459
[5536:35801]172.16.136.96 15844 -- 72.163.4.161 443 session id [32]: 035c491015a9b70203363469921
[5536:35801]172.16.136.96 15844 -- 72.163.4.161 443 cipher_suite: [c02f] TLS_ECDHE_RSA_WITH_AES_
[5536:35801]172.16.136.96 15844 -- 72.163.4.161 443
[5536:35801]172.16.136.96 15844 -- 72.163.4.161 443 ---extensions---
[5536:35801]172.16.136.96 15844 -- 72.163.4.161 443 alpn_extension[16]: len[5] alpn_list_len[3]
[5536:35801]172.16.136.96 15844 -- 72.163.4.161 443 ec_point_formats[11]: len[2] 00
```





TLS Decryption Challenges

TLS 1.2 vs 1.3 Handshake



Under the hood: TLS Probing

We receive a **Client Hello** towards a **non-cached TLS server**.

```
[INF] 172.16.12.20 59087 -- 89.238.73.97 443 Evaluating CH modify decision for flow 172.16.12.20 59087 -- 89.238.73.97 443
[INF] 172.16.12.20 59087 -- 89.238.73.97 443 did not get back a cert for client hello
```

The policy evaluation results in a **don't modify decision**...

```
[INFO]: 172.16.12.20 59087 -- 89.238.73.97 443 No rules matched, decision is don't modify
```

...nevertheless, **we trigger the TLS Probe** to acquire the certificate.

```
[TRACE]: 172.16.12.20 59087 -- 89.238.73.97 443 certificate visibility probe requested
```

Firewall sends the Client Hello over a **new TCP socket**.

```
[DEBUG] 172.16.12.20 31143 -> 89.238.73.97 443 length [517]
[TRACE] 172.16.12.20 31143 -> 89.238.73.97 443 client_hello: len [508]
[DEBUG] 172.16.12.20 31143 -> 89.238.73.97 443 client hello modify automatic for probe flow
```

TLS 1.2
Client Hello

The **certificate** received over the Probe connection **is cached**.

```
[TRACE]: 172.16.12.20 31143 -- 89.238.73.97 443 certificate: len [4275]
```

```
[TRACE]: 172.16.12.20 31143 -- 89.238.73.97 443 original certificate added to cache
```

Server
Certificate

The **Probe connection** serves no additional purpose; hence it **gets reset**.

```
172.16.12.20 31143 -- 89.238.73.97 443 certificate visibility probe flow being reset
```

```
[TRACE]: 172.16.12.20 59087 -- 89.238.73.97 443 F:c->s retry packet, reset record to previous
```

Firewall **re-evaluates the policy** having the **certificate available** in the **cache**.

```
[INF] 172.16.12.20 59087 -- 89.238.73.97 443 Evaluating CH modify decision for flow 172.16.12.20 59087 -- 89.238.73.97 443
[INF] 172.16.12.20 59087 -- 89.238.73.97 443 Got back a cert for client hello
```

The **initial Client hello** is **re-injected** for processing.

```
[INFO]: 172.16.12.20 59087 -- 89.238.73.97 443 Matched rule: Decrypt Issued by R3, decision is modify
```

The final policy evaluation results in a **modify decision**

Under the hood: Certificate Cache

```
> system support ssl-cache-export
```

```
Getting server certificates...Done. File ssl_server_certs.txt was  
successfully created at /ngfw/var/common/ folder.
```

```
> expert
```

```
admin@csftd:~$ cat /ngfw/var/common/ssl_server_certs.txt
```

```
Cache Age (s): 25
```

```
Subject: CN=secure.eicar.org
```

```
-----BEGIN CERTIFICATE-----
```

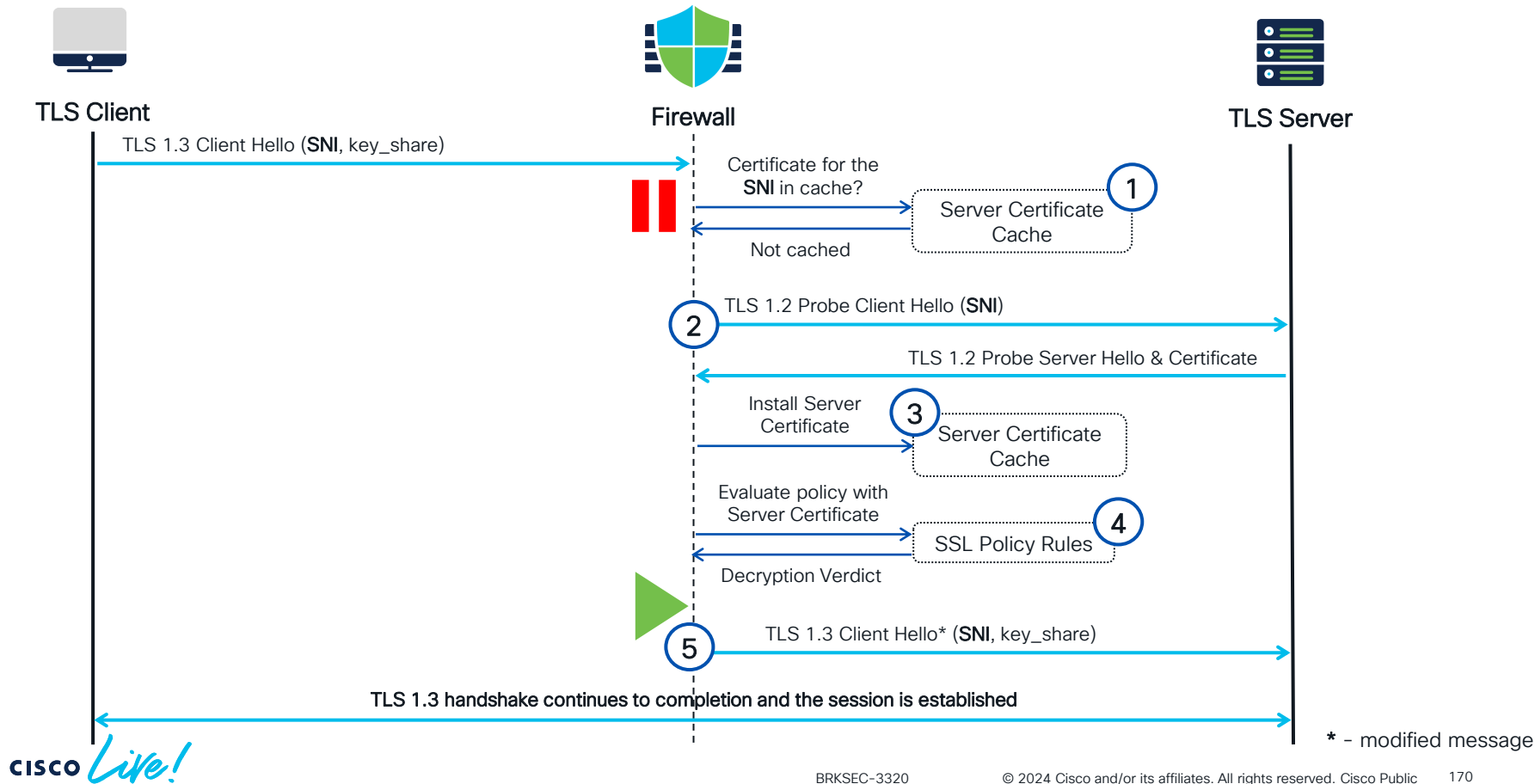
```
MIIGJTCCBQ2gAwIBAgISBH2TXac0/SMQabVwlmh83VG4MA0GCSqGSIb3DQEBCwUA
```

