Playback: A TLS 1.3 Story

Alfonso García Alguacil *and* Alejo Murillo Moya

Cisco

Who are we?

Alfonso García Alguacil

Senior Security Consultant

CISCO

https://www.linkedin.com/in/alfonso-garcia-alguacil/

Alejo Murillo Moya

Red Team Lead EMEAR

CISCO

https://www.linkedin.com/in/alexismm2/

Introducing TLS 1.3

- The Good
  - KISS – Only 5 ciphers supported
  - No vulnerable to known attacks against previous versions of TLS
  - Welcome Forward Secrecy
  - Formal security analysis performed to the protocol

Introducing TLS 1.3

- The Good
  - KISS – Only 5 ciphers supported
  - No vulnerable to known attacks against previous versions of TLS
  - Welcome Forward Secrecy
  - Formal security analysis performed to the protocol

Introducing TLS 1.3

- The Good
  - KISS – Only 5 ciphers supported
  - No vulnerable to known attacks against previous versions of TLS
  - Welcome Forward Secrecy
  - Formal security analysis performed to the protocol

Introducing TLS 1.3

- The Good
  - KISS – Only 5 ciphers supported
  - No vulnerable to known attacks against previous versions of TLS
  - Welcome Forward Secrecy
  - Formal security analysis performed to the protocol

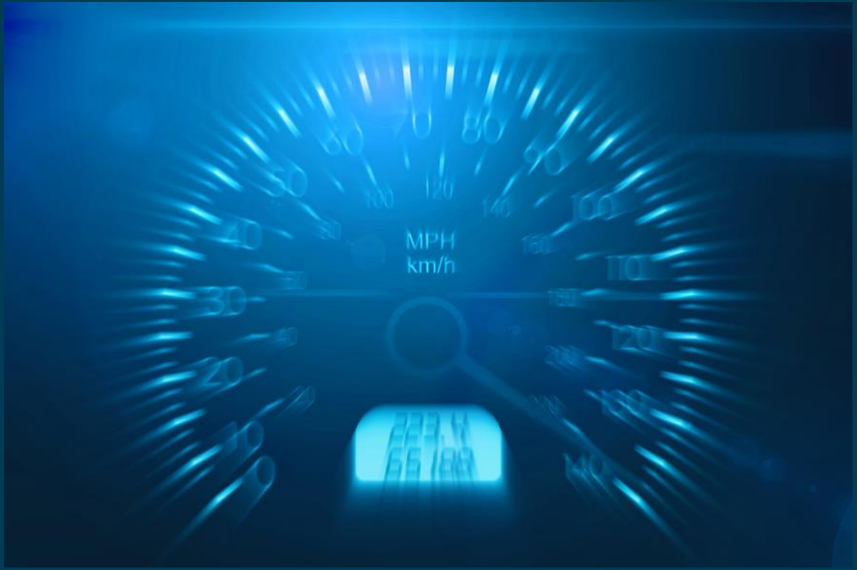
Introducing TLS 1.3

- The Bad
  - Protocol tainted due to “compatibility issues”

Introducing TLS 1.3

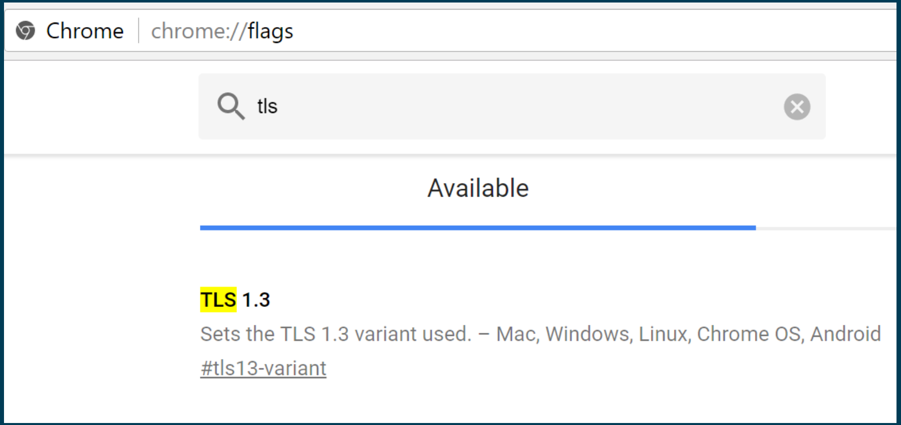
- The Ugly
  - New protocol feature: 0-RTT

0-RTT: Tough decisions



Why should I care?

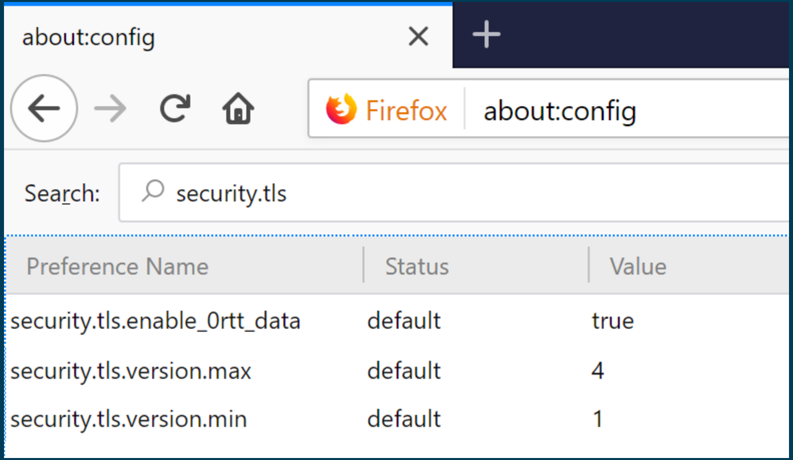
"Your browsers…



TLS 1.3

Sets the TLS 1.3 variant used. – Mac, Windows, Linux, Chrome OS, Android

#tls13-variant



Search: security.tls

Preference Name	Status	Value
security.tls.enable_0rtt_data	default	true
security.tls.version.max	default	4
security.tls.version.min	default	1

Why should I care?

… implementations …

OpenSSL
Cryptography and SSL/TLS Toolkit

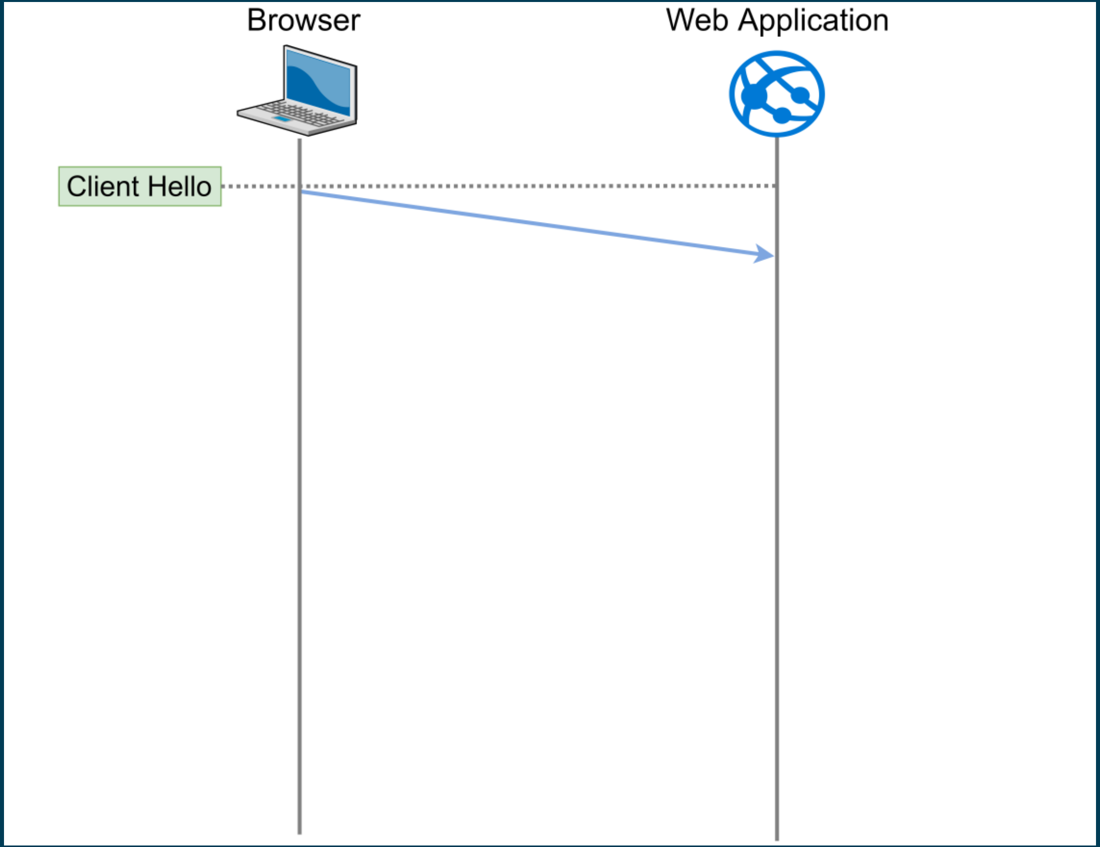
BoringSSL

Why should I care?

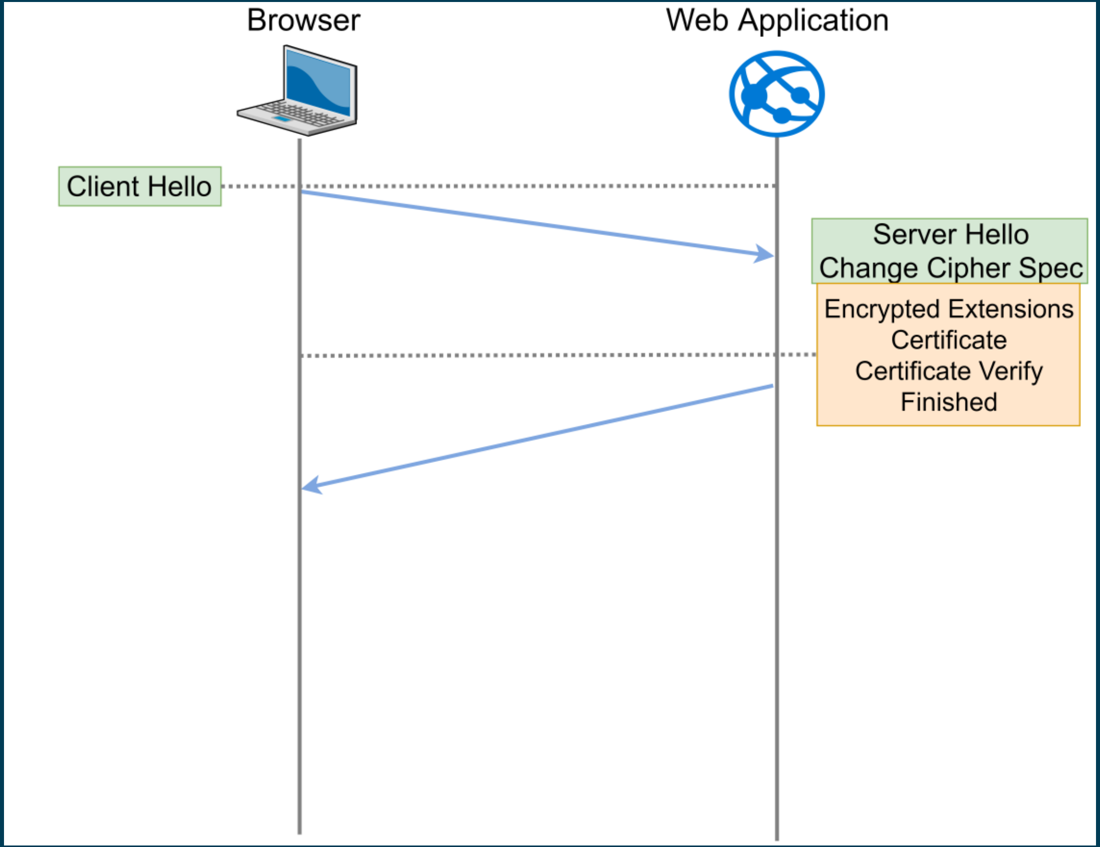
… and CDNs may be supporting TLS 1.3 with 0-RTT”

