Cisco Webex App

Questions?
Use Cisco Webex App to chat with the speaker after the session

How
1. Find this session in the Cisco Live Mobile App
2. Click “Join the Discussion”
3. Install the Webex App or go directly to the Webex space
4. Enter messages/questions in the Webex space

Webex spaces will be moderated by the speaker until June 9, 2023.

https://ciscolive.ciscoevents.com/ciscolivebot/#BRKSEC-3320
Demystifying TLS Decryption and Encrypted Visibility Engine on Cisco Secure Firewall Threat Defence

Christopher Grabowski
Technical Marketing Engineer, Security Business Group &
George M Koikara
Principal Engineer, Security Business Group
BRKSEC-3320
Your TLS Speaker

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

Christopher Grabowski
Technical Marketing Engineer
CCIE Security #42466
What is an average parent tempted to do when seeing this at home?
TLS Encryption is Almost Everywhere...

Over 90% of Internet traffic is being encrypted with Transport Layer Security (TLS)

Percentage of pages loaded over HTTPs in Chrome by platform

Percentage of Alexa Top 1M redirecting to HTTPS

Sources:
- https://transparencyreport.google.com/https/overview
- https://scotthelme.co.uk/top-1-million-analysis-november-2021
... and So Are the Threats!!!

“76% OF CRITICAL AND HIGH-RISK THREATS detected by Cisco Secure Network Analytics were discovered in ENCRYPTED TRAFFIC”

Cisco Encrypted Traffic Analytics White Paper, Cisco Systems

"If attackers know that most defenders aren’t scanning encrypted traffic [...] it doesn’t take high volume for attacks to succeed.”

Internet Security Report Q3 2022, WatchGuard

"More than 85% of attacks now use encrypted channels across various stages of the kill chain (phishing, malware delivery, C&C activity, and more)”

The State of Encrypted Attacks 2022, ThreatLabz

“Of all respondents who were victims of a cyberattack, nearly half claimed the attack leveraged SSL traffic to evade detection. Another 15 percent were unsure.”

Uncovering Hidden Threats within Encrypted Traffic 2018, Ponemon Institute
Well, duh?!
Decrypt and inspect your traffic!!!
The TLS Decryption Paradox

TLS was designed to ensure malicious actors cannot see or alter what you transmit over the open network.

With TLS decryption, we are trying to break into a protocol specifically designed to protect against it...
Top Respondent Obstacles for TLS Decryption

- Lack of proper tools - hardware and software
- Concern of degradation of service
- Challenging technically to configure and operate
- Insufficient Resources and capacity
Agenda

- A TLS 1.3 Handshake Walkthrough
- TLS Decryption Under the Hood
- TLS Decryption Challenges
- Introduction to QUIC
- Challenges posed by QUIC
- EVE Overview
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 Session Flow – Client Hello

The TLS Client

My offer is cool!!! The server will accept for sure...

Why do we consider the TLS 1.3 handshake optimistic?
Understanding a TLS Session Flow – Client Hello

The client makes a guess of server’s preferences to accelerate the handshake...

... and blindly generates a public/private key pair.
Understanding a TLS Session Flow – Client Hello

**CLIENT HELLO**

- **Client Version** = 3.3 (TLS 1.2)
- **Session ID**: bbcc848ec..<RANDOM>
- **Cipher Suite (list)**:
  - TLS_AES_128_GCM_SHA256
  - TLS_AES_256_GCM_SHA384
  - TLS_CHACHA20_POLY1305_SHA256
  - ...
- **SNI**: www.cisco.com
- **Extension: Supported Versions** = TLS 1.2, TLS 1.3
- **Extension: Key Share**:
  - **Group**: x25519
  - **Length**: 32
  - **Key Exchange**: PUB

Knowing, client’s public key so **early in the handshake**, allows TLS 1.3 to send most of the **Server’s messages encrypted**.

The Client provides an **ordered list** of supported **cipher suites**. The higher on the list, the more preferable by the Client.

The **client’s ECDHE public key** is shared with the TLS server in the **first handshake** message.
Understanding a TLS Session Flow – TLS Version Negotiation

The Client Version is set to TLS 1.2.

- **Client HELLO**
  - Client Version = 3.3 (TLS 1.2)
  - Session ID: bbcc848ec...<RANDOM>
  - Cipher Suite (list) =
    - TLS_AES_128_GCM_SHA256
    - TLS_AES_256_GCM_SHA384
    - TLS_CHACHA20_POLY1305_SHA256
    - ...
  - SNI = www.cisco.com
  - Extension: Supported Versions = TLS 1.2, TLS 1.3
  - Extension: Key Share =
    - Group: x25519
    - Length: 32
    - Key Exchange:

**TLS 1.2?! ...is that correct?**
The Experimental Launch of TLS 1.3 Draft (back in 2017)

6% failure rate
(vs. TLS 1.2)

TLS 1.3 Draft 22 added “Middlebox Compatibility Mode”
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

The Client Version and Session ID fields are redundant from TLS 1.3 standpoint. We see them in the handshake to pass through ossified middlebox devices.

The Supported Versions extension is the only indication the client supports TLS 1.3.
Understanding a TLS Session Flow – Server Name Indication (SNI)

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

SSL Client

Client Hello

Client Version = 3.3 (TLS 1.2)
Session ID: bbcc848ec..<RANDOM>

Cipher Suite (list) =
TLS_AES_128_GCM_SHA256
TLS_AES_256_GCM_SHA384
TLS_CHACHA20_POLY1305_SHA256
...