```
[...]
```

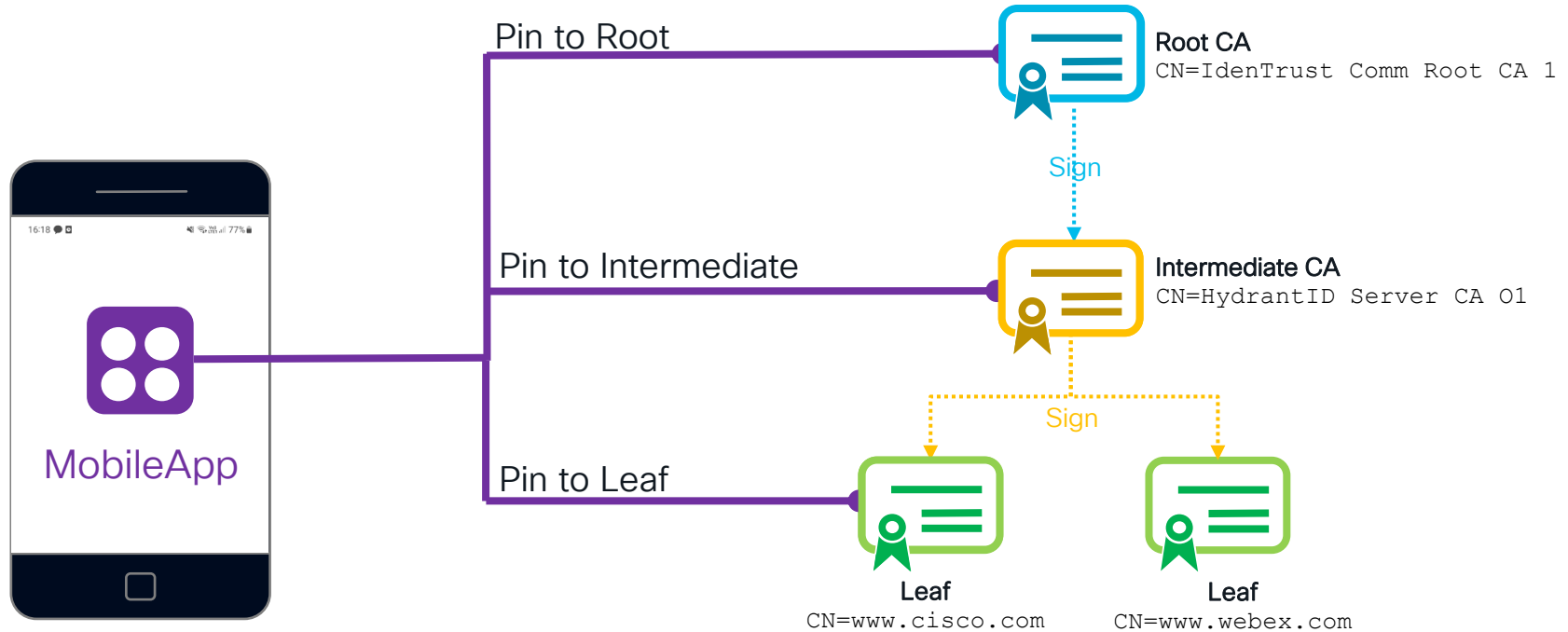
```
a9eI9KBs/gfDPWXqn0rrGXPhSixOmBPpzVPpSl9Y9/KLrxYk9jA8pOQ=
```