TLS 1.3 Handshake

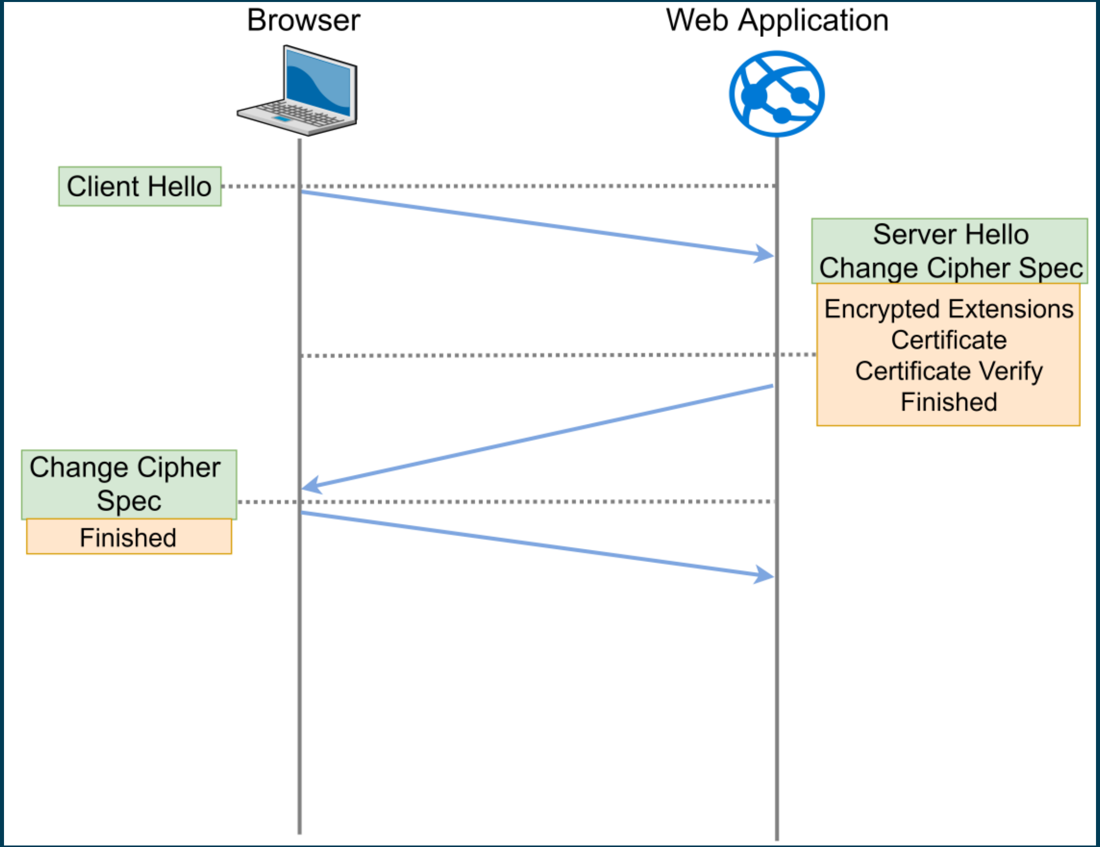
TLS 1.3 Handshake



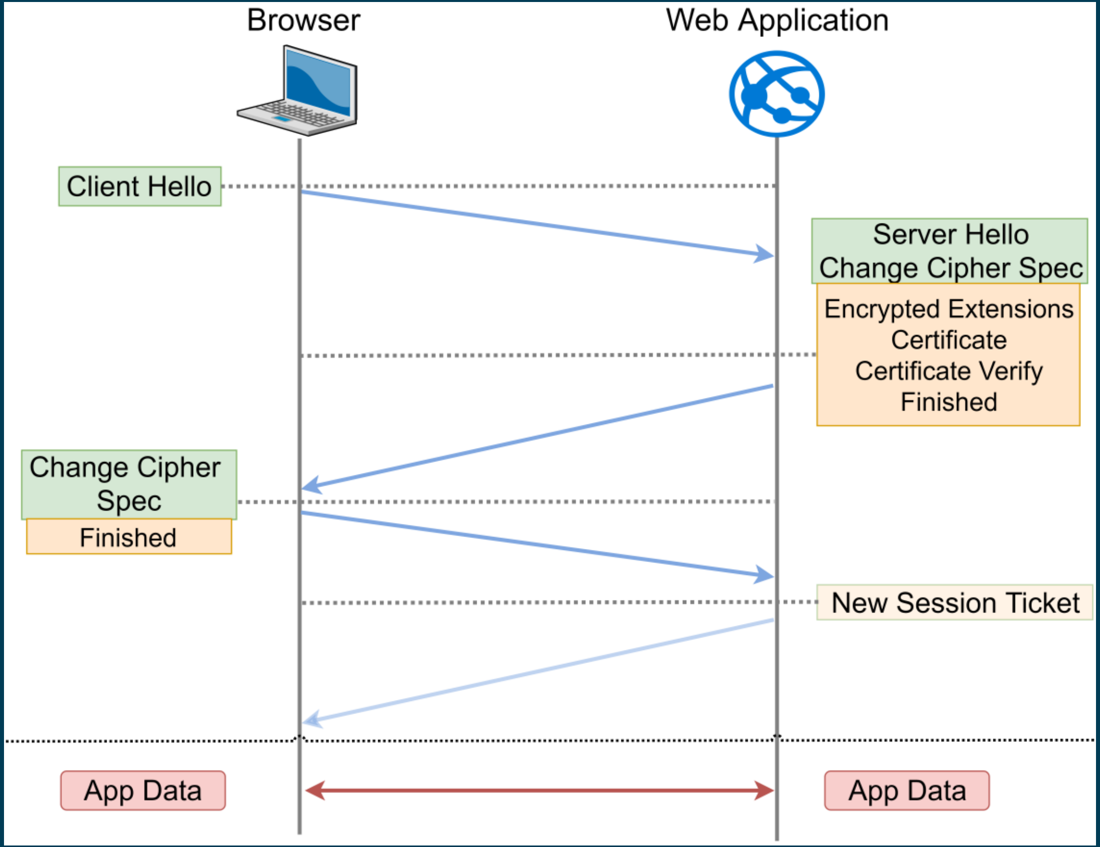
TLS 1.3 Handshake



TLS 1.3 Handshake

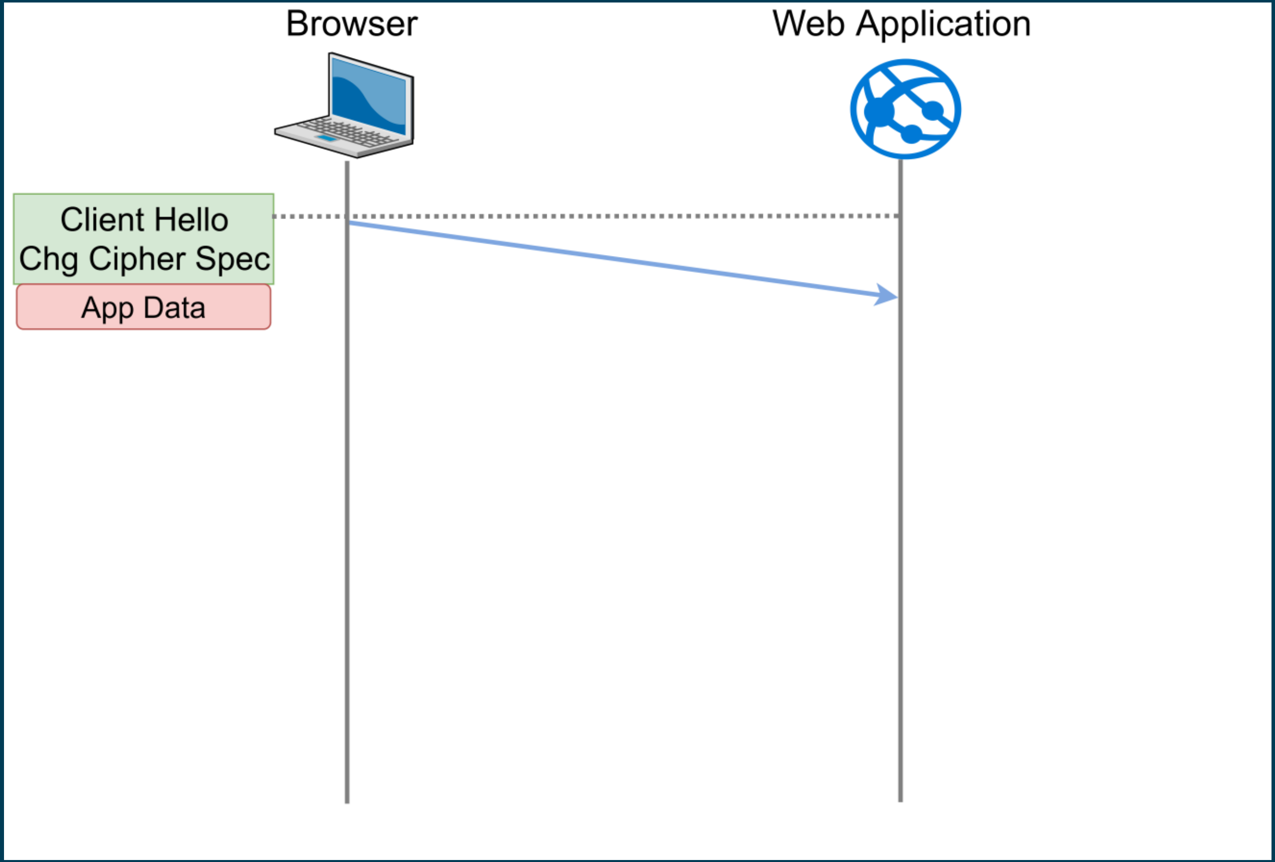


TLS 1.3 Handshake

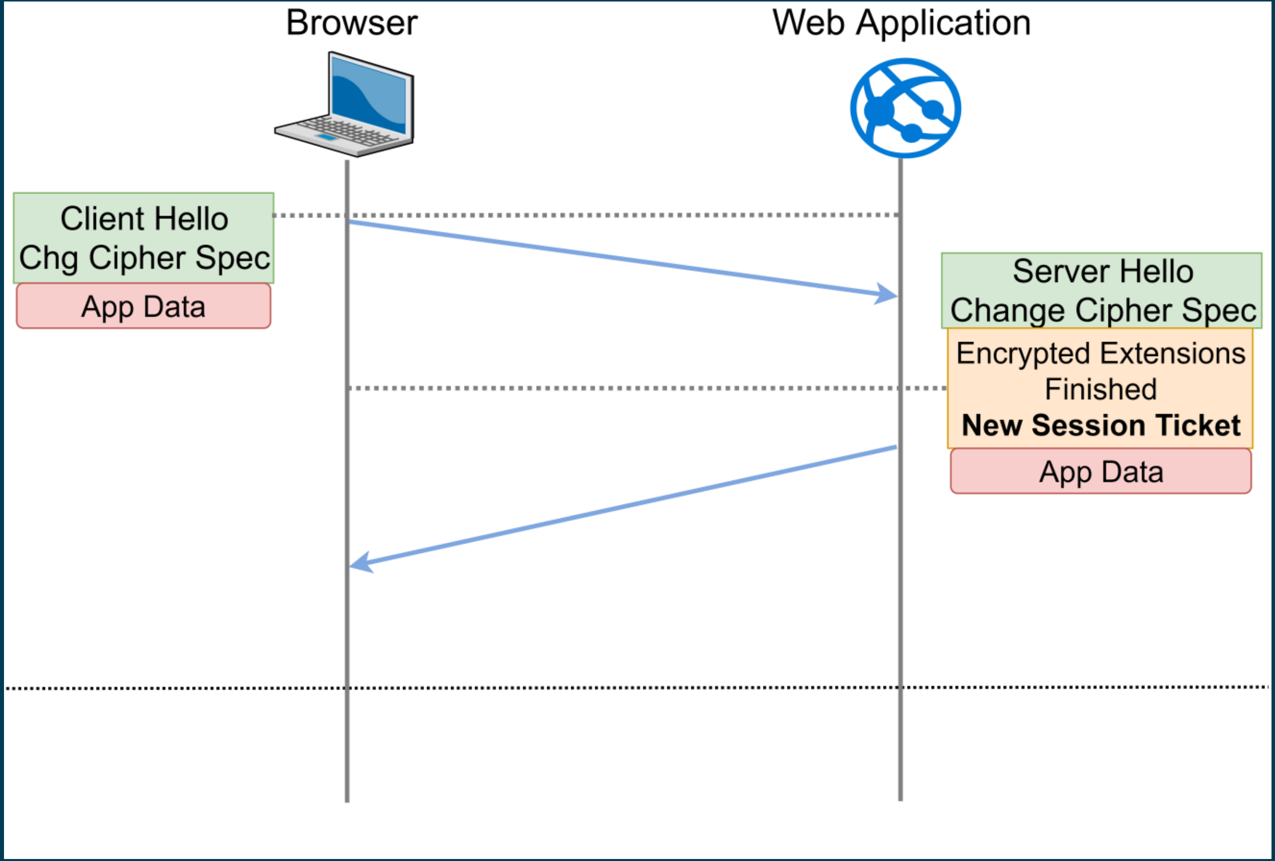


TLS 1.3 0-RTT

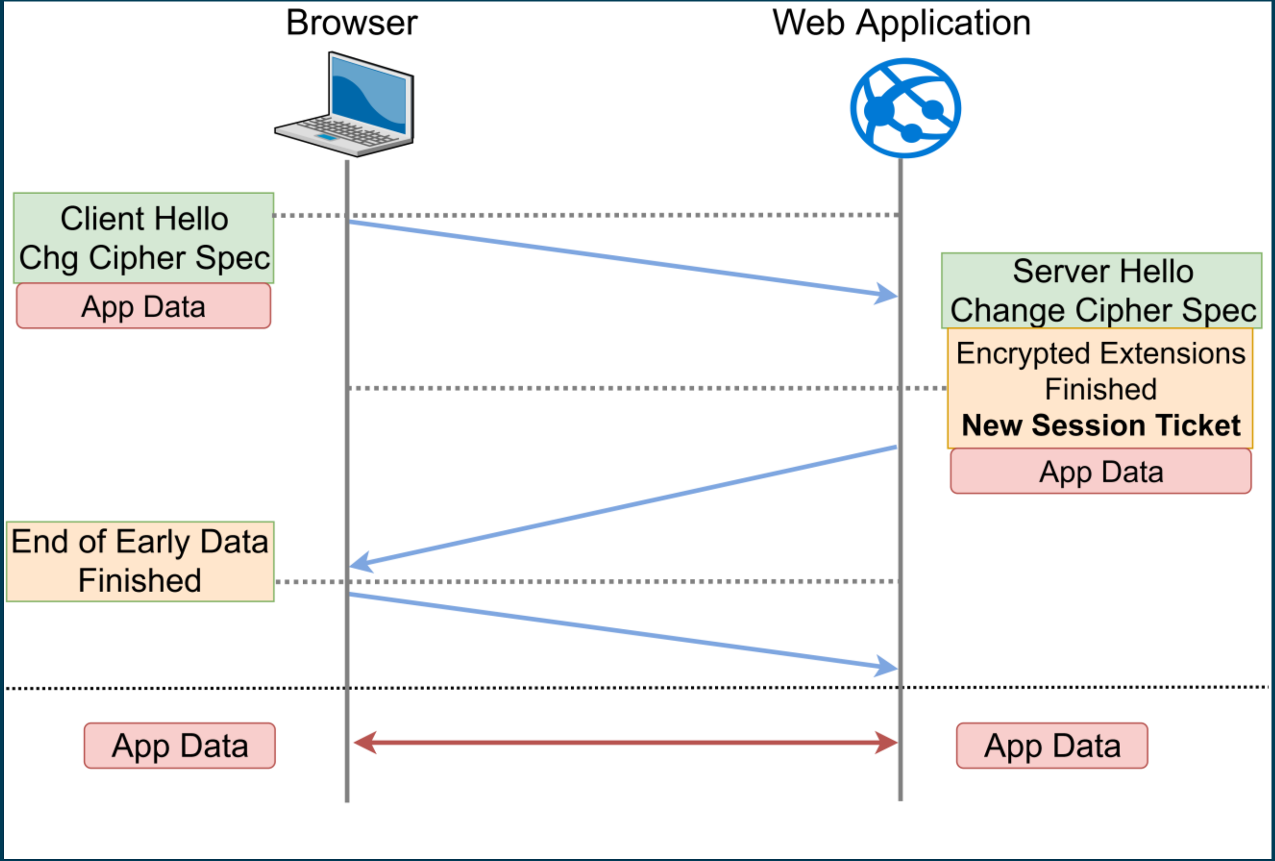
TLS 1.3 0-RTT


Browser
Web Application
Client Hello
Chg Cipher Spec
App Data

TLS 1.3 0-RTT