SNI = www.cisco.com

Extension: Supported Versions =
TLS 1.2, TLS 1.3

Extension: Key Share =
Group: x25519
Length: 32
Key Exchange:

SSL Server

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

IP Address: 72.163.4.161

opendns.com
cisco.com
webex.com
Understanding a TLS Session Flow – Server Name Indication (SNI)

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

**CLIENT HELLO**
- Client Version = 3.3 (TLS 1.2)
- Session ID: bbcc848ec..<RANDOM>
- Cipher Suite (list) =
  - TLS_AES_128_GCM_SHA256
  - TLS_AES_256_GCM_SHA384
  - TLS_CHACHA20_POLY1305_SHA256
  - ...
- SNI = www.cisco.com
- Extension: Supported Versions = TLS 1.2, TLS 1.3
- Extension: Key Share =
  - Group: x25519
  - Length: 32
  - Key Exchange:

**SNI = www.cisco.com**
Understanding a TLS Session Flow – Server Name Indication (SNI)

Client Version = 3.3 (TLS 1.2)
Session ID: bbcc846ec...
Cipher Suite (list) =
  - TLS_AES_128_GCM_SHA256
  - TLS_AES_256_GCM_SHA384
  - TLS_CHACHA20_POLY1305_SHA256
  ...
SNI = www.cisco.com
Extension: Supported Versions = TLS 1.2, TLS 1.3
Extension: Key Share =
  Group: x25519
  Length: 32
  Key Exchange:

Why do we need the clear text Server Name Indication (SNI) extension?
Understanding a TLS Session Flow – Server Hello

The server continues the Middlebox Compatibility masquerade and echoes TLS 1.2 Client Version and Session ID back to the client.

The server indicates the negotiated TLS 1.3 version along with server selected Cipher Suite.
Understanding a TLS Session Flow – Server Hello

CLIENT HELLO

SERVER HELLO
Server Version = 3.3 (TLS 1.2)
Session ID: bbcc848ec..<Echoed from Client>
Cipher Suite = TLS_AES_256_GCM_SHA384
Extension: Supported Versions = TLS 1.3

If client’s offer is not acceptable, the Server corrects the mismatch and sends HelloRetryRequest instead of Server Hello – forcing the client to restart the handshake with appropriate Key Share.

If the client’s public key is acceptable, the Server generates its own public/private key pair and sends it to the client in Key Share extension.
Understanding a TLS Session Flow – Calculating the Shared Keying Material

The same keying material is now available on both the Client and the Server for bulk encryption.

Both ends use a hash of Client Hello and Server Hello messages in shared secret calculation.

Even if a single bit if changed by a MITM device, the calculation results in error and handshake fails in a downstream exchange.

I must be very cautious...
Understanding a TLS Session Flow – Encrypted Handshake

The server continues the Middlebox Compatibility masquerade and sends a dummy Change Cipher Spec.
Understanding a TLS Session Flow – Encrypted Handshake

It used to be much easier with TLS 1.2...

All messages afterward are fully encrypted. This includes server certificate, ALPN and other extensions.
Understanding a TLS Session Flow – Encrypted Extensions and Certificate

Server provides its own certificate (www.cisco.com) along with all certificates in the trust chain up to the root.

Server has to authenticate itself to client. It uses its certificate private key (www.cisco.com) to sign hash of the handshake messages.

Server may securely share additional extensions with the client. Application-Layer Protocol Negotiation (ALPN) being the most common.
Understanding a TLS Session Flow – Server Finished

The FINISHED message verifies:
• the handshake was successful,
• the keys work, and
• the handshake has not been tampered with.

The hash is built from the master encryption key and the payload of all handshake records previous to this one seen by the server.

Even if a single bit is changed by a MITM entity in any of the previous handshake messages, the hash will differ on the Client side. I must be very cautious…
Understanding a TLS Session Flow – Client Side Finish

The client side dummy Change Cipher Spec.

The Client FINISHED includes hash of master encryption key (lock) and all previous messages (including Server’s Handshake Finished).

If the Verify Data is not valid, on either side, the receiving peer must terminate the handshake with “decrypt_error” alert.

The client sends:
- CLIENT HELLO
- CLIENT CHANGE CIPHER SPEC
- CLIENT FINISHED

The server encrypts:
- SERVER HELLO, SERVER CHANGE CIPHER SPEC
- Encrypted SERVER ENCRYPTED EXTENSIONS, SERVER CERTIFICATE, CERTIFICATE VERIFY

Verify Data = cf919626f1360c536aaad73a
Understanding a TLS Session Flow – Client Side Finish

Agree on crypto algorithms and exchange the public DH

Verify the handshake

Encrypt application data

Encrypt application data

Calculate the shared key and switch to encryption

Share the server certificate and prove ownership of the private key
TLS 1.3 Decryption Under the Hood
# Decryption Policy – Rule Conditions

<table>
<thead>
<tr>
<th>Before TLS Handshake</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTD has the L2-L4 information about the flow</td>
</tr>
<tr>
<td>Interface Zones, Networks, Geolocation, VLAN Tags, User Identity, Protocol and Ports</td>
</tr>
<tr>
<td>TLS handshake inspection not required</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Low</th>
<th>Performance Impact / Visibility</th>
<th>High</th>
</tr>
</thead>
</table>
### Decryption Policy – Rule Conditions

<table>
<thead>
<tr>
<th>Before TLS Handshake</th>
<th>TLS Client Hello</th>
<th>TLS Server Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>FTD has the L2-L4 information about the flow</td>
<td>FTD has the additional information of SNI (Server Name Identification)</td>
<td>Server Certificate information and Server Hello</td>
</tr>
<tr>
<td>Interface Zones, Networks, Geolocation, VLAN Tags, User Identity, Protocol and Ports</td>
<td>Application, URL Category and Reputation</td>
<td>Certificate attributes, Ciphers, Versions, SNI mismatch</td>
</tr>
<tr>
<td>TLS handshake inspection not required</td>
<td>Initial match using SNI</td>
<td>Most reliable information</td>
</tr>
</tbody>
</table>

| Low | Performance Impact / Visibility | High |
TLS Session Decryption Flow - Client Hello

TLS Client

TLS 1.3 Client Hello (SNI = www.cisco.com)

Firewall

Shall I decrypt this connection? I have to make a decision now.