```
-----END CERTIFICATE-----
```

TLS Server Cache & Probing



What is Certificate Pinning?

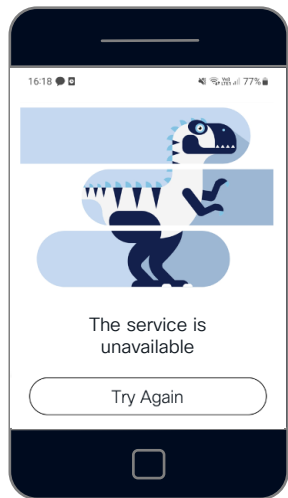


Why is Certificate Pinning a Problem?



is signed by a system trusted Root CA...

...but I have this one hardcoded:



Firewall



TLS Server



TLS 1.2 Client Hello (SNI = login.edenred.io)

TLS 1.2 Client Hello* (SNI = login.edenred.io)

TLS 1.2 Server Hello*

TLS 1.2 Server Hello

TLS 1.2 Server Certificate*

TLS 1.2 Server Certificate

TLS 1.2 Server Key Exchange*

TLS 1.2 Server Key Exchange

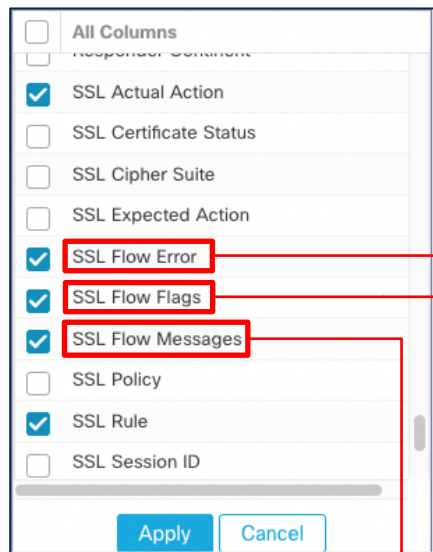
TLS 1.2 Server Hello Done*

TLS 1.2 Server Hello Done

TLS ALERT (optional) + TCP RST

* - modified message

Troubleshooting Certificate Pinning



Applications sending TLS Alert + TCP Reset

Applications sending TCP Reset only



Packet Capture

Troubleshooting Certificate Pinning

☐ All Columns

☐ Response Comment

☒ SSL Actual Action

☐ SSL Certificate Status

☐ SSL Cipher Suite

☐ SSL Expected Action

☒ SSL Flow Error

☒ SSL Flow Flags

☒ SSL Flow Messages

☐ SSL Policy

☒ SSL Rule

☐ SSL Session ID

Applications sending TLS Alert + TCP Reset	Applications sending TCP Reset only
ALERT_SEEN no APP_DATA_C2S no APP_DATA_S2C	no ALERT_SEEN no APP_DATA_C2S no APP_DATA_S2C
CLIENT_ALERT CLIENT_HELLO SERVER_HELLO SERVER_CERT SERVER_KEY_EXCH SERVER_HELLO_DONE	CLIENT_HELLO SERVER_HELLO SERVER_CERT SERVER_KEY_EXCH SERVER_HELLO_DONE Optionally: CLIENT_CHANGE_CIPHER_SPEC CLIENT_FINISHED SERVER_CHANGE_CIPHER_SPEC SERVER_FINISHED
SUCCESS	SUCCESS
Client sourced TLS Alert (clear text) Client sourced TCP Reset	Client sourced TCP Reset only Optionally: Client sourced TLS Alert (encrypted)



Packet Capture

Troubleshooting Certificate Pinning - Capture

The image shows a Wireshark packet capture of a TLS connection. A smartphone icon labeled 'MobileApp' is overlaid on the left, with arrows indicating the flow of data. The packet list on the right shows the following:

No.	Time	Source	Info	SPORT	DPORT	Length	Protocol
4	0.000030	192.168.10.37	Client Hello	46616	443	583	TLSv1.2
6	0.028746	3.120.204.252	Server Hello	443	46616	164	TLSv1.2
11	0.000016	3.120.204.252	Certificate	443	46616	93	TLSv1.2
18	0.001342	3.120.204.252	Server Key Exchange	443	46616	404	TLSv1.2
19	0.000016	3.120.204.252	Server Hello Done	443	46616	75	TLSv1.2
21	0.002365	192.168.10.37	Alert (Level: Fatal, Description: Certificate Unknown)	46616	443	73	TLSv1.2
23	0.000320	192.168.10.37	46616 → 443 [RST, ACK] Seq=525 Ack=3464 Win=79872 Len=0...	46616	443	66	TCP
24	0.005142	192.168.10.37	46616 → 443 [RST] Seq=518 Win=0 Len=0	46616	443	54	TCP
25	0.000016	192.168.10.37	46616 → 443 [RST] Seq=525 Win=0 Len=0	46616	443	54	TCP

Below the packet list, the details of Frame 21 are shown:

- Frame 21: 73 bytes on wire (584 bits), 73 bytes captured (584 bits)
- Ethernet II, Src: 8e:9f:aa:0e:8b:39 (8e:9f:aa:0e:8b:39), Dst: Cisco_f5:28:c9 (08:4f:a9:f5:28:c9)
- Internet Protocol Version 4, Src: 192.168.10.37, Dst: 3.120.204.252
- Transmission Control Protocol, Src Port: 46616, Dst Port: 443, Seq: 518, Ack: 3464, Len: 7
- Transport Layer Security
 - TLSv1.2 Record Layer: Alert (Level: Fatal, Description: Certificate Unknown)
 - Content Type: Alert (21)
 - Version: TLS 1.2 (0x0303)
 - Length: 2
 - Alert Message
 - Level: Fatal (2)
 - Description: Certificate Unknown (46)

A red dashed box highlights the 'Alert Message' details, and a red arrow points from a text box to it.

The Alert Message indicates the spoofed re-signed certificate was not recognized by the application.

Under the Hood: Failure Due to Certificate Pinning

```
[DEBUG]: 192.168.10.37 46616 -- 3.120.204.252 443 192.168.10.37 46616 -> 3.120.204.252 443 length [517]
[TRACE]: 192.168.10.37 46616 -- 3.120.204.252 443 client_hello: len [508]
[...]
server_name[0]: len[32] server name indication: login.edenred.io

[INFO]: 192.168.10.37 46616 -- 3.120.204.252 443 Matched rule: Decrypt All from IP, decision is modify
```



Modified
Client Hello



Resigned
Server
Certificate

```
[DEBUG]: 192.168.10.37 46616 -- 3.120.204.252 443 3.120.204.252 443 -> 192.168.10.37 46616 length [98]
[TRACE]: 192.168.10.37 46616 -- 3.120.204.252 443 server_hello: len [89]
[TRACE]: 192.168.10.37 46616 -- 3.120.204.252 443 certificate: len [2817]
[TRACE]: 192.168.10.37 46616 -- 3.120.204.252 443 server_key_exchange: len [329]
[TRACE]: 192.168.10.37 46616 -- 3.120.204.252 443 server_hello_done: len [0]
```