Browser
Web Application
Client Hello
Chg Cipher Spec
App Data
Server Hello
Change Cipher Spec
Encrypted Extensions
Finished
New Session Ticket
App Data

TLS 1.3 0-RTT

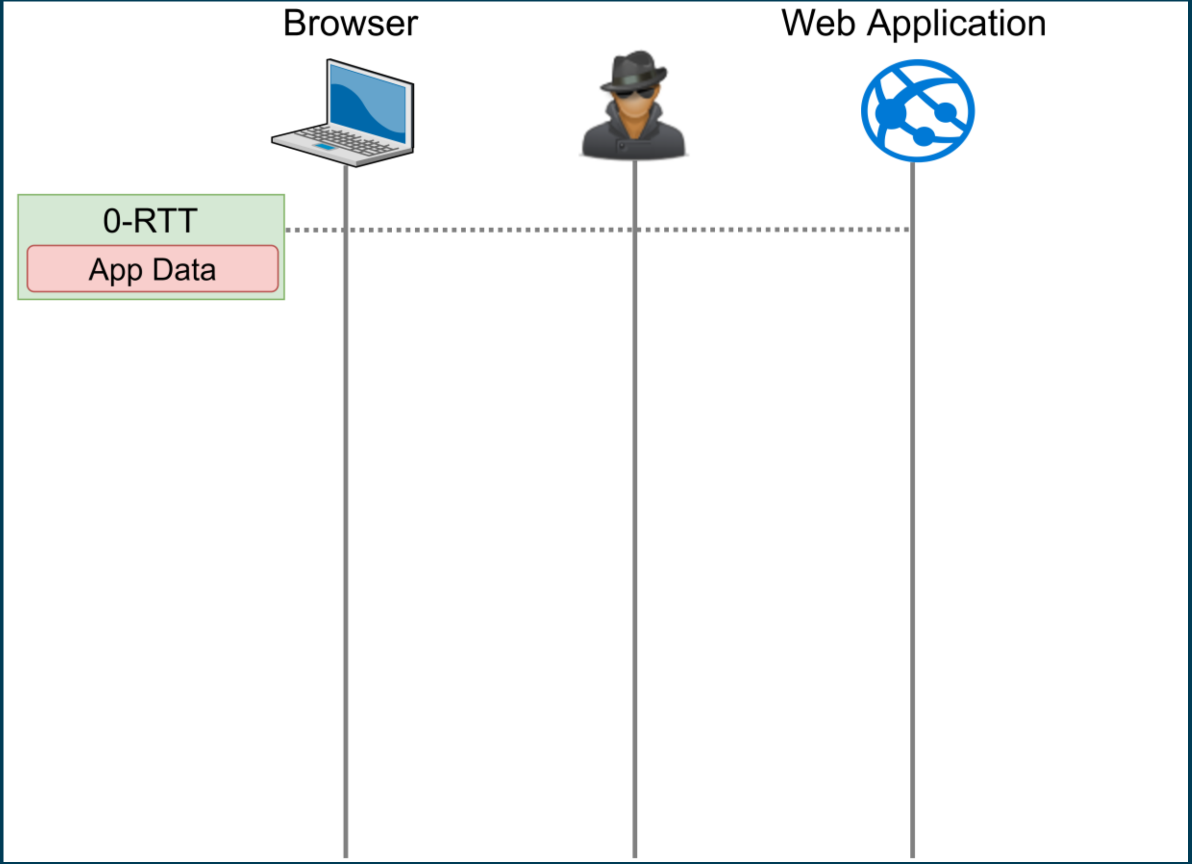

Browser
Web Application
Client Hello
Chg Cipher Spec
App Data
Server Hello
Change Cipher Spec
Encrypted Extensions
Finished
New Session Ticket
App Data
End of Early Data
Finished
App Data
App Data