TLS Server
URL Detectors (SNI)

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.

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

- **Matches:**
  - www.acme.com
  - secure.acme.com

- **Does not match:**
  - www.sub.acme.com
  - top.svc.acme.com

Configure your own Distinguished Names objects to match traffic.
You can use Talos provided Application Detectors.

Decryption Policy supports SNI based Application Detectors only, hence the lower number comparing to Access Control Policy.
TLS Session Decryption Flow - Client Hello

TLS Client

TLS 1.3 Client Hello (SNI = www.cisco.com)

Firewall

Certificate for the SNI in cache?

Server Certificate Cache

Not cached

Server Certificate

Cache

Evaluate policy with Server Certificate

SSL Policy Rules

Decision

TLS Server

TLS 1.2 Probe Server Hello & Certificate

Install Server Certificate

TLS 1.2 Probe Client Hello (SNI)

© 2023 Cisco and/or its affiliates. All rights reserved. Cisco Public
TLS Session Decryption Flow – Client Hello

TLS Client

TLS 1.3 Client Hello (SNI = www.cisco.com)

Firewall

Certificate for the SNI in cache?

Certificate

Server Certificate Cache

Evaluate policy with Server Certificate

SSL Policy Rules

Decision

TLS Server
The decision is to decrypt this connection.

I need to remove unsupported ciphers and options...

**BRKSEC-3320**

Handshake with the Client
Firewall = Server

Handshake with the Server
Firewall = Client

* - modified message
Under the hood: CH Processing – Modify

Generate a new Random and zeroize Session ID.

Remove unsupported Cipher Suites.

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

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

session id [0]: <ZEROIZED>
cipher_suites len[20]: fafa 0113 0213 0313 2bc0 2fc0 2cc0 30c0 a9cc a8cc 13c0 14c0 9c00 9d00 2f00 3500

session id [0]: <ZEROIZED>
cipher_suites len[20]: fafa 0113 0213 0313 2bc0 2fc0 2cc0 30c0 a9cc a8cc 13c0 14c0 9c00 9d00 2f00 3500

session id [0]: <ZEROIZED>
cipher_suites len[20]: fafa 0113 0213 0313 2bc0 2fc0 2cc0 30c0 a9cc a8cc 13c0 14c0 9c00 9d00 2f00 3500

session id [0]: <ZEROIZED>
cipher_suites len[20]: fafa 0113 0213 0313 2bc0 2fc0 2cc0 30c0 a9cc a8cc 13c0 14c0 9c00 9d00 2f00 3500

session id [0]: <ZEROIZED>
cipher_suites len[20]: fafa 0113 0213 0313 2bc0 2fc0 2cc0 30c0 a9cc a8cc 13c0 14c0 9c00 9d00 2f00 3500

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]
I’ll check the policy again and make the **final verdict**.
Block Weak Ciphers and TLS/SSL Versions

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

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

At this stage we can match with a specific server Certificate.
TLS Session Decryption Flow – Certificate Check

TLS Client

TLS 1.3 Client Hello (SNI = www.cisco.com)

Firewall

TLS 1.3 Client Hello* (SNI = www.cisco.com)

TLS Server

TLS 1.3 Server Hello

TLSServer

TLS 1.3 Server Finish

Issuer:
Common Name: HydrantID Server CA 01
Organization: IdenTrust

Certificate Subject Alternative Names:
DNS Name: cisco-images.cisco.com
DNS Name: cisco.com
DNS Name: www-01.cisco.com
DNS Name: www-02.cisco.com
DNS Name: www-rtp.cisco.com
DNS Name: www.cisco.com
DNS Name: www.mediafiles-cisco.com
DNS Name: www.static-cisco.com
DNS Name: www1.cisco.com
DNS Name: www2.cisco.com
DNS Name: www3.cisco.com

* - modified message

Do I trust the issuer of this certificate?
Trusted CA Certificates

The firewall comes with a predefined set of Trusted CAs.

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

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.
TLS Session Decryption Flow – CN/SAN and SNI Crosscheck

SNI = www.cisco.com

Does the SNI match the SAN/CN?

Certificate Subject Alternative Name (SAN):
- DNS Name: cisco-images.cisco.com
- DNS Name: cisco.com
- DNS Name: www-01.cisco.com
- DNS Name: www-02.cisco.com
- DNS Name: www-rtp.cisco.com
- DNS Name: www.cisco.com
- DNS Name: www.mediafiles-cisco.com
- DNS Name: www.static-cisco.com
- DNS Name: www1.cisco.com
- DNS Name: www2.cisco.com
- DNS Name: www3.cisco.com
TLS Session Decryption Flow – Late do not decrypt

I made a **wrong decision** and modified Client Hello...
The **Verdict** is **DoNotDecrypt**.

---

**TLS 1.3 Client Hello** (SNI = **www.cisco.com**)

**Firewall**

**TLS 1.3 Server Hello**

**Cache & Trash**

**TLS 1.3 Encrypted Certificate**

**TLS 1.3 Certificate Verify**

**TLS 1.3 Server Finish**

---

* - modified message
## Block Server SNI Mismatch

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

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.
I don’t know the private key of cisco.com. I’ll spoof the certificate.

Subject:
Common Name: www.cisco.com
Organization: Cisco Systems Inc.

Issuer:
Common Name: SSL-Decrypt-SubCA.lab.local

Certificate Subject Alternative Name:
DNS Name: cisco-images.cisco.com
DNS Name: cisco.com
DNS Name: www-01.cisco.com
DNS Name: www-02.cisco.com
DNS Name: www-rtp.cisco.com
DNS Name: www.cisco.com
DNS Name: www.mediafiles-cisco.com
DNS Name: www.static-cisco.com
DNS Name: www1.cisco.com
DNS Name: www2.cisco.com
DNS Name: www3.cisco.com

* - modified message
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.
TLS Session Decryption Flow – Client Check

Do I trust the issuer of this certificate?