```
[DEBUG]: 192.168.10.37 46616 -- 3.120.204.252 443 192.168.10.37 46616 -> 3.120.204.252 443 length [7]
[DEBUG]: 192.168.10.37 46616 -- 3.120.204.252 443 S:c->s saw record type [alert]
[TRACE]: 192.168.10.37 46616 -- 3.120.204.252 443 q size [0] next action from queue: F:c->s action
[log_connection_event] source [flow] module [ALERT_PROCESSOR]
```



**TLS
ALERT!!**

Pinned Applications Tag

Add Rule

Name: Bypass Decryption ☒ Enabled Insert: below rule 8

Action: **Do not decrypt**

Applications

Application Filters: Clear All Filters X

Available Applications (40)

Application	Count
NSG	1860
office 365	18
old/obsolete	1
opens port	4
pinned certificate	40
recent vulnerabilities	20
safesearch supported	11
safesearch unsupported	37

Selected Applications and Filters (0)

any

Callout 1: You can use this tag it to match application traffic that should bypass decryption.

Callout 2: Pinned Certificate applications tag is available in the SSL Policy.

Buttons: Cancel Add

Cisco Provided Undecryptable Sites (1/2)

Add Rule

Name: ☒ Enabled

Action: Do not decrypt

...and can create an exclusion rule with "Do not decrypt" action.

Zones Networks VLAN Tags Users Applications Ports Category Certificate **DN** Cert Status Cipher Suite Version Logging

Available DNs

- Cisco-Undecryptable-Sites**
- blah
- CN_api.smarththings.com
- CN_apps.apple.com
- CN_ciscopark.com
- CN_citrixonline.com
- CN_core.windows.net
- CN_data.microsoft.com

Add to Subject Add to Issuer

Subject DNs (0) Issuer DNs (0)

You can also use a pre-defined DN Object containing know Undecryptable Sites...

Enter DN or CN Add Enter DN or CN Add

Cancel Add

Cisco Provided Undecryptable Sites (2/2)

Firewall Management Center
Objects / Object Management

Overview Analysis Policies Devices Objects Integration

Deploy 🔍 ⚙️ 👤 admin | cisco SECURE

Object Groups

Add Distinguished Name Group 🔍 Filter