As you can see…

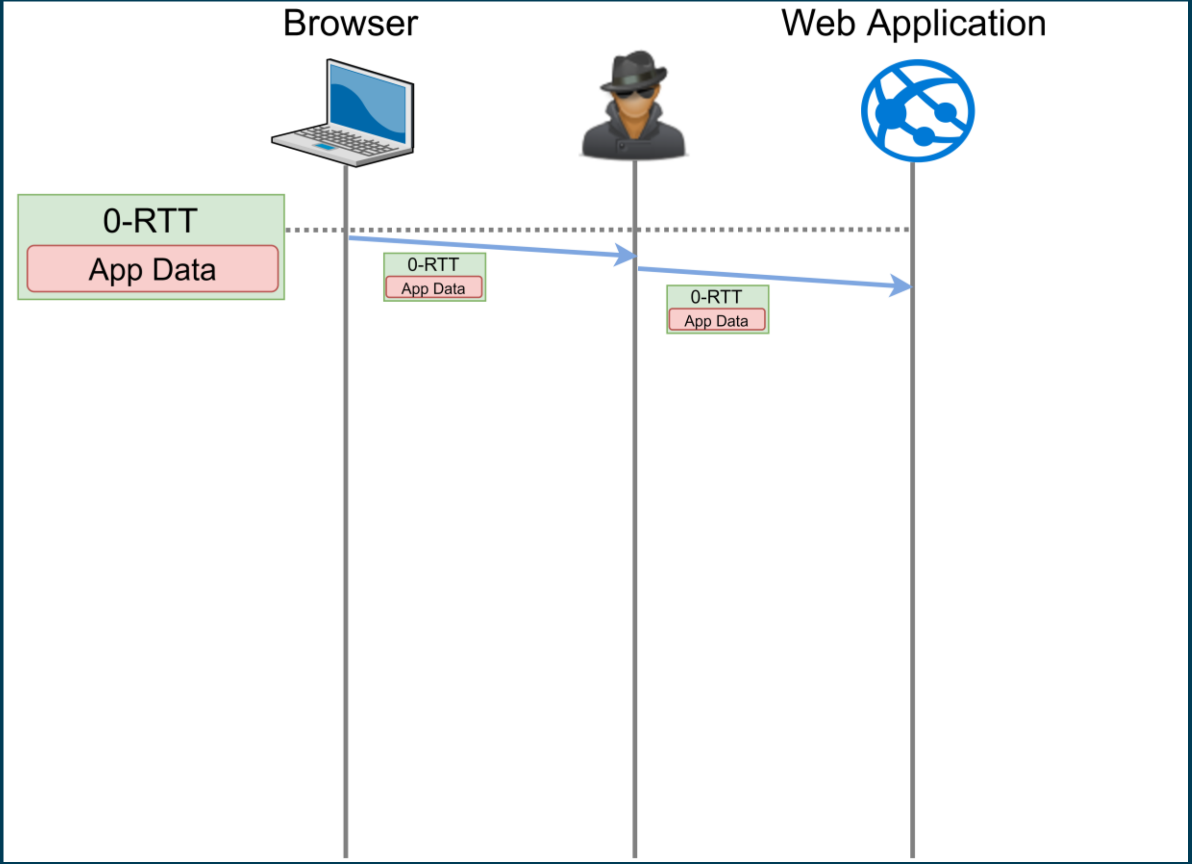
it may be possible to do REPLAY
REPLAY
REPLAY
REPLAY
REPLAY attacks!