ISSUER:
Common Name: SSL-Decrypt-SubCA.lab.local

CERTIFICATE
Subject Alternative Name:
DNS Name: cisco-images.cisco.com
DNS Name: cisco.com
DNS Name: www-01.cisco.com
DNS Name: www.rtp.cisco.com
DNS Name: www.cisco.com
DNS Name: www.mediasfiles-cisco.com
DNS Name: www.static-cisco.com
DNS Name: www1.cisco.com
DNS Name: www2.cisco.com
DNS Name: www3.cisco.com

* - modified message
Screenshot of resigned certificate

Resigned Certificate

Field: Version
Value: V3

Field: Serial number
Value: 123bee63ff3e1322ed59300

Field: Signature algorithm
Value: sha256RSA

Field: Signature hash algorithm
Value: sha236

Field: Issuer
Value: SSL-Depliant-SuCA.labs.local, l...

Field: Valid from
Value: Wednesday, February 16, 2023...

Field: Valid to
Value: Thursday, February 16, 2023...

Field: Subject
Value: US, California, San Jose, Cisco...

Field: Public key
Value: RSA (2048 bits)

Field: Public key parameters
Value: 05 00

Field: Subject Alternative Name
Value: DNS Name=820-w5m65g56c66s7665c64656f77

Field: Subject Key Identifier
Value: b1e3ec0d9a5f6d0a550c650f76

Field: Enhanced Key Usage
Value: Server Authentication (1.3.6.1.4.1.3194)

Field: SCT List
Value: V3, adf3fe57c7c593c93f6f84b3f3f09

Field: Key Usage
Value: Digital Signature, Key Encipher...

Field: Thumbprint
Value: b6f5144e3c93f6f84b3f3f09

Original Certificate

Field: Version
Value: V3

Field: Serial number
Value: 4b0017f9f44e5f23133d85f82d6...

Field: Signature algorithm
Value: sha256RSA

Field: Signature hash algorithm
Value: sha236

Field: Issuer
Value: Hyphen4ID Server CA O1, Hyd...

Field: Valid from
Value: Wednesday, February 16, 2023...

Field: Valid to
Value: Thursday, February 16, 2023...

Field: Subject
Value: US, California, San Jose, Cisco...

Field: Public key
Value: RSA (2048 bits)

Field: Public key parameters
Value: 05 00

Field: Authority Information Access
Value: [Authority Info Access - Access points]

Field: Authority Key Identifier
Value: KeyID=0000000000000000000000000000000000000000

Field: Certificate Policies
Value: [Cisco Policy Policy ID 0]

Field: CRL Distribution Points
Value: [CRL Distribution Points Distribution Points]

Field: Subject Alternative Name
Value: DNS Name=820-w5m65g56c66s7665c64656f77

Field: Subject Key Identifier
Value: b1e3ec0d9a5f6d0a550c650f76

Field: Enhanced Key Usage
Value: Server Authentication (1.3.6...
TLS Session Decryption Flow – PIG-in-the-Middle

- **TLS Client**
- **Firewall**
- **TLS Server**

**Client Side TLS**

**Server Side TLS**

**AVC, NGIPS, File Policy**
TLS Decryption Challenges
It Is Not an Easy World for a Man (-in-the-Middle)

- TLS 1.3
- Certificate Pinning
- DNS over HTTPs
- Encrypted Client Hello
- Encrypted SNI
- 0-RTT
- QUIC
TLS 1.2 vs 1.3 Handshake

TLS 1.2

- Client Hello (SNI)
- Server Hello
- Certificate
- Server Key Exchange
- Other Handshake Messages
- Encrypted
- Finished
- Application Data

TLS 1.3

- Client Hello (SNI, key_share)
- Server Hello (key_share)
- Encrypted
- Certificate
- Other Server Handshake
- Finished
- Finished
- Application Data
TLS Server Cache & Probing

TLS Client

- TLS 1.3 Client Hello (SNI, key_share)

Firewall

1. Certificate for the SNI in cache?
   - Not cached
2. TLS 1.2 Probe Client Hello (SNI)
3. Install Server Certificate
4. Evaluate policy with Server Certificate
5. Decryption Verdict

TLS Server

- TLS 1.3 Client Hello* (SNI, key_share)

- TLS 1.2 Probe Server Hello & Certificate
- Server Certificate Cache
- SSL Policy Rules

**TLS 1.3 handshake continues to completion and the session is established**

* - modified message
What is Certificate Pinning?

- Pin to Root
  - Root CA
    - CN=IdenTrust Comm Root CA 1

- Pin to Intermediate
  - Intermediate CA
    - CN=HydrantID Server CA 01

- Pin to Leaf
  - Leaf
    - CN=www.cisco.com
  - Leaf
    - CN=www.webex.com
Why is Certificate Pinning a Problem?

The service is unavailable is signed by a system trusted Root CA...

...but I have this one hardcoded:

---

Firewall

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

- TLS 1.2 Server Hello*
- TLS 1.2 Server Certificate*
- TLS 1.2 Server Key Exchange*
- TLS 1.2 Server Hello Done*

TLS ALERT (optional) + TCP RST

---

TLS Server

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

- TLS 1.2 Server Hello
- TLS 1.2 Server Certificate
- TLS 1.2 Server Key Exchange
- TLS 1.2 Server Hello Done

* - modified message
Troubleshooting Certificate Pinning - Capture

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

You can use this tag to match application traffic that should bypass decryption.