Each distinguished name object represents the distinguished name listed for a public key certificate's subject or issuer. You can use distinguished name object groups in SSL rules to control encrypted traffic based on whether the client and server negotiated the SSL session using a server certificate with the distinguished name as subject or issuer.

Name	Value
Cisco-Undecryptable-Sites	CN_sls.microsoft.com CN_deviceenrollment.apple.com CN_gs-loc.apple.com CN_vortex-win.data.microsoft.com CN_tbsc.apple.com There are 51 more items in this group...

Name: CN_ess.apple.com - Value :

Name: CN_apps.apple.com - Value :

Name: CN_pindorama.amazon.com - Value :

Name: CN_api.smarthings.com - Value :

Name: CN_android.clients.google.com - Value :

Name: CN_crl.entrust.net - Value :

Name: CN_logmein.com - Value :

Name: CN_latinum.amazon.com - Value :

Name: CN_data.microsoft.com - Value :

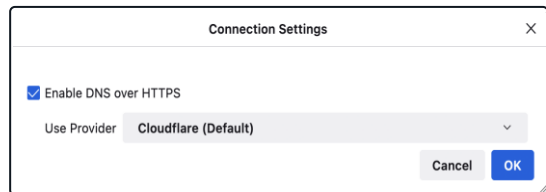
Name: CN_rhn.redhat.com - Value :

Name: CN_icloud.com - Value :

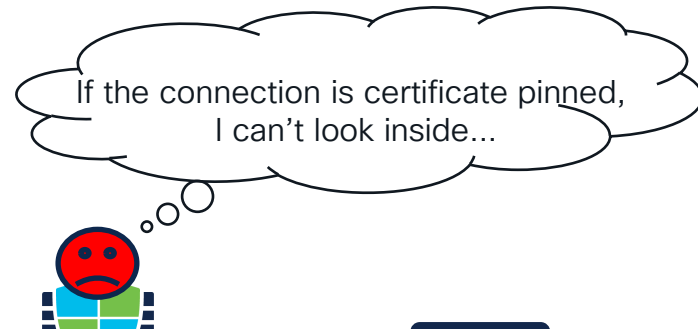
DNS over HTTPs

IETF standard (RFC8484) proposed to:

- allow web applications to access DNS information **via browser API**
- prevent **on-path devices from interfering** with DNS



Client



DNS over HTTPs Challenges

- The web browser hijacks OS DNS
- Bypass DNS based security controls and logging
- Delegates DNS control to a content provider (e.g. Cloudflare)
- Difficult to block by firewalls (SNI and/or IP based only)

DoH is a very **effective distribution method** of keying material for **SNI obfuscation techniques** like Encrypted SNI or Encrypted Client Hello.

Cloudflare
DoH Resolver

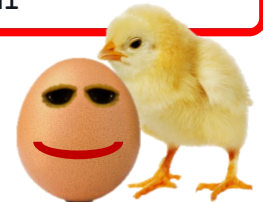


DNS Records:

A, AAAA - IPv4/v6 Addresses

SVCB/HTTPS RR - **Encrypted Client Hello**

TXT _esni - **Encrypted SNI**



DoH Blocking on FirePower

Add Rule

Name: ☒ Enabled Insert:

Action: Time Range:

Zones Networks VLAN Tags Users Applications Ports URLs Dynamic Attributes Inspection Logging Comments

Application Filters Clear All Filters X

Available Applications (1) X

All apps matching the filter

DNS over HTTPS

Add to Rule

Selected Applications and Filters (0)

any

Cancel Add

You can **block DNS over HTTPs** with a rule in your **Access Control Policy**.

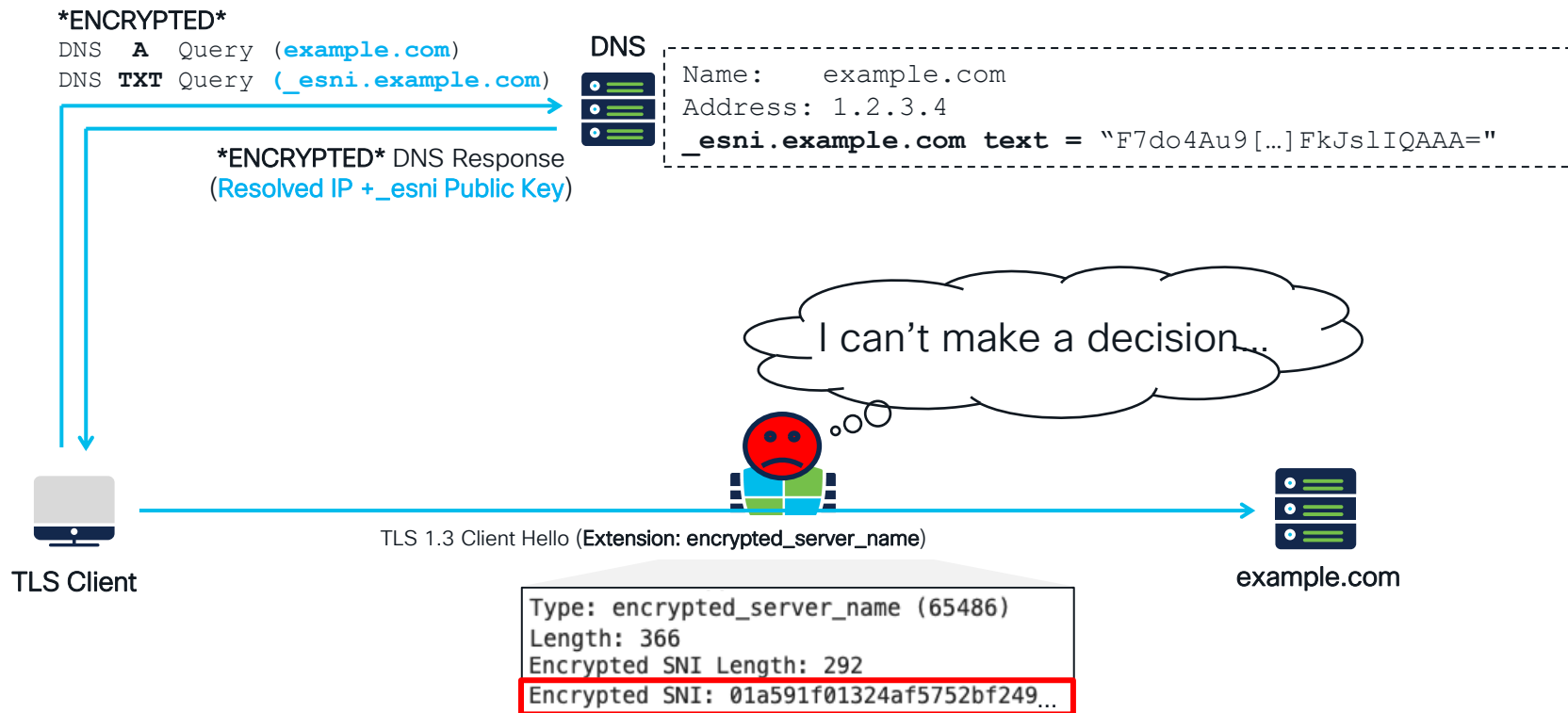
TALOS provides an **Application Detector** matching DNS over HTTPs traffic.

Encrypted SNI (ESNI) – a “Dodo” Protocol

- An **experimental feature** available in Firefox up to release 84.0
- Cloudflare used to provide an ESNI test page
- **Never reached** an RFC Proposed Standard
- Evolved into **Encrypted Client Hello (ECH)**



Encrypted SNI (ESNI)



Blocking ESNI Requests

Home Decryption Policy

Enter Description

Rules Trusted CA Certificates Undecryptable Actions **Advanced Settings**

Applies to 7.1.0 and later

☒ Block flows requesting ESNI

☐ Disable HTTP/3 advertisement

☐ Propagate untrusted server certificates to clients

Applies to 7.2.0 and later

☐ Enable TLS 1.3 Decryption

Applies to 7.3.0 and later

☐ Enable adaptive TLS server identity probe

Advanced options are available only with Snort 3

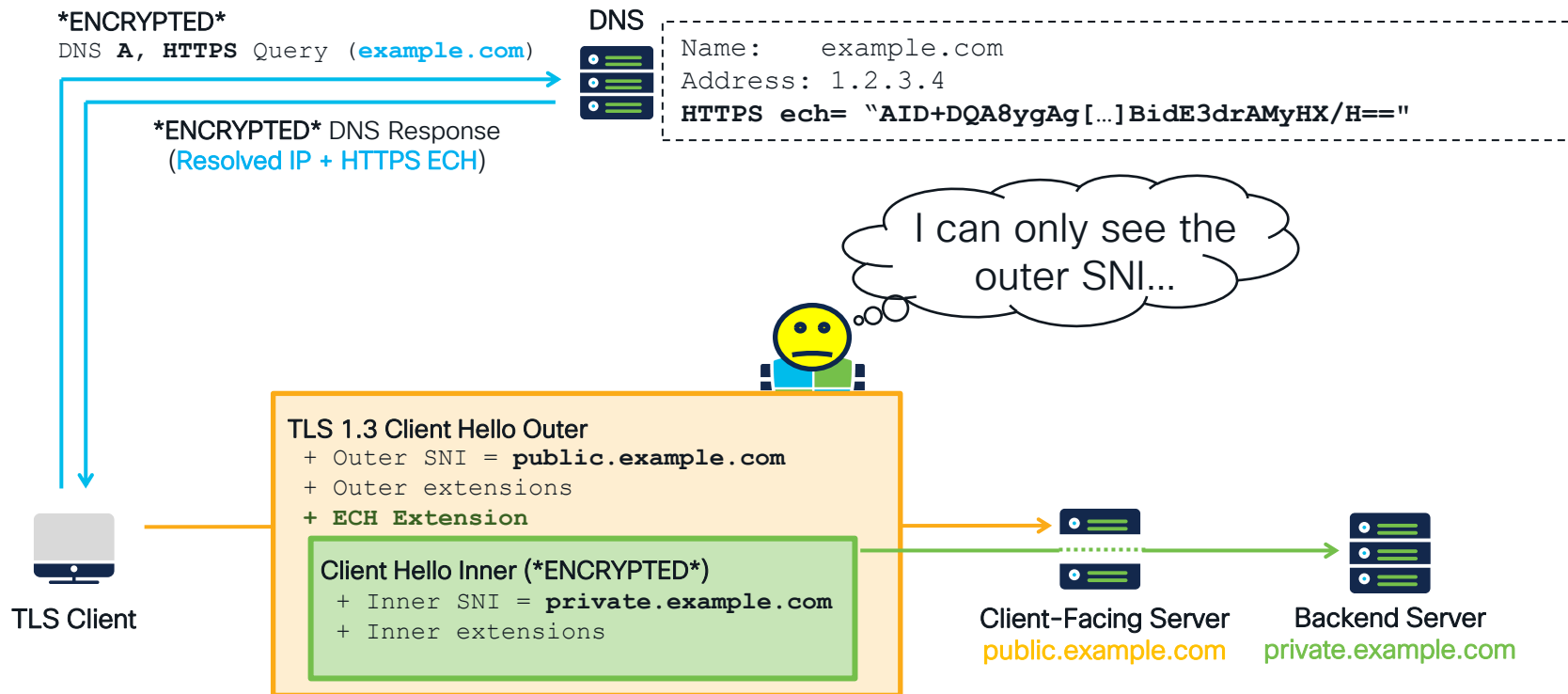
[Revert to Defaults](#)

Select this option to block connections with Client Hello **containing ESNI extension**.

Encrypted Client Hello

- ECH is an IETF Draft version 17 considered fairly stable
- **Minor footprint currently** – less than 1% of flows in Cisco EVE's dataset
- Wider adoption of ECH will make TLS decryption process even more involved
- Today, Cisco Secure Firewall **ignores** ECH

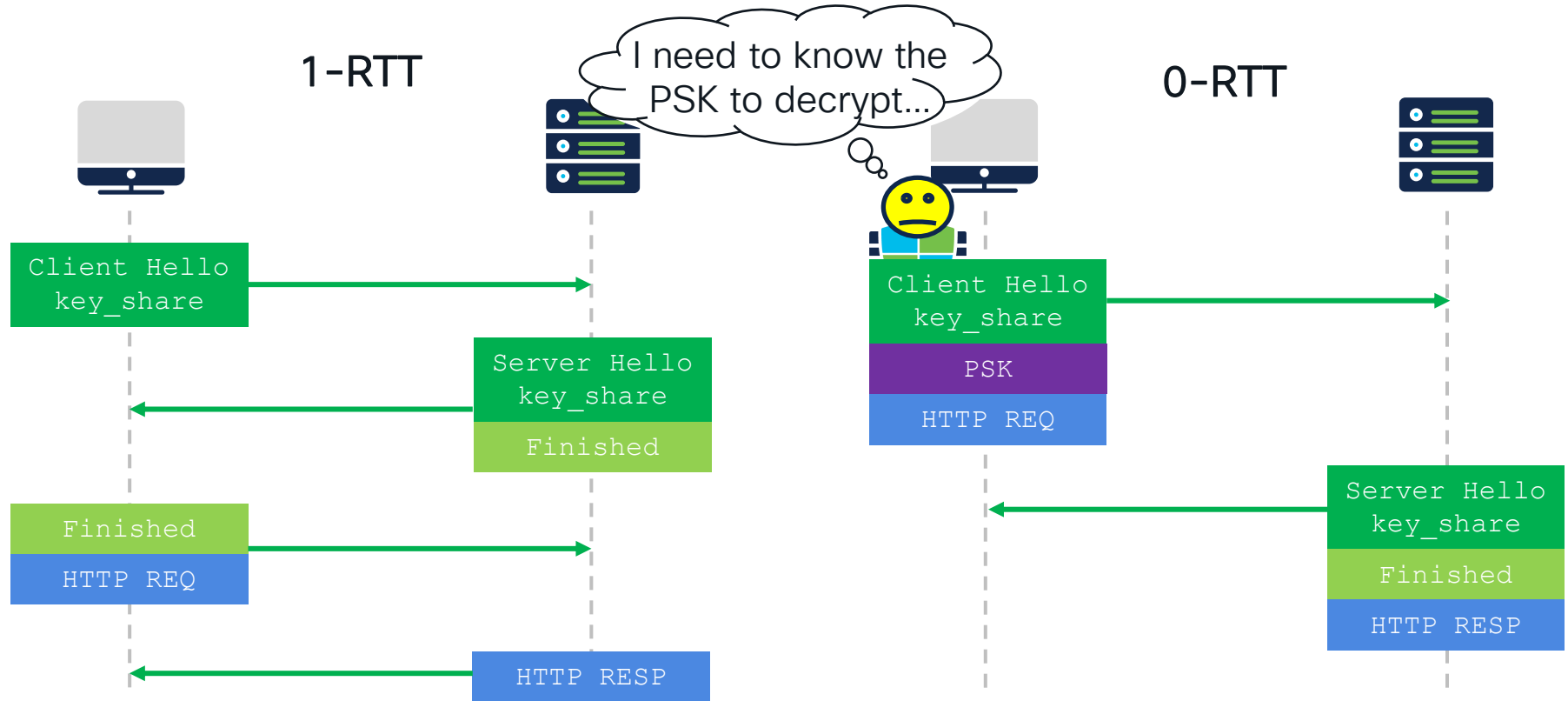
Encrypted Client Hello (ECH)



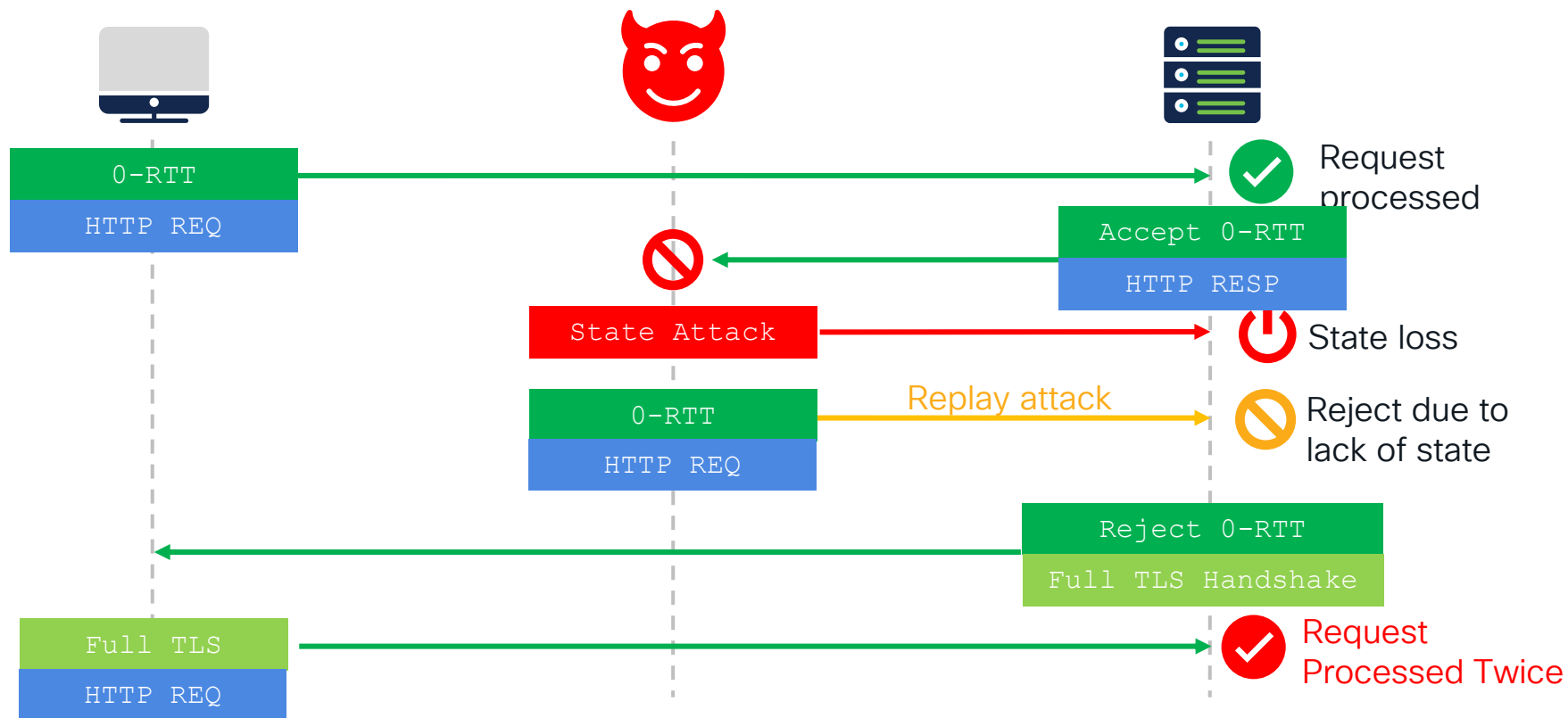
0-RTT

- 0-RTT is a technique baked into TLS 1.3/QUIC that **accelerates application response time**
- It requires the client to **possess a pre-shared key prior to the connection**