TLS 1.3 0-RTT replay attack

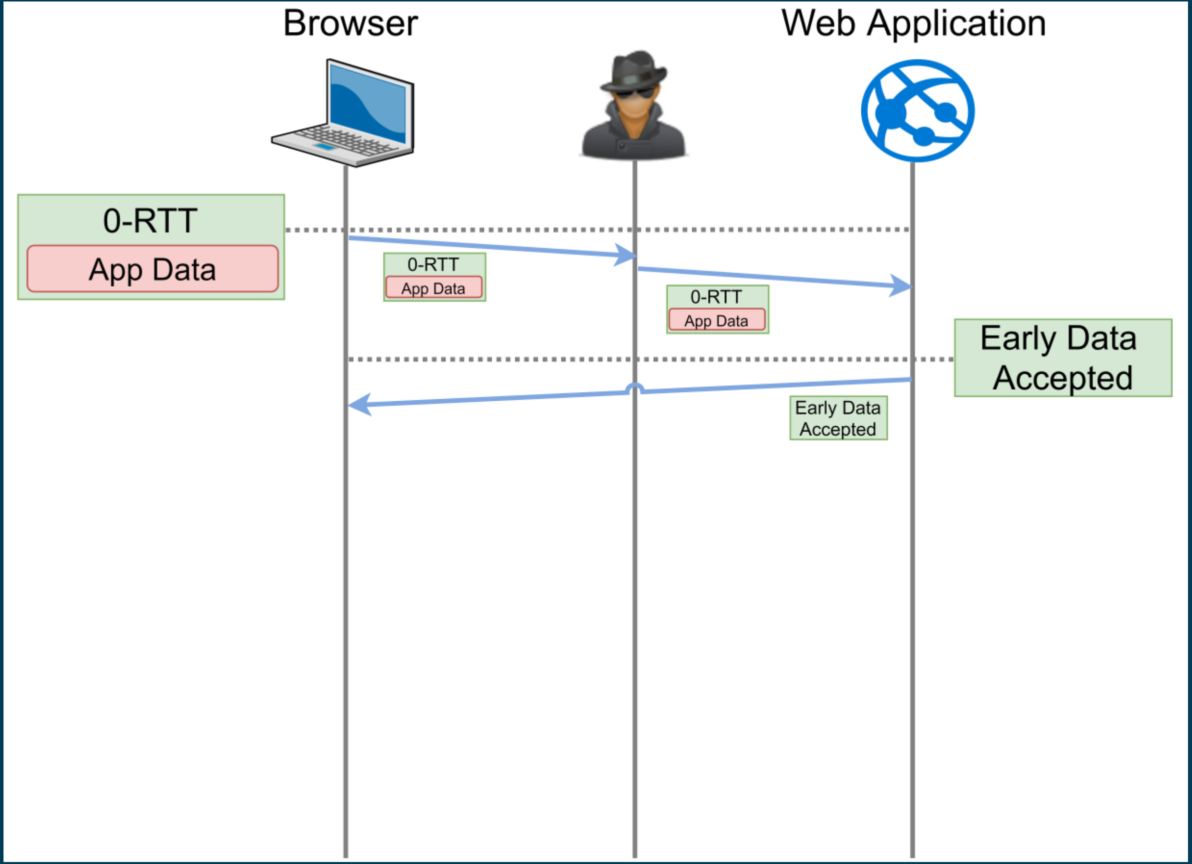
TLS 1.3 0-RTT Replay



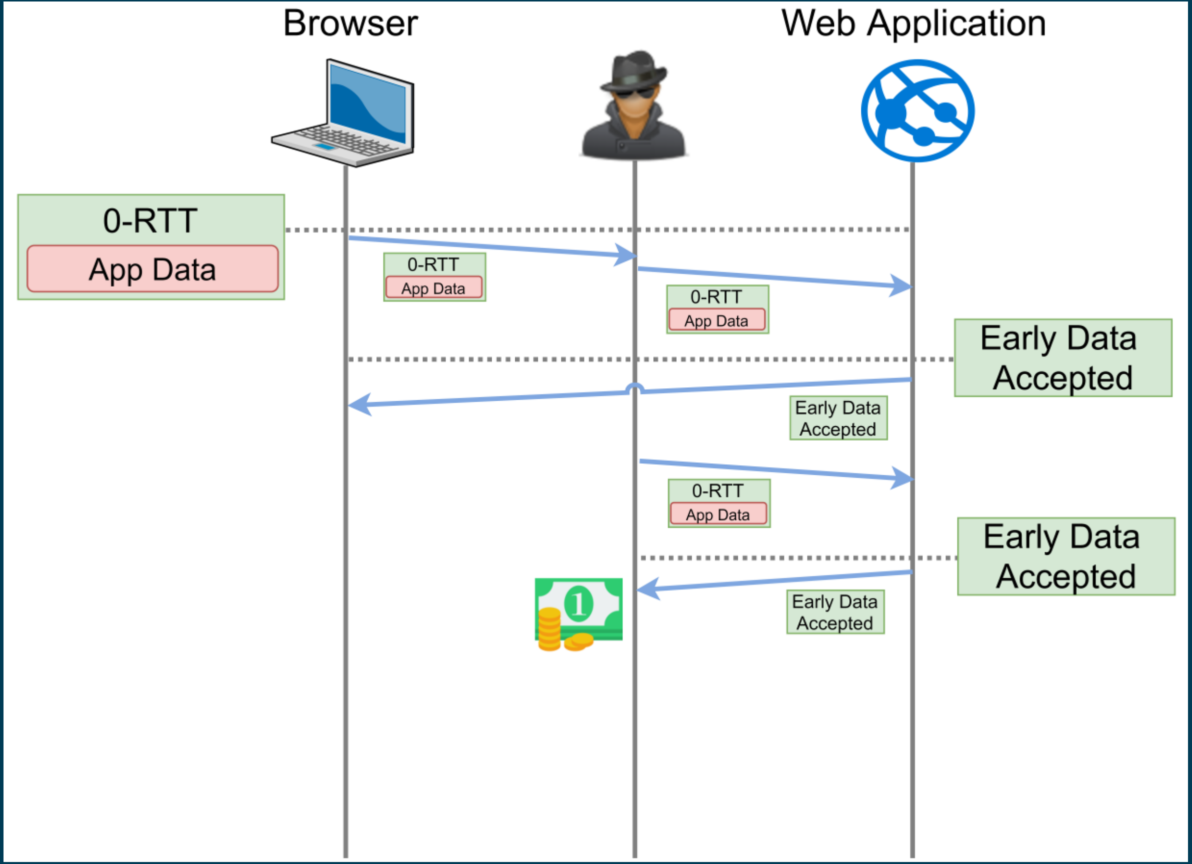
TLS 1.3 0-RTT Replay



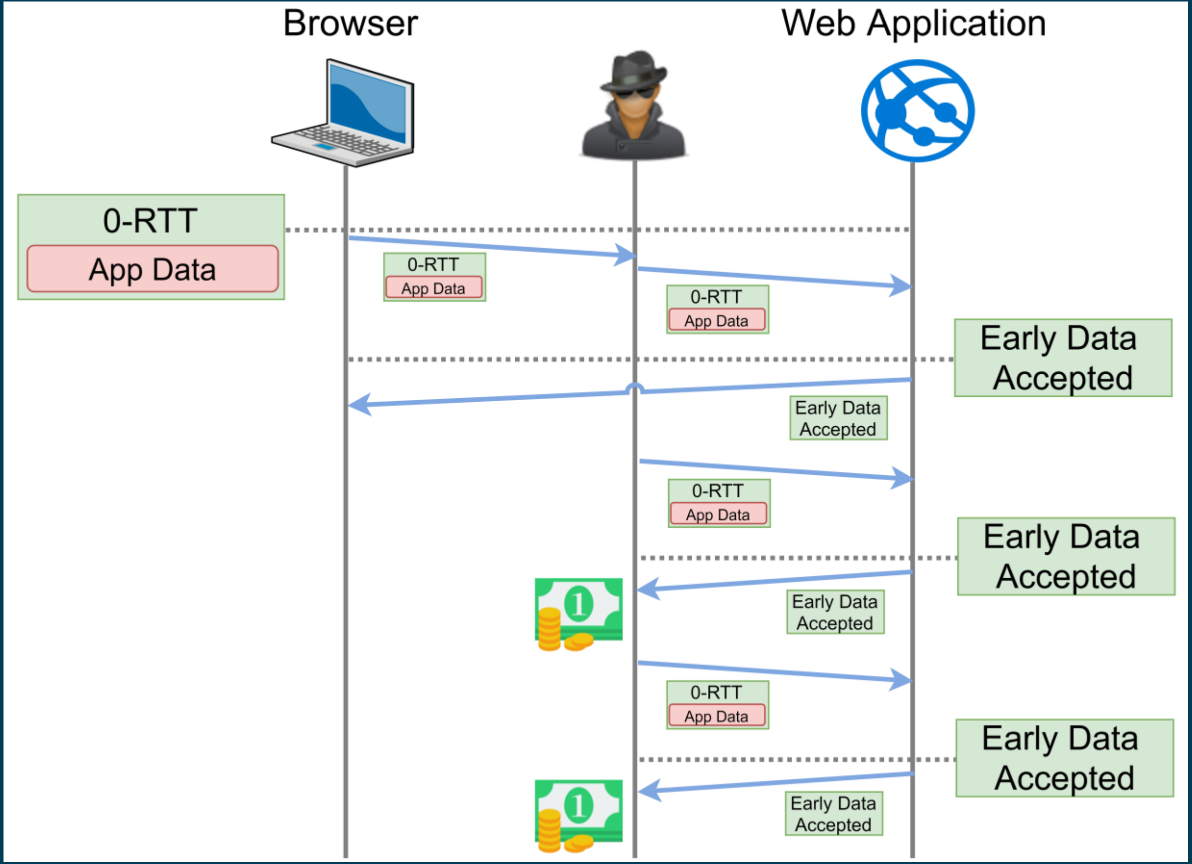
TLS 1.3 0-RTT Replay



TLS 1.3 0-RTT Replay



TLS 1.3 0-RTT Replay



Anti-replay protections

Single-Use Tickets

Anti-replay protections

Single-Use Tickets

Client-Hello Recording

Anti-replay protections

Single-Use Tickets

Client-Hello Recording

“Freshness” checks

Anti-replay protections

Single-Use Tickets

Client-Hello Recording

“Freshness” checks

Application profiles

Anti-replay protections

Single-Use Tickets

Client-Hello Recording

“Freshness” checks

Application profiles

Separate API

Anti-replay protections and mitigations

Anti-replay PROTECTIONS (Jul-2018)

	Single-Use Tickets	Client-Hello Recording	Application Profile	Other protections
OpenSSL Cryptography and SSL/TLS Toolkit	✓		n/a	Different API for handling 0-RTT
BoringSSL			n/a	0-RTT disabled by default
CLOUDFLARE	✓		Partial (HTTP Header)	0-RTT disabled. “safe” methods, no params
	n/a		n/a	0-RTT not available
	n/a		n/a	0-RTT only on “safe” methods

Anatomy of an attack

- **Vantage point** in the network
- Browser and server with TLS 1.3 and 0-RTT enabled
- GET not being a “*safe method*” (a.k.a. RFC meets reality)

Anatomy of an attack

- Vantage point in the network
- Browser and server with TLS 1.3 and 0-RTT enabled
- GET not being a “*safe method*” (a.k.a. RFC meets reality)

Anatomy of an attack

- Vantage point in the network
- Browser and server with TLS 1.3 and 0-RTT enabled
- GET not being a "*safe method*" (a.k.a. RFC meets reality)

The browser behaviour

- **The browser decides** when to send 0-RTT data, which reduces the window for attacks

DEMO

Improving our attack

- The browser decides when to send 0-RTT data, which reduces the window for attacks

- Could it be possible to **control** when to send 0-RTT data?

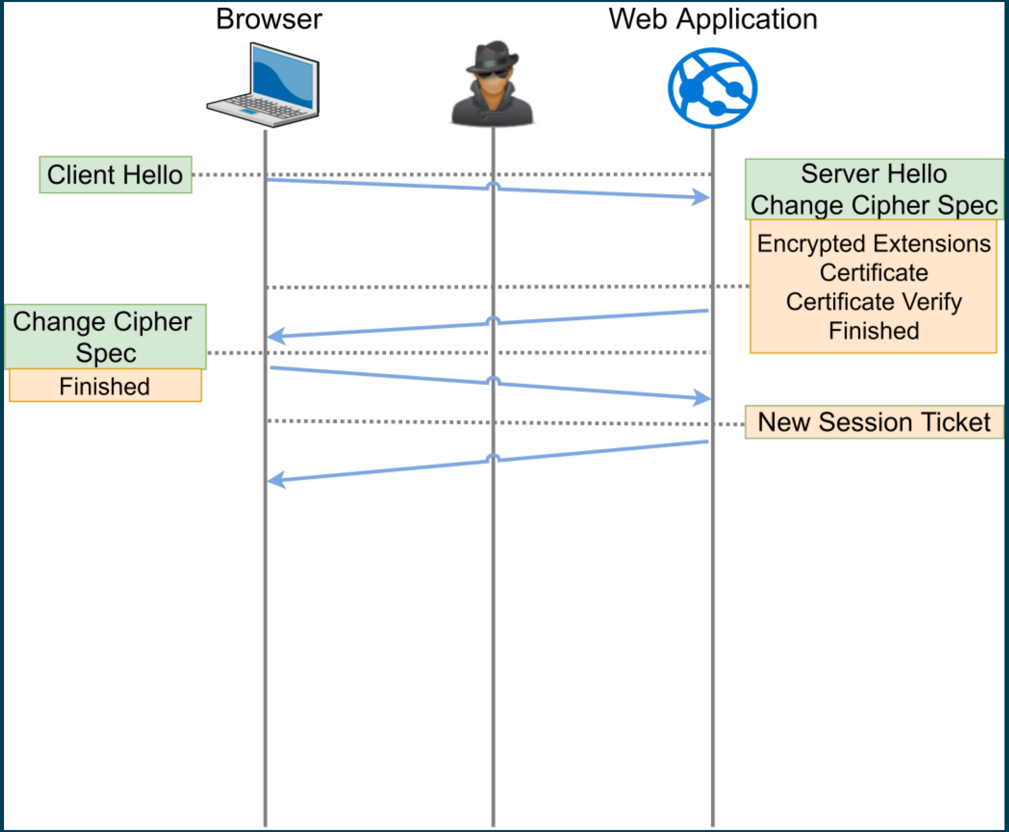
Improving our attack

- **The browser decides** when to send 0-RTT data, which reduces the window for attacks

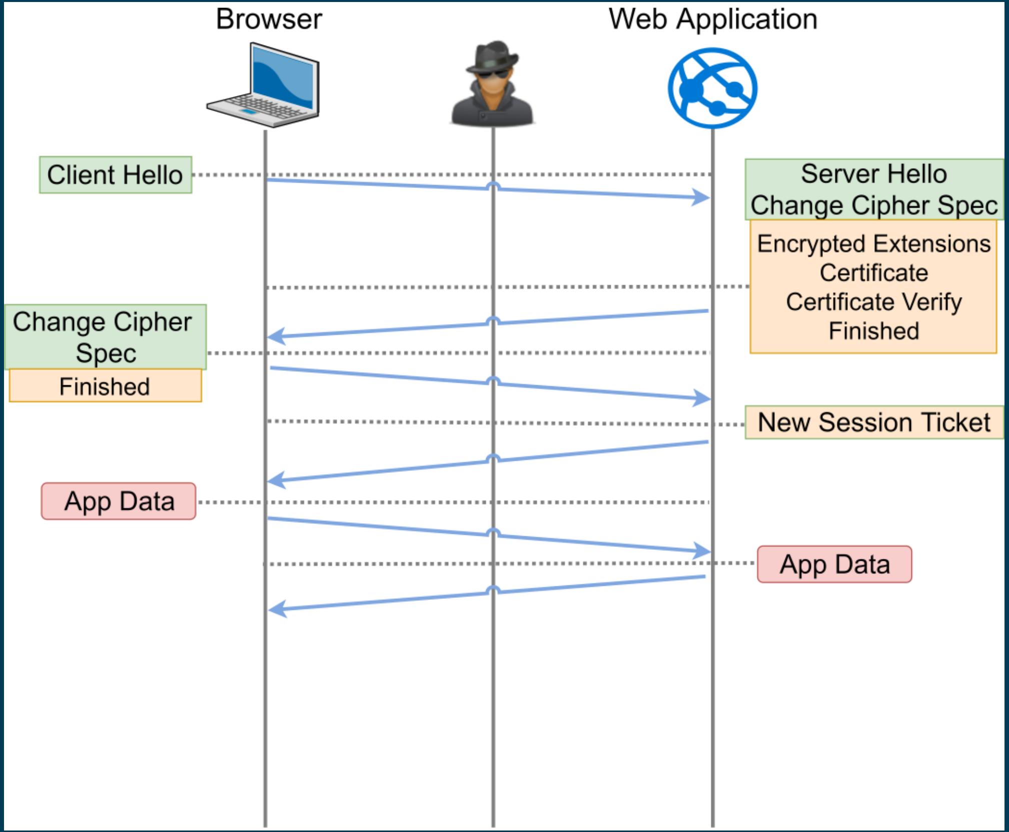
- Could it be possible to **control** when to send 0-RTT data?