Pinned Certificate applications tag is available in the SSL Policy.
Cisco Provided Undecryptable Sites (1/2)

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

You can also use a pre-defined DN Object containing know Undecryptable Sites...
Cisco Provided Undecryptable Sites (2/2)

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: 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.trustnet - Value: 
Name: CN_logmein - Value: 
Name: CN_latinum.amazon.com - Value: 
Name: CN_data.microsoft.com - Value: 
Name: CN_redhat.com - Value: 
Name: CN_icloud.com - Value: 

There are 51 more items in this group...
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

If the connection is certificate pinned, I can’t look inside...
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
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)

I can’t make a decision...

TLS 1.3 Client Hello (Extension: encrypted_server_name)

Type: encrypted_server_name (65486)
Length: 366
Encrypted SNI Length: 292
Encrypted SNI: 01a591f01324af5752bf249...

DNS A Query (example.com)
DNS TXT Query (_esni.example.com)

*ENCRYPTED* DNS Response (Resolved IP + _esni Public Key)

Name: example.com
Address: 1.2.3.4
_esni.example.com text = "F7do4Au9[…]FkJslIQAAA="

TLS Client
Blocking ESNI Requests

Select this option to block connections with Client Hello containing ENSI extension.
Encrypted Client Hello

- ECH is an IETF Draft version 16 considered fairly stable
- **Minor footprint currently** – less than 0.2% 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)

**TLS 1.3 Client Hello Outer**
- Outer SNI = public.example.com
- Outer extensions

**+ Client Hello Inner (ENCRIPTED)***
- Inner SNI = private.example.com
- Inner extensions

**DNS**
- Name: example.com
- Address: 1.2.3.4
- HTTPS ech = "AID+DQA8ygAg[…]BidE3drAMyHX/H=="

**I can only see the outer SNI...**

---

© 2023 Cisco and/or its affiliates. All rights reserved. Cisco Public
You can block DNS over HTTPs with a rule in your Access Control Policy.

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

1-RTT

Client Hello key_share

Server Hello key_share

Finished

HTTP REQ

HTTP RESP

0-RTT

Client Hello key_share

Server Hello key_share

Finished

HTTP REQ

HTTP RESP

I need to know the PSK to decrypt...
Now let’s do a QUIC change of speakers...
Your QUIC Speaker

George Koikara
Principal Engineer

Based in Bangalore, India

With Cisco since June 2019

Architect for QUIC and ZTNA for Secure Firewall

Hold multiple patents in domain of security
A brief history

• IETF proposed standard (RFCs 9000–9003) – started at Google in 2014

• A new secure transport protocol with baked-in encryption

• Used by ~ 14% of the websites and over 35% of Google’s traffic

• Over ~70% of Facebook traffic is on QUIC

• Provides significant improvements for application and content providers:
  • Youtube Video rebuffers reduced by up to 18%
  • Google Search latency reduction up to 8%
Pre Poll

Join at slido.com
#1546 305
Passcode: clusquic
Why QUIC ?
“Change is the essential process of all existence.”

- Mr. Spock, "Let That Be Your Last Battlefield"
Head of Line Blocking..  NOW PLAYING

PROGNOSIS NEGATIVE
Head of Line Blocking HTTP/1.1

www.ciscolive.com
### Head of Line Blocking HTTP/1.1

The image contains a table showing web page component details, including status, method, domain, file, initiator, type, transferred size, and size. The table is presented in a natural text format with appropriate alignment and structure.
Head of Line Blocking HTTP/1.1

GET /c/cdcrSwitch.js

var revisionsFix = document.createElement('style');
revisionsFix.innerHTML = `.cdc-eot-toc .docId .docHeaderComponent .infobar .seeRevisions { display:none; }`

document.head.appendChild(revisionsFix);
Head of Line Blocking HTTP/1.1

GET /c/cdcrSwitch.js

```javascript
var revisionsFix = document.createElement('style');
revisionsFix.innerHTML = '.cdc-eot-toc .docld .docHeaderComponent .infobar .seeRevisions { display:none; }';
document.head.appendChild(revisionsFix);
```

HTTP 1.1 200 OK
Content-Length : 2000

```javascript
var revisionsFix = document.createElement('style');
revisionsFix.innerHTML = '.cdc-eot-toc .docld .docHeaderComponent .infobar .seeRevisions { display:none; }';
document.head.appendChild(revisionsFix);
```
Head of Line Blocking HTTP/1.1

GET /c/cdcrSwitch.js

Browser

/c/cdcrSwitch.js

```javascript
var revisionsFix = document.createElement('style');
revisionsFix.innerHTML = 'cdc-eot-toc.docld .docHeaderComponent .infobar .seeRevisions { display:none; }';
document.head.appendChild(revisionsFix);
```

HTTP 1.1 200 OK
Content-Length: 2000

TCP PKT 1
HTTP 1.1 200 OK
Content-Length: 2000

TCP PKT 2

```javascript
var revisionsFix = document.createElement('style');
revisionsFix.innerHTML = 'cdc-eot-toc.docld .docHeaderComponent .infobar .seeRevisions { display:none; }';
document.head.appendChild(revisionsFix);
```
Head of Line Blocking HTTP/1.1

GET /c/cdcrSwitch.js  GET /c/cisco.png

```
var revisionsFix = document.createElement('style');
revisionsFix.innerHTML = '//cdcr-switch cdcd .docHeaderComponent
 .infoBar .seeRevisions { display:none; }
document.head.appendChild(revisionsFix);
```

Browser
Head of Line Blocking HTTP/1.1

GET /c/cdcrSwitch.js  GET /c/cisco.png

Browser

TCP PKT 1  HTTP 1.1 200 OK  Content-Length: 2000

TCP PKT 2

TCP PKT 3  HTTP 1.1 200 OK  Content-Length: 800

Cannot be multiplexed
Head of Line Blocking HTTP/1.1

- Work around:
  - Browsers open multiple TCP connections, typically 6
  - Shard content over multiple domains for load distribution
- Drawback
  - TCP setup overhead
  - 3-way handshake
  - TLS setup overhead for HTTPS connection
The TLS angle

- HTTP client/server puts the data to be transmitted to the TLS layer which encrypts it in entirety.
- This huge blob of encrypted data is now split over multiple TCP packets.
The TLS angle

- HTTP client/server puts the data to be transmitted to the TLS layer which encrypts it in entirety.