- Cisco Secure Firewall **strips 0-RTT flag from Client Hello** (if decrypting the flow)

0-RTT Saves One Round Trip on Reconnect (TLS 1.3)




...but there is no free lunch with 0-RTT



Performance Considerations

We Are Proud of Ours TLS Numbers!

Features	3110	3120	3130	3140
Throughput: FW + AVC + IPS (1024B)	17 Gbps	21 Gbps	38 Gbps	45 Gbps
TLS ¹	4.8 Gbps	6.7 Gbps	9.1 Gbps	11.5 Gbps

Features	4215	4225	4245 
Throughput: FW + AVC + IPS (1024B)	65 Gbps	80 Gbps	140 Gbps
TLS (Hardware Decryption) ¹	20 Gbps	30 Gbps	45 Gbps

¹ – measured with 50% TLS 1.2 with AES256-SHA with RSA 2048B keys



Check my
datasheet!

Plan Your Deployment with Performance Estimator

<https://ngfwpe.cisco.com> (Partner Access Required)

Specify the deployment mode and the required throughput.

Choose security features AVC, URL, AMP, IPS and Snort version.

The screenshot shows the Cisco Performance Estimator tool interface. Red dashed boxes and arrows highlight the following areas:

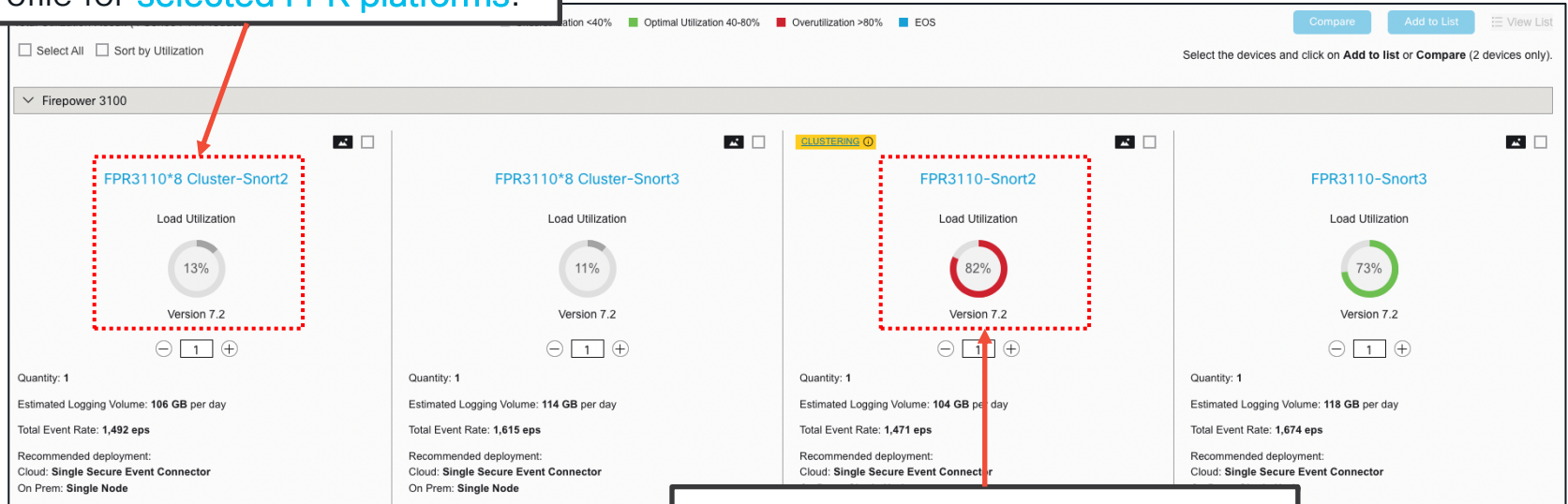
- Throughput:** A red dashed box around the 'Throughput' section, with an arrow pointing to it from the 'Specify the deployment mode and the required throughput' callout. It includes radio buttons for 'Inline Pairs' and 'Routed Mode', a dropdown for '2', and radio buttons for 'Mbps' and 'Gbps'.
- Network Profile:** A red dashed box around the 'Network Profile (Packet Size Mix)' section, with an arrow pointing to it from the 'Check the average packet length in the current network traffic' callout. It includes tabs for 'Default', 'Small', 'Datasheet', and 'Custom', and a dropdown for '733.50B Average Packet Size'.
- Enabled Features:** A red dashed box around the 'Enabled Features' section, with an arrow pointing to it from the 'Choose security features AVC, URL, AMP, IPS and Snort version' callout. It includes checkboxes for 'Base (AVC)', 'Threat (IPS)', 'Content (URL Filtering)', and 'Malware (AMP)', and a section for 'TLS Decryption and VPN IPsec' with sliders for 'TLS Decryption' (set to 50%), 'VPN IPsec' (set to 0%), and 'Clear Text' (set to 50%).
- Advanced Filters:** A red dashed box around the 'Advanced Filters - Operating Systems (Firepower Threat Defense)' section, with an arrow pointing to it from the 'Specify the percentage of encrypted traffic in the required throughput' callout. It includes dropdowns for 'Base Version' (set to 'Tested Version 7.2'), 'Compare Performance' (set to 'Select Tested Version'), 'Model Series' (set to 'Select Model Series'), 'Power Supply' (set to 'Select Power Supply'), 'High Availability' (set to 'Select High Availability'), 'Interfaces' (set to 'Select Interfaces'), and 'Operating Systems' (set to 'Select Operating Sys...').

Check the average packet length in the current network traffic.

Specify the percentage of encrypted traffic in the required throughput.

Performance Estimator – Results

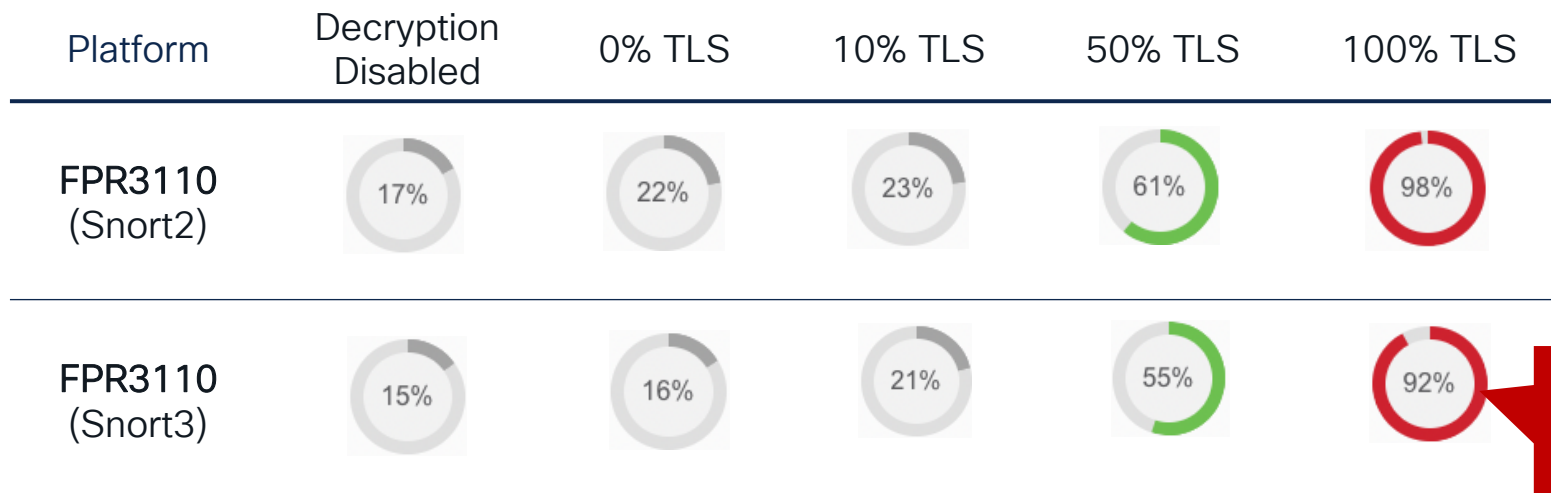
The output provides approximate **CPU utilization** for the chosen traffic profile for **selected FPR platforms**.



The **color of the graph** indicates anticipated under/over utilization of the appliance.

Firepower 3110 Snort2 vs. Snort3 – CPU Load

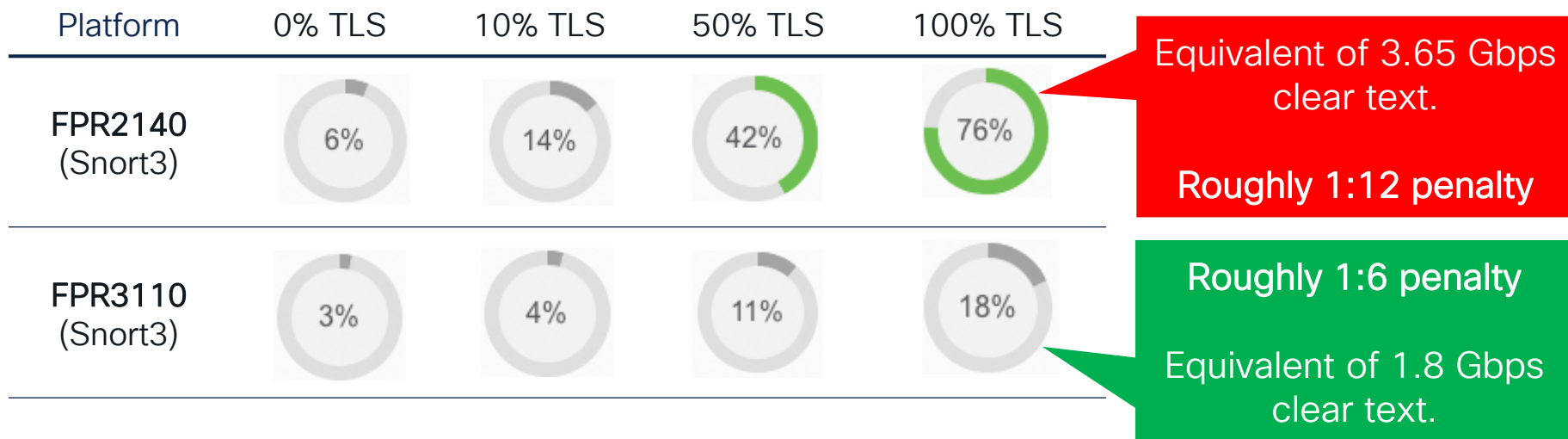
TRAFFIC PROFILE: 1.5Gbps with 733.5B packet running FW+AVC+IPS



Equivalent of
9 Gbps clear
text traffic

Firepower 2140 vs. 3110 - CPU Load

TRAFFIC PROFILE: 300 Mbps with 733.5B packet running FW+AVC+IPS



The background features a vibrant, multi-colored abstract design. On the left, there are horizontal, wavy bands of color in shades of red, orange, yellow, and green. On the right, a bright white light source emits a series of sharp, radiating lines in various colors, including blue, green, and yellow, creating a sunburst effect.

cisco *Live!*

Let's go