YES

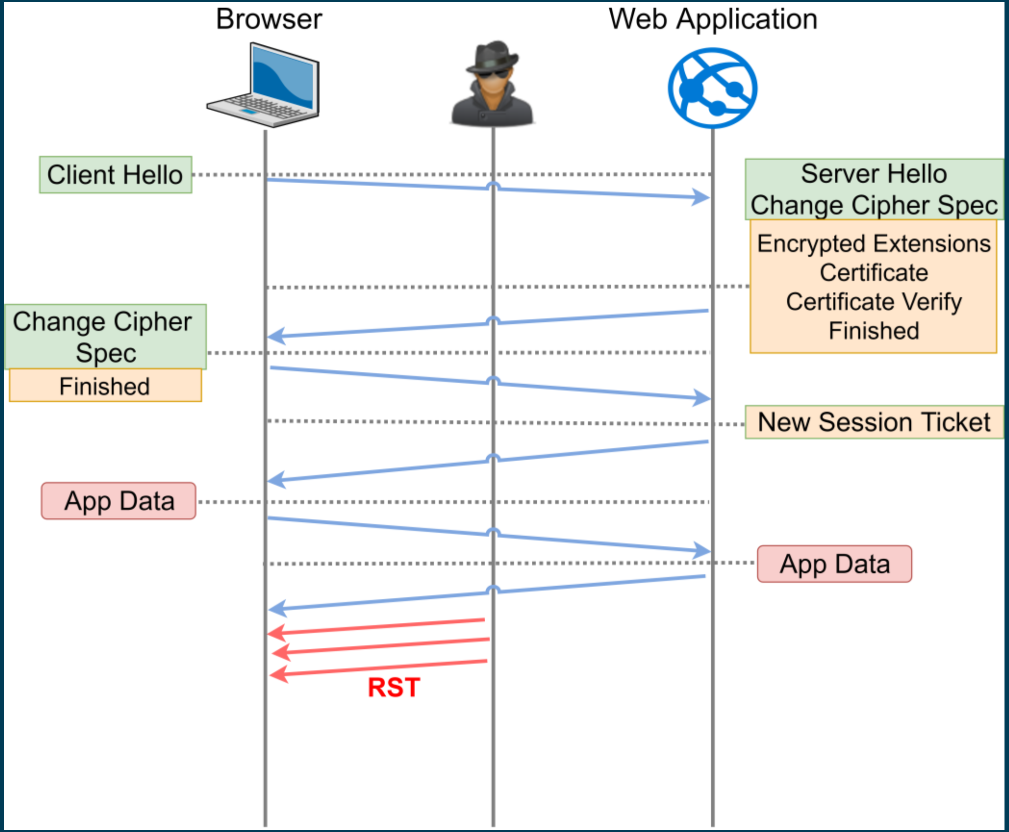
Controlling the browser



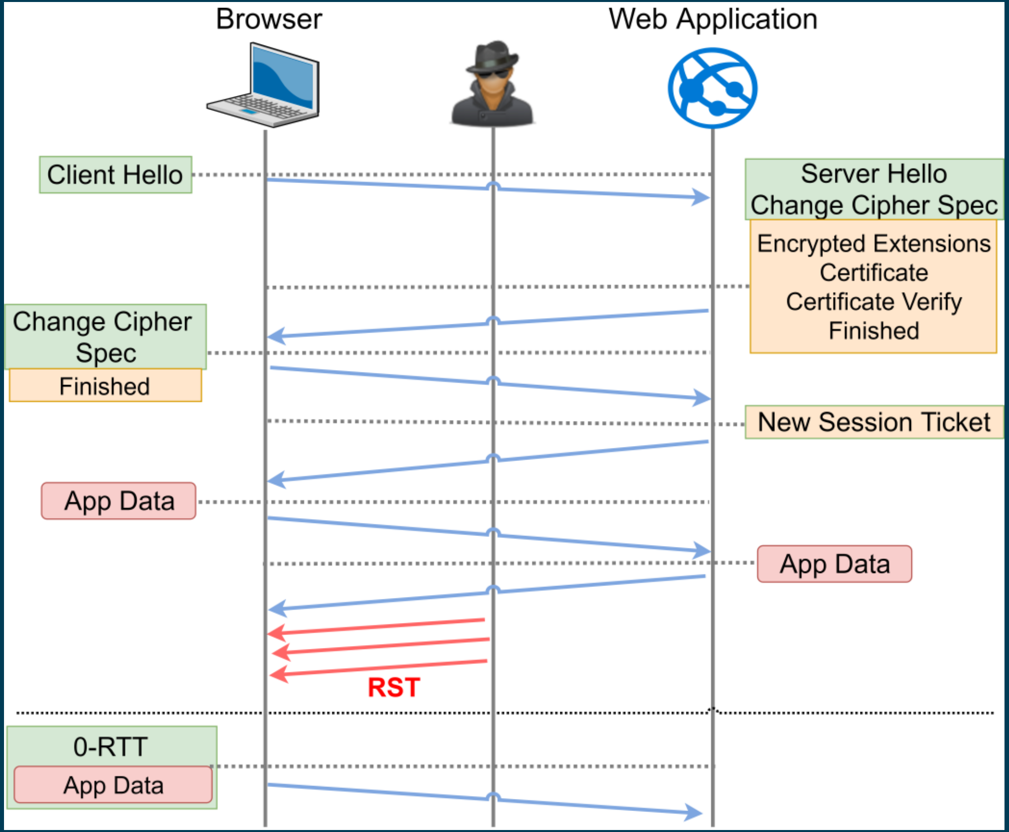
Controlling the browser



Controlling the browser



Controlling the browser



DEMO

Anti-replay protections

Single-Use Tickets

Client-Hello Recording

“Freshness” checks

Application profiles

Separate API

Improving our attack

- Imagine that somehow the TLS library and server actually **perfectly prevent** any replay attack on 0-RTT

Improving our attack

- Imagine that somehow the TLS library and server actually **perfectly prevent** any replay attack on 0-RTT

- Could it be possible to do replay attacks?

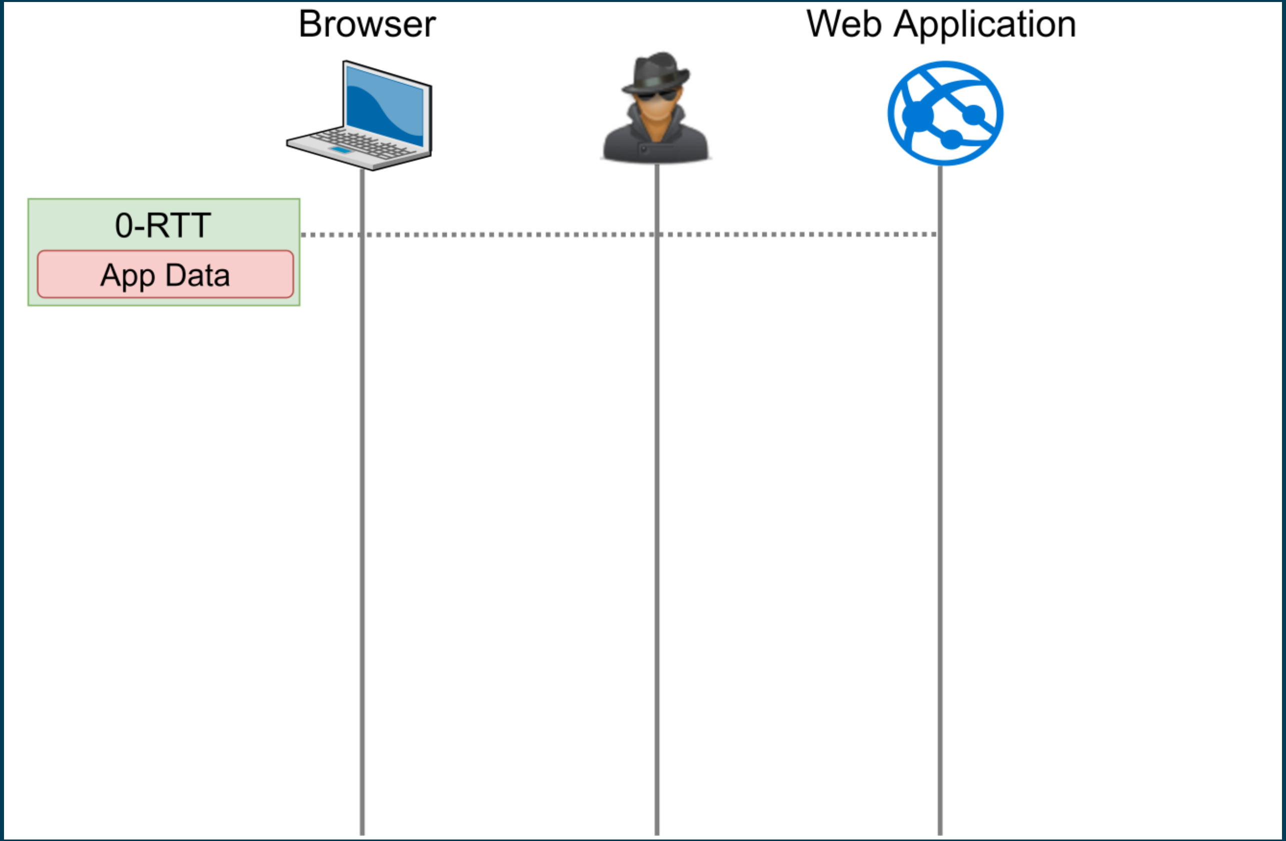
Improving our attack

- Imagine that somehow the TLS library and server actually **perfectly prevent** any replay attack on 0-RTT

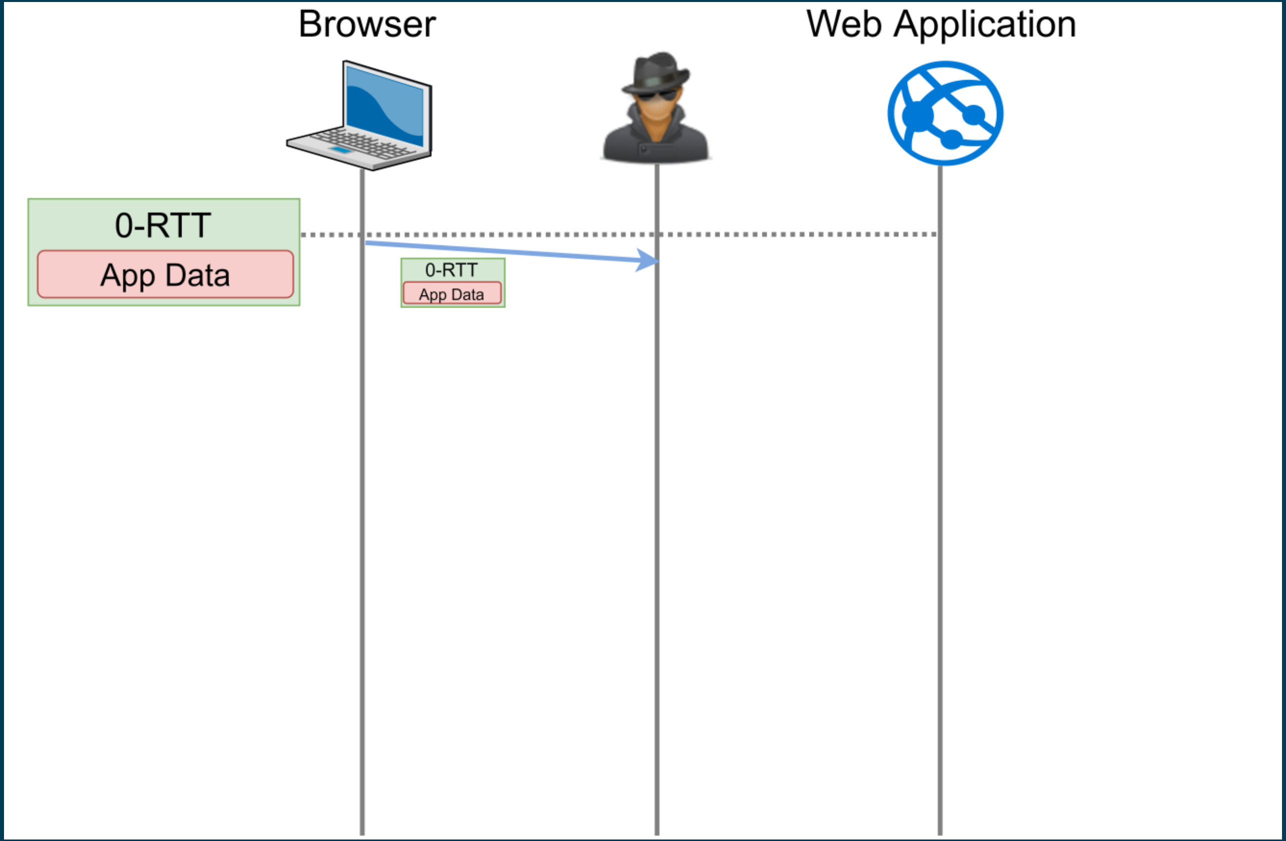
- Could it be possible to do replay attacks?