- This huge blob of encrypted data is now split over multiple TCP packets.
The TLS angle

- HTTP client/server puts the data to be transmitted to the TLS layer which encrypts it in entirety.
- This huge blob of encrypted data is now split over multiple TCP packets.
The TLS angle

- HTTP client/server puts the data to be transmitted to the TLS layer which encrypts it in entirety.
- This huge blob of encrypted data is now split over multiple TCP packets.
The TLS angle

- HTTP client/server puts the data to be transmitted to the TLS layer which encrypts it in entirety.
- This huge blob of encrypted data is now split over multiple TCP packets.

```
var revisionsFix = document.createElement('style');
revisionsFix.innerHTML = `.cdc-eot-toc .docHeaderComponent
  .infobar .seeRevisions { display:none; }`
; document.head.appendChild(revisionsFix);

document.addEventListener(DOMContentLoaded, function() { 
```

```
HTTP 1.1 200 OK
Content-Length: 2000
```

```
var revisionsFix = document.createElement('style');
revisionsFix.innerHTML = `.cdc-eot-toc .docHeaderComponent
  .infobar .seeRevisions { display:none; }`
; document.head.appendChild(revisionsFix);

document.addEventListener(DOMContentLoaded, function() {
```

```
Encrypted Data
```

```
TCP PKT 1  TLS Record Part 1  Encrypted Data  TCP PKT 2  TLS Record Part 2  Encrypted Data
```

```
TLS Encrypt()```

```
```
The TLS angle (contd..)

- On the receiving side, all these packets have to be received by the TLS layer so that it can form the full record for decryption.

- In case of transmission errors, retransmits at the TCP layer, the TLS layer has to wait to get all packets.
The TLS angle (contd..)

- On the receiving side, all these packets have to be received by the TLS layer so that it can form the full record for decryption.

- In case of transmission errors, retransmits at the TCP layer, the TLS layer has to wait to get all packets.
The TLS angle (contd..)

- On the receiving side, all these packets have to be received by the TLS layer so that it can form the full record for decryption.

- In case of transmission errors, retransmits at the TCP layer, the TLS layer has to wait to get all packets.
The TLS angle (contd..)

- On the receiving side, all these packets have to be received by the TLS layer so that it can form the full record for decryption.

- In case of transmission errors, retransmits at the TCP layer, the TLS layer has to wait to get all packets.
The TLS angle (contd..)

- On the receiving side, all these packets have to be received by the TLS layer so that it can form the full record for decryption.

- In case of transmission errors, retransmits at the TCP layer, the TLS layer has to wait to get all packets.
Problem Summary

HTTP 1.1
• Data sent before blocks data sent later
• Cannot be multiplexed

TLS
• Transmission errors cause delayed decrypt
• Setup times are huge
ADVENT OF HTTP/2
HTTP/2

• Stream based protocol
• Multiple HTTP content Streams

• Truly multiplexed
• One TCP connection
# HTTP/2

- Stream based protocol
- Multiple HTTP content Streams
- Truly multiplexed
- One TCP connection

<table>
<thead>
<tr>
<th>TCP PKT 1</th>
<th>HEADERS: Stream ID:1</th>
<th>status 200</th>
<th>DATA: Stream ID:1</th>
<th>status 200</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Content-Length : 100</td>
<td></td>
<td>Content-Length : 100</td>
<td></td>
</tr>
</tbody>
</table>

```
var revisionsFix = document.createElement('style ');
revisionsFix.innerHTML = ' .cdc-eot-toc .docid
.<br />
```

<table>
<thead>
<tr>
<th>TCP PKT 2</th>
<th>Data: Stream ID:2</th>
<th>status 200</th>
<th>DATA: Stream ID:2</th>
<th>status 200</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Content-Length : 50</td>
<td></td>
<td>Content-Length : 500</td>
<td></td>
</tr>
</tbody>
</table>

```
if (typeof ctm === "undefined")
ctm = {};
//Function encapsulates all GDPR related features and attributes
```

<table>
<thead>
<tr>
<th>TCP PKT 3</th>
<th>Data: Stream ID:1 Length:600</th>
<th>document.addEventListener('DOMContentLoaded', (event) =&gt; {</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```
//Event Queue for call backs if not defined
var eventqueue = [];
//Optanon Wrapper Function Counter
```

© 2023 Cisco and/or its affiliates. All rights reserved. Cisco Public
Problem Recap

HTTP 1.1

• Data sent earlier blocks data sent later
• Cannot be multiplexed

TLS

• Transmission errors cause delayed decrypt
• Setup times are huge

SOLVED

NOT SOLVED
A Not so New Problem .... Yet Again

<table>
<thead>
<tr>
<th>TCP PKT 1</th>
<th>HEADERS: Stream ID:1</th>
<th>DATA:</th>
<th>status 200</th>
<th>Content-Length: 100</th>
<th>status 200</th>
<th>Content-Length: 900</th>
<th>var revisionsFix = document.createElement('style') + revisionsFix.innerHTML = <code>;cdcced-to decrement...</code></th>
</tr>
</thead>
</table>

- TCP guarantees in order packet delivery
- Assume packet ‘2’ lost
- Packet ‘1’ delivered to application
- Packet ‘3’ buffered
  - Until packet ‘2’ retransmitted and received successfully
- Packet ‘3’ may be containing a different stream, but blocked by lost packet ‘2’
- TCP is unaware of payload data
- Remember: TLS problem still exists

Packet Lost
A Not so New Problem .... Yet Again

- TCP guarantees in order packet delivery
- Assume packet ‘2’ lost
- Packet ‘1’ delivered to application
- Packet ‘3’ buffered
  - Until packet ‘2’ retransmitted and received successfully
- Packet ‘3’ may be containing a different stream, but blocked by lost packet ‘2’
- TCP is unaware of payload data
- Remember: TLS problem still exists
HTTP/3 over QUIC PROTOCOL
Need for HTTP/3

HTTP/1.1

- Multiplexed streams
- Reduced connection times
- Increased Privacy
- Eliminate TCP head of line blocking

HTTP/2
HTTP/3

- Removal of streams at Application layer
- Simple compared to HTTP/2
- In some ways similar to HTTP/1
  - Multiple connections are instead carried over multiple streams on the same connection.
- Works only on QUIC
  - Streaming protocol → streams from HTTP/2 moved down to transport layer
Quick UDP Internet Connections – QUIC

- Generic for all kinds of traffic
- NOT ONLY HTTP
- Always encrypted
- No opt-in, no opt-out
TCP vs QUIC

<table>
<thead>
<tr>
<th>TCP</th>
<th>Encrypted</th>
</tr>
</thead>
<tbody>
<tr>
<td>Src Port</td>
<td></td>
</tr>
<tr>
<td>Dest Port</td>
<td></td>
</tr>
<tr>
<td>Seq No</td>
<td></td>
</tr>
<tr>
<td>ACK No</td>
<td></td>
</tr>
<tr>
<td>Flags</td>
<td></td>
</tr>
<tr>
<td>Windows</td>
<td></td>
</tr>
<tr>
<td>Options</td>
<td>Payload</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>UDP</th>
<th>QUIC (open)</th>
<th>QUIC (encrypted)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Src Port</td>
<td>Flags</td>
<td>Packet No</td>
</tr>
<tr>
<td>Dest Port</td>
<td>Connection ID</td>
<td>Frame</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ACK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Window</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Options</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Payload</td>
</tr>
</tbody>
</table>
Where Does QUIC Fit in the Protocol Stack?

- **H/2**
  - Header Compression, Server Push
  - Prioritization
  - Stream Multiplexing

- **TLS**
  - Authentication
  - Key Negotiation
  - Session Resumption
  - Encryption & Decryption

- **TCP**
  - Reliability
  - Congestion Control
  - Port Numbers

- **H/3**
  - Header Compression, Server Push
  - Prioritization

- **QUIC**
  - Stream Multiplexing
  - Authentication
  - Key Negotiation
  - Session Resumption
  - Encryption & Decryption
  - Reliability
  - Congestion Control

- **UDP**
  - Port Numbers
QUIC Connection Setup

Client

QUIC Initial + Client Hello

QUIC Initial + Server Hello

+ QUIC Handshake + Certs

QUIC Handshake Finished

Server

1-RTT Encrypted Data
Connection Comparison

HTTP over TCP+TLS

- SYN
- SYN + ACK
- ACK
- TLS CLIENT HELLO
- TLS SERVER HELLO
- TLS FIN
- HTTP REQUEST
- HTTP RESPONSE

HTTP over QUIC

- QUIC
- QUIC
- QUIC
- HTTP REQUEST
- HTTP RESPONSE
Connection Identifiers

Client

Src: CID a,
Dst: CID dummy, CH

Local: a, b, c, d, ..., h
Remote: x, p, q, r, s, t, u

DCID: p

HS Complete

Encrypted Channel

Src: CID x,
Dst: CID a, SH

Local: x, p, q, r, s, t, u
Remote: a, b, c, d, ..., h

New CIDs: CID b, CID c, ..., CID h

NEW CIDs: CID p, CID q, ..., CID u

Server

Long Headers

Remote:

Short Headers

DCID: b
QUIC Packets

• One UDP payload can have multiple QUIC packets
• Each packet has multiple frames
Connection Migration

- QUIC uses Connection Identifiers (CID) for identifying connections, not IP
- Any CID could be used to identify the connection
- The set of CIDs, except the first pair, are exchanged over encrypted channel
- IP of peers can change any time, CIDs help keep the connections alive
- For privacy, new CID to be used on IP migration
- Migration can happen from IPv4 to IPv6 and vice-versa
Connection Management with TCP

123.123.10.10

10.23.10.50
Connection Management with TCP

- New TCP + TLS Connection has to be established.
- Increases Latency
Connection Management with QUIC

CID $a, b, c, d$

123.123.10.10

CID $p, q, r, s, t, u$

10.23.10.50
Connection Management with QUIC

- Old connection continues
- No Key renegotiation
Man in the Middle

**CID:** \( p, q, r, s, t, u \)

**ISP:** 25.30.140.151

**192.168.2.20**

**152.123.7.90**

**ISP**
Man in the Middle

CIDs exchanged over encrypted channel make it difficult to trace connections
Other Properties of QUIC

- **Unique Packet numbers**: Even retransmitted packets have different packet numbers.
- **Connection Resumption**: Connecting to a server which negotiated a 0-RTT secret in the previous session.
- **Congestion Control at stream layer**: Leverages TLS 1.3 0-RTT.
- **Optimised ACKing**:
Performance Benefits

- Mileage varies
  - Implementation
  - Congestion control options used

- Low improvement in stable/reliable networks
- Moderate improvement in unstable networks

Challenges for Firewalls
Challenges for existing Security devices

<table>
<thead>
<tr>
<th>Challenge</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>FW is blind to QUIC traffic</td>
<td>Can at most block UDP/443</td>
</tr>
<tr>
<td></td>
<td>No mechanism to inspect encrypted traffic</td>
</tr>
<tr>
<td>Only option to block QUIC traffic</td>
<td>Recommended by almost all FW vendors</td>
</tr>
<tr>
<td>IP cannot be used to identify connections</td>
<td>Would be difficult to create ACLs and other transport level parameters</td>
</tr>
<tr>
<td></td>
<td>based rules</td>
</tr>
<tr>
<td></td>
<td>Much of the content would be in the streams</td>
</tr>
</tbody>
</table>
Challenges to adding QUIC support

- **QUIC is IP agnostic**
  - Firewalls use IP to track and maintain connection states. Changing IPs require Firewalls to decide if they should allow or block QUIC traffic.