YES

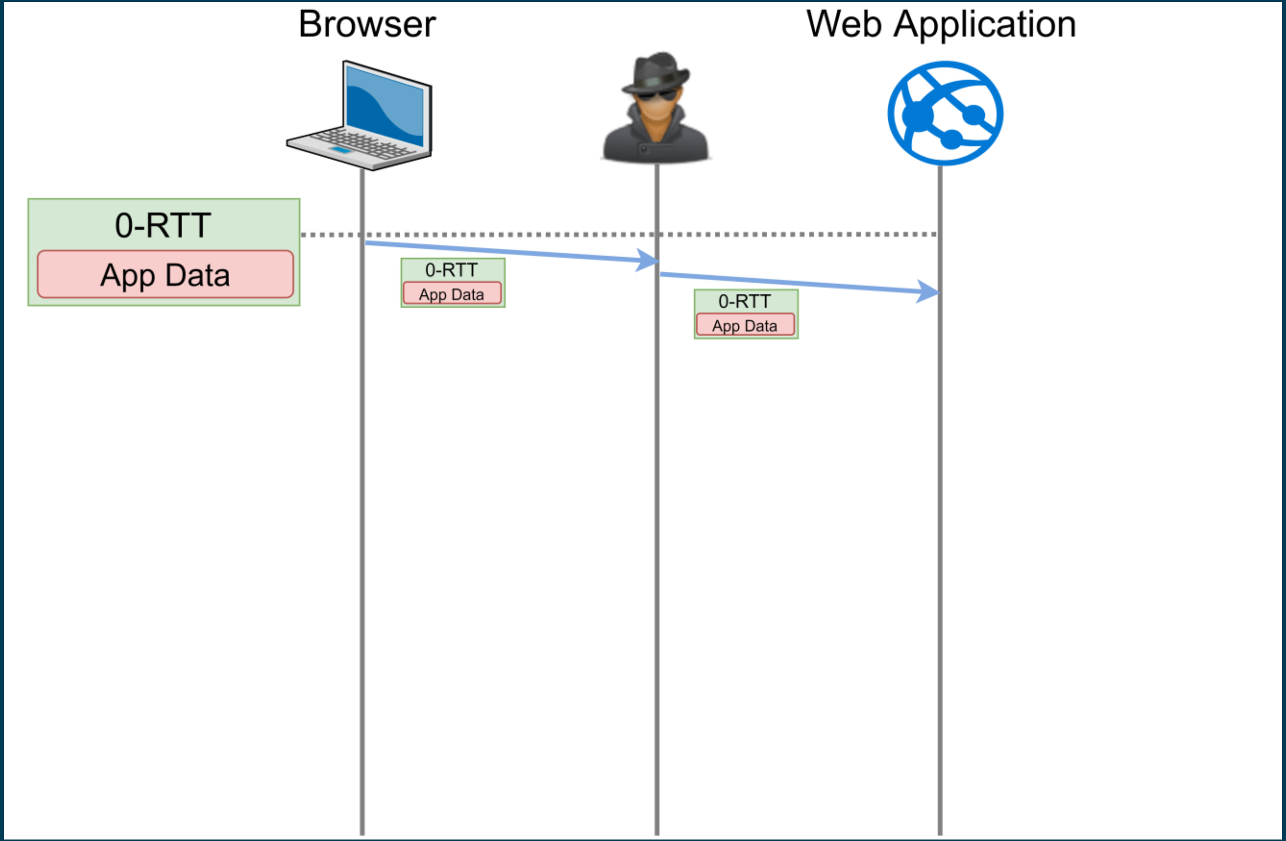
Universal replay attack



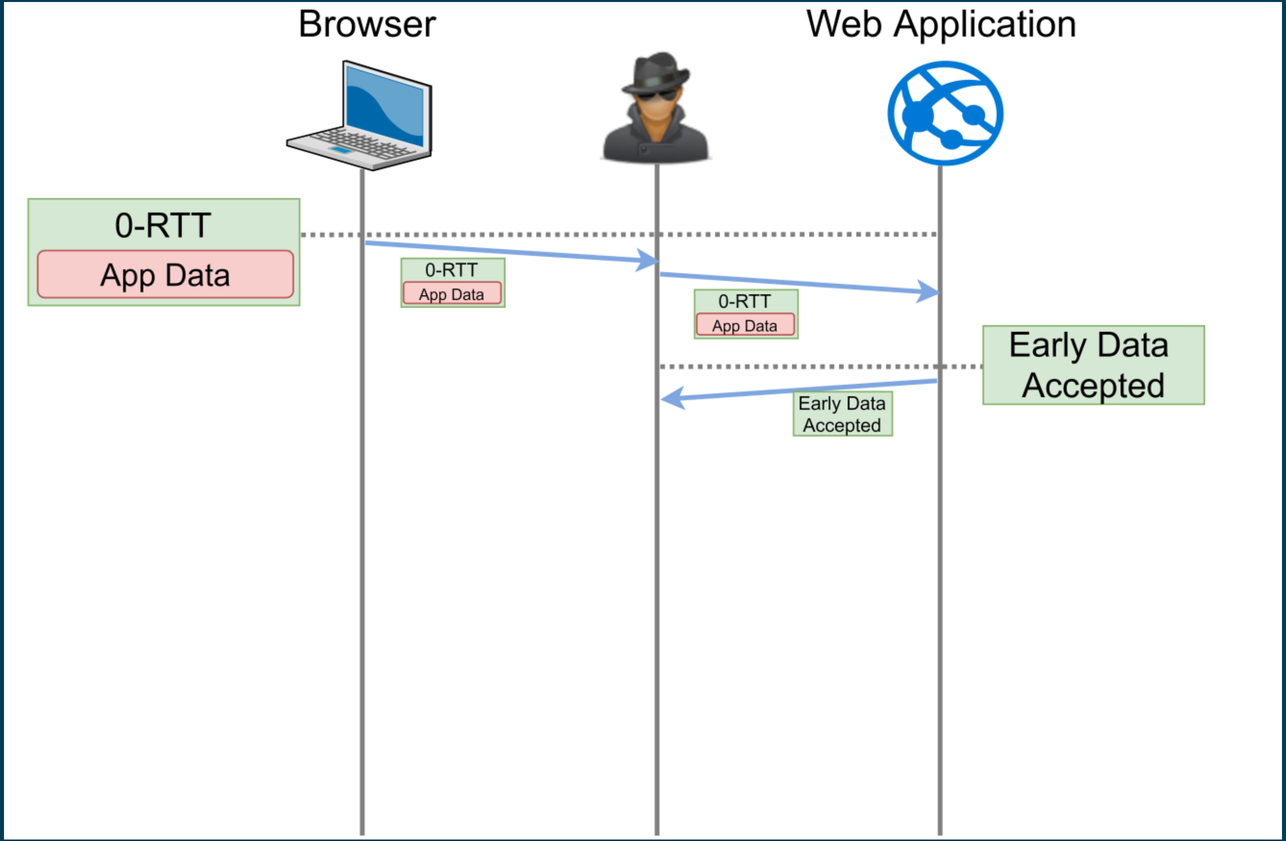
Universal replay attack



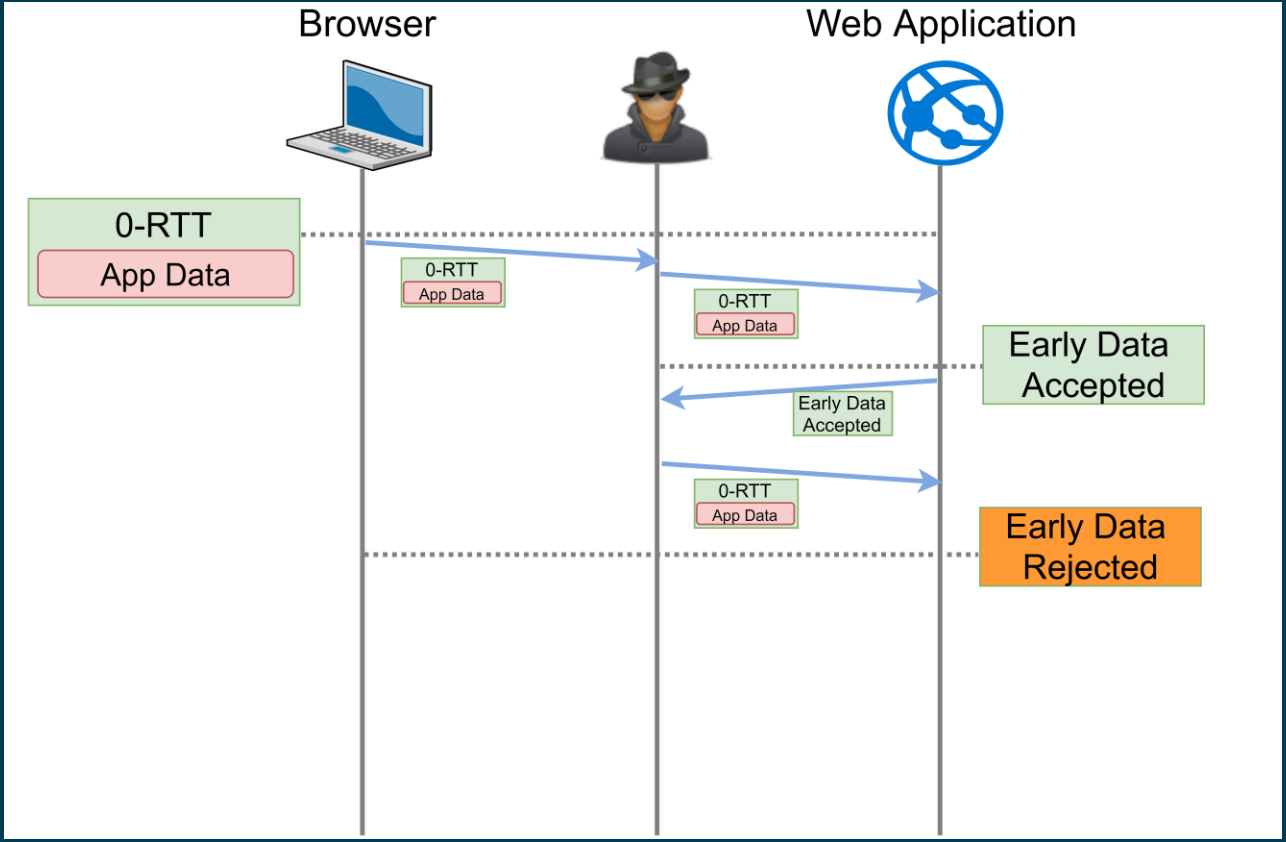
Universal replay attack



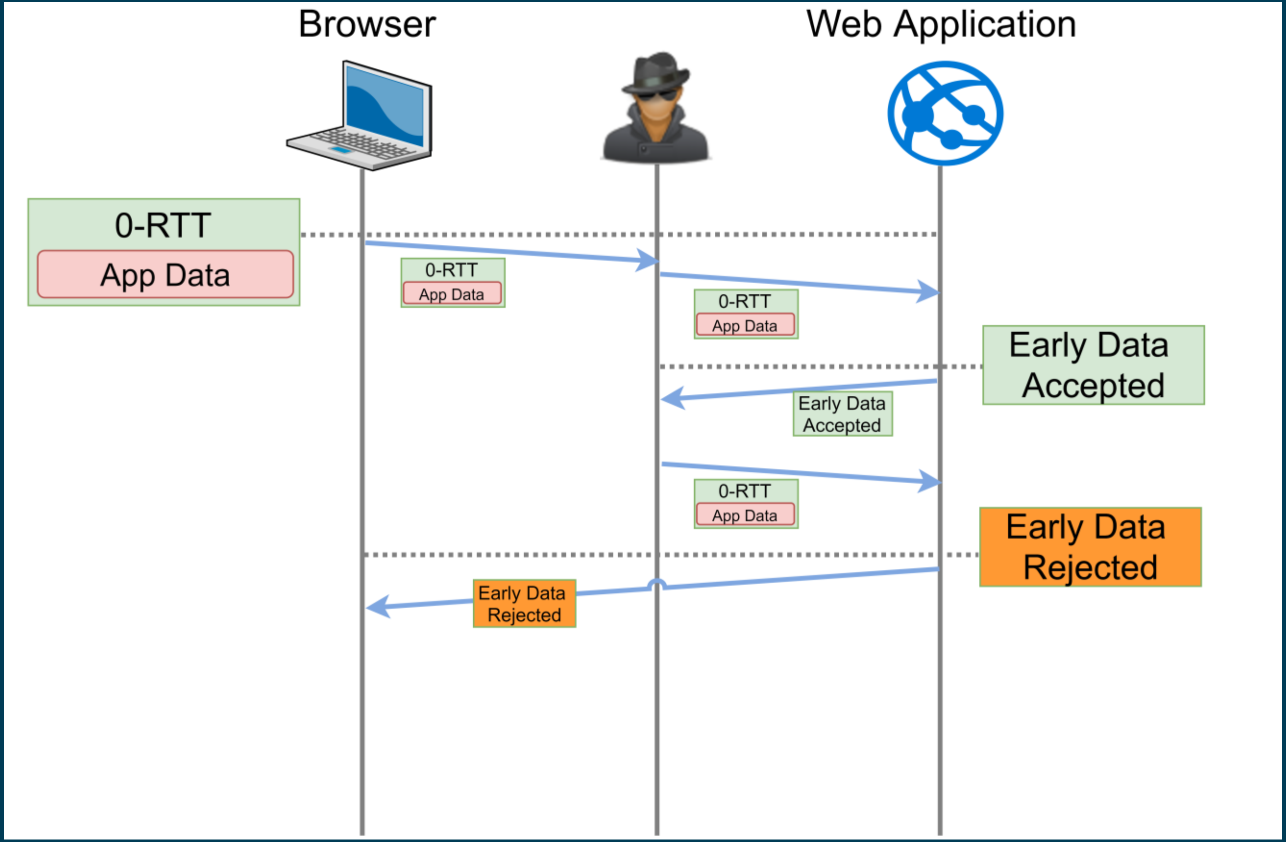
Universal replay attack



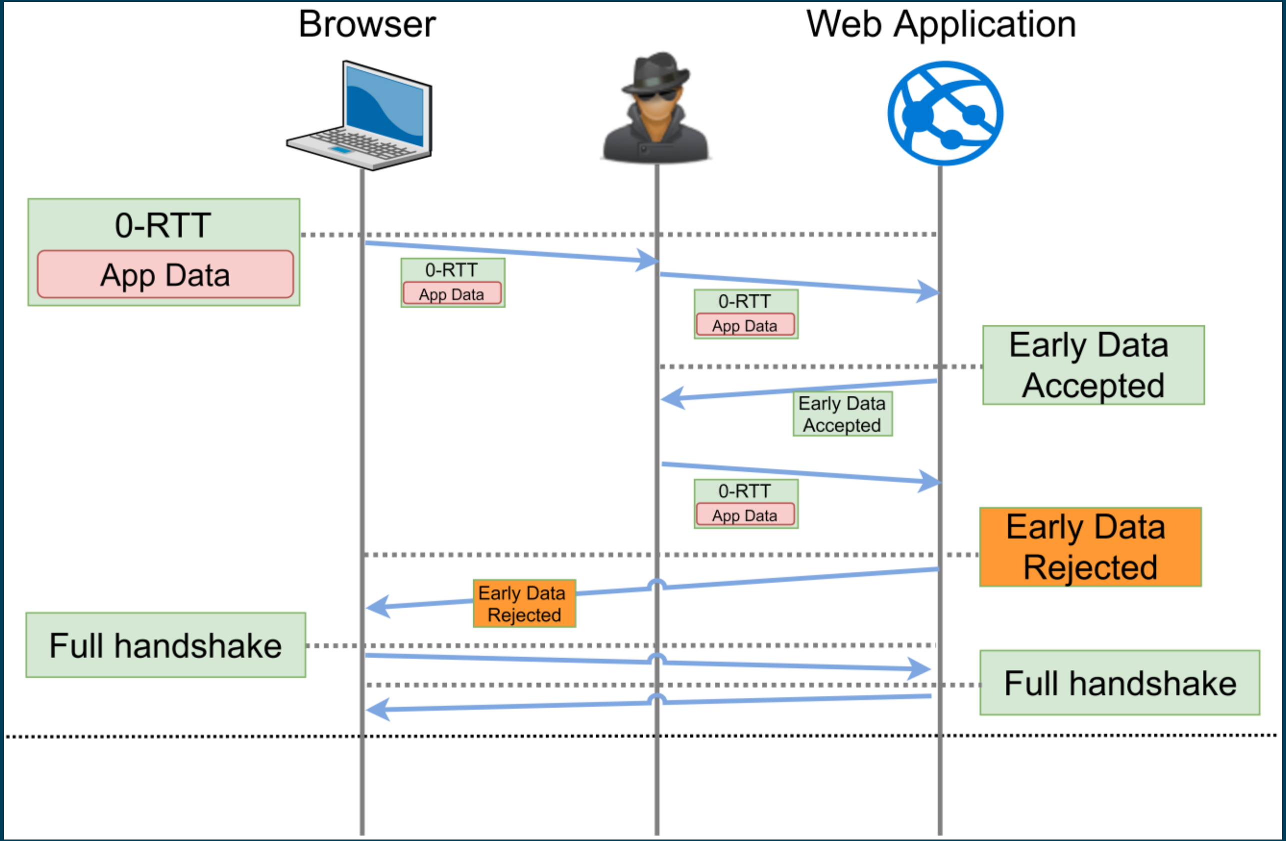
Universal replay attack



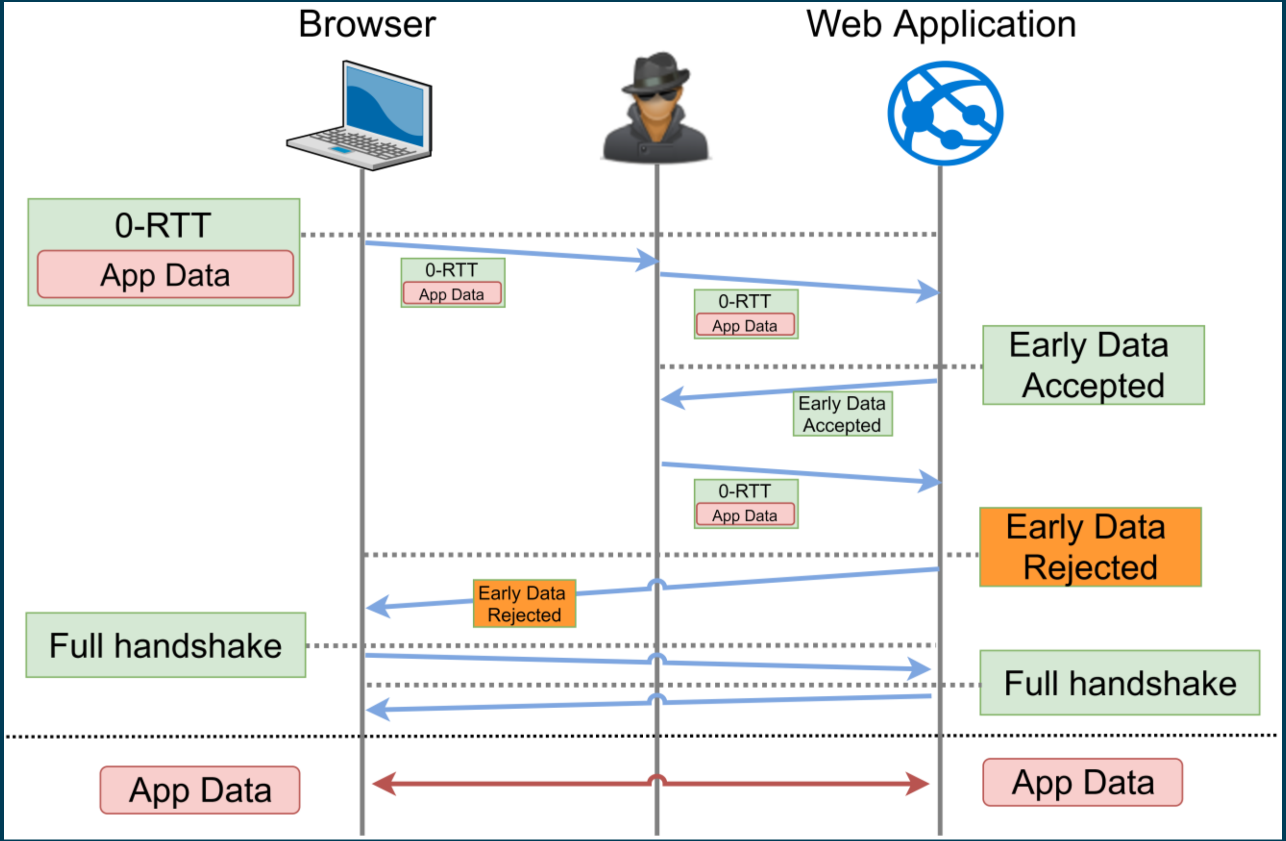
Universal replay attack



Universal replay attack



Universal replay attack



DEMO

Side effects of 0-RTT

- 0-RTT creates a **dependency between the application and** the underlying TLS 1.3 **protocol**
- The application will need to be **0-RTT aware**.
- Enabling 0-RTT could leave you application vulnerable to **replay attacks**
- Ultimately, the **last line of defence** would be the application itself.

Mitigations

- Disable 0-RTT
- Ensure that your application does not allow replays (e.g. CSRF). Ensure that REST services are developed properly
- Create an strict application profile after careful analysis.

Main takeaways

- **Adopt TLS 1.3**, but be aware could lead to a vulnerable application if 0-RTT is being used.
- **Your application needs to be 0-RTT aware** to prevent side effects.
- You will need to **take in account layers below your application**, as its configuration may protect or expose you against replay attacks

Thanks!