- **Stream Based**
  - Reassembly of streams can be done only after decryption of packets.
Challenges to adding QUIC support

Connection ID Based
- One flow can have multiple Connection IDs.
- Source Connection ID is optional in data packets.
- Firewalls have to understand and manage Connection IDs

All packets Encrypted
- Encryption at packet level – 2 levels of decryption
- More packets to be decrypted.
- Unlike TLS where TLS records span multiple packets which needs lesser crypto invocations
- Compute intensive
Work based on QUIC

- DoQ – DNS over QUIC
- HTTP/3
- MS SMB
- RTP
- QUIC Load balancing
- Multiplexed Application Substrate over QUIC Encryption (MASQUE)
- QUIC Multi Path
- BGP
- SSH
- QUIC Multi Path

#CiscoLive
Follow up Poll

Join at
slido.com
#1546 305
Passcode: clusquic
QUIC Inspection in Cisco Secure Firewall
Coming soon

• Inspects content per stream
• Block individual streams
Encrypted Visibility Engine
TLS Decryption is Expensive

Over 90% of Internet traffic being encrypted with Transport Layer Security (TLS)

Mobile app and SaaS flows are not decryptable

TLS 1.3 encrypts Server Certificate Information

TLS decryption has a steep performance impact

Use techniques to make decisions on TLS traffic without decryption

Almost 63% of servers prefer TLS 1.3 to other protocols (Aug/2021)
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 (0x1304)
  - Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0x130B)
  - Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0x130A)
  - Cipher Suite: TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256 (0x15c9)
  - Cipher Suite: TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256 (0x15c8)
  - Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0x131a)
  - Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0x1319)
  - Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0x1306)
  - Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0x1305)
  - Cipher Suite: TLS_DHE_RSA_WITH_AES_256_CBC_SHA (0x1303)
  - Cipher Suite: TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x1309)
  - Cipher Suite: TLS_RSA_WITH_AES_256_GCM_SHA384 (0x0093)

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**

**TLS ClientHello**

- Cipher Suites (10 suites)
  - Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0x130c)
  - Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0x130b)
  - Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0x130a)
  - Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0x1309)
  - Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0x1306)
  - Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0x1305)
  - Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0x1304)
  - Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0x1303)
  - Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0x1302)
  - Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0x1301)

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

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

The **Destination Context** comprises of **IP and TCP/UDP ports**...

... as well as the destination server’s **domain name**.
Fingerprinting Analysis at the Firewall

TLS Client Hello

- Internet Protocol Version 4, Src: 10.55.62.51
- Transmission Control Protocol, Src Port: 59536, Dst Port: 443
- Transport Layer Security
  - TLSv1.2 Record Layer: Handshake Protocol: Client Hello
    - Content Type: Handshake (22)
      - Version: TLS 1.0 (0x0301)
        - Length: 512
      - Handshake Protocol: Client Hello
        - Handshake Type: Client Hello (1)
          - Version: TLS 1.2 (0x0303)
            - Random: c3b792dec134b1fed6e43e3e445f2d8a8eeba21e1489757055d9478d69ee665
            - Session ID Length: 32
            - Session ID: 3f9db5885fd4f02942edd45f161a16e5847dcd9c7f77a8137c9a008c8aa53ff1
            - Cipher Suites Length: 32
              - Cipher Suites (16 suites)
            - Extension: server_name (len=14)
              - Type: server_name (0)
                - Length: 14
                  - Server Name Indication extension
                    - Server Name list length: 12
                    - Server Name Type: host_name (0)
                      - Server Name length: 9
                        - Server Name: gazeta.pl
              - Extension: extended_master_secret (len=0)
              - Extension: renegotiation_info (len=1)

OUTPUT
- Client Process Name
- OS Name
- Malware Probability

EVE
- Destination IP
- Destination Port
- TLS Fingerprint
- Domain Name

© 2023 Cisco and/or its affiliates. All rights reserved. Cisco Public

BRKSEC-3320
Offload Your Firewall with ML-Based Technology

- Detects malware in encrypted flows
- Can be used to allow/block in firewall policy
- Triggers Indications of Compromise
- Minimal performance impact
- Enriches Endpoint DB with Application and OS
Learn more about EVE

Using the Cisco Secure Firewall with the Encrypted Visibility Engine

• BRKSEC-2099
• By Costas Kleopa
  • Wednesday, Jun 71:00 PM – 2:00 PM PDT
  • Level 2, Breakers DJ
Fill out your session surveys!

Attendees who fill out a minimum of four session surveys and the overall event survey will get Cisco Live-branded socks (while supplies last)!

Attendees will also earn 100 points in the Cisco Live Game for every survey completed.

These points help you get on the leaderboard and increase your chances of winning daily and grand prizes.
• Visit the Cisco Showcase for related demos

• Book your one-on-one Meet the Engineer meeting

• Attend the interactive education with DevNet, Capture the Flag, and Walk-in Labs

• Visit the On-Demand Library for more sessions at www.CiscoLive.com/on-demand
Thank you
Gamify your Cisco Live experience! Get points for attending this session!

How:
1. Open the Cisco Events App.
2. Click on ‘Cisco Live Challenge’ in the side menu.
3. Click on View Your Badges at the top.
4. Click the + at the bottom of the screen and scan the QR code: