PLAYBACK: A TLS 1.3 STORY
WHO ARE WE?

Alfonso García Alguacil

Alejo Murillo Moya

CISCO

CISCO
INTRODUCING TLS 1.3

The **Good**

- KISS – Only 5 ciphers supported
- No vulnerable to the attacks impacting previous versions
- Welcome Forward Secrecy
- Formal security analysis performed to the protocol
INTRODUCING TLS 1.3

The **Good**

- KISS – Only 5 ciphers supported
- No vulnerable to the attacks impacting previous versions
- Welcome Forward Secrecy
- Formal security analysis performed to the protocol
INTRODUCING TLS 1.3

The Good

- KISS – Only 5 ciphers supported
- No vulnerable to the attacks impacting previous versions
- Welcome Forward Secrecy
- Formal security analysis performed to the protocol
INTRODUCING TLS 1.3

The **Good**

- KISS – Only 5 ciphers supported
- No vulnerable to the attacks impacting previous versions
- 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**

- 0-RTT (this talk 😊)
0-RTT: SPEED AT A COST
TELL ME AGAIN...

WHY SHOULD I CARE?
Your browsers…

…and CDNs may already be supporting TLS 1.3 0-RTT!

…implementations…

OpenSSL

Cryptography and SSL/TLS Toolkit

BoringSSL
TLS 1.3 HANDSHAKE

Browser

Client Hello

Web Application
TLS 1.3 HANDSHAKE
TLS 1.3 HANDSHAKE

Browser

Client Hello

Change Cipher Spec

Finished

Web Application

Server Hello
Change Cipher Spec

Encrypted Extensions
Certificate
Certificate Verify
Finished

Finished
TLS 1.3 HANDSHAKE

Browser

Client Hello

Change Cipher Spec

Finished

Web Application

Server Hello

Change Cipher Spec

Encrypted Extensions

Certificate

Certificate Verify

Finished

New Session Ticket

App Data

App Data
TLS 1.3 0-RTT

Browser

Web Application

Client Hello
Chg Cipher Spec
App Data
TLS 1.3 0-RTT

Diagram showing the communication between a Browser and a Web Application. The process includes:
- **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

<table>
<thead>
<tr>
<th>Client Hello</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chg Cipher Spec</td>
</tr>
<tr>
<td>App Data</td>
</tr>
</tbody>
</table>

Web Application

<table>
<thead>
<tr>
<th>Server Hello</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change Cipher Spec</td>
</tr>
<tr>
<td>Encrypted Extensions</td>
</tr>
<tr>
<td>Finished</td>
</tr>
<tr>
<td>New Session Ticket</td>
</tr>
<tr>
<td>App Data</td>
</tr>
</tbody>
</table>

End of Early Data Finished

App Data

App Data
As you can see...

it may be possible to do **REPLAY**
**REPLAY**
**REPLAY**
**REPLAY**
**REPLAY**
**REPLAY** attacks!
TLS 1.3 0-RTT REPLAY
TLS 1.3 0-RTT REPLAY
TLS 1.3 0-RTT REPLAY

Browser

0-RTT
App Data

Web Application

0-RTT
App Data

Early Data
Accepted

0-RTT
App Data

Early Data
Accepted

0-RTT
App Data

Early Data
Accepted
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

“Freshness” checks

Client-Hello Recording
ANTI-REPLAY PROTECTIONS

Single-Use Tickets

“Freshness” checks

Client-Hello Recording

Application profiles
ANTI-REPLAY PROTECTIONS

Single-Use Tickets

“Freshness” checks

Separate API

Client-Hello Recording

Application profiles
## ANTI-REPLAY PROTECTIONS (JUL-2018)

<table>
<thead>
<tr>
<th></th>
<th>Single-Use Tickets</th>
<th>Client-Hello Recording</th>
<th>Application Profile</th>
<th>Other protections</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OpenSSL</strong></td>
<td>![Green Checkmark]</td>
<td></td>
<td>n/a</td>
<td>Different API for handling 0-RTT</td>
</tr>
<tr>
<td><strong>BoringSSL</strong></td>
<td></td>
<td></td>
<td>n/a</td>
<td>0-RTT disabled by default</td>
</tr>
<tr>
<td><strong>Cloudflare</strong></td>
<td>![Green Checkmark]</td>
<td>![Green Checkmark]</td>
<td>Partial (HTTP Header)</td>
<td>0-RTT only on &quot;safe&quot; methods, no params</td>
</tr>
<tr>
<td><strong>Google Chrome</strong></td>
<td>n/a</td>
<td></td>
<td>n/a</td>
<td>0-RTT not available</td>
</tr>
<tr>
<td><strong>Mozilla Firefox</strong></td>
<td>n/a</td>
<td></td>
<td>n/a</td>
<td>0-RTT only on &quot;safe&quot; methods</td>
</tr>
</tbody>
</table>
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

Browser

Client Hello
Change Cipher Spec
Finished

Web Application

Server Hello
Change Cipher Spec
Encrypted Extensions
Certificate
Certificate Verify
Finished

New Session Ticket
CONTROLLING THE BROWSER
CONTROLLING THE BROWSER

Diagram showing the flow of data and messages between a browser and a web application, including steps such as Client Hello, Change Cipher Spec, Finished, App Data, Server Hello, Change Cipher Spec, Encrypted Extensions Certificate, Certificate Verify, Finished, New Session Ticket, and App Data.
CONTROLLING THE BROWSER
ANTI-REPLAY PROTECTIONS

Single-Use Tickets

“Freshness” checks

Separate API

Client-Hello Recording

Application profiles
IMPROVING OUR ATTACK (AGAIN)

- Imagine that somehow the TLS library and server actually **perfectly prevent** any replay attack on 0-RTT.
IMPROVING OUR ATTACK (AGAIN)

• 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 (AGAIN)

• 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

Diagram showing the process of a universal replay attack between a browser and a web application, highlighting the differences between 0-RTT and full handshake scenarios.
DEMO
TOOL: HIGH-LEVEL DESCRIPTION

• Assumes a vantage point in the network

• Provides creation of templates for encrypted traffic.

• Supports the two attacks described on this presentation.

• Available at https://github.com/portcullislabs/tlsplayback
SIDE EFFECTS OF 0-RTT

- It is important to understand that 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. strict CSRF). Ensure that REST services are developed properly

• Create an strict application profile after careful analysis
KEY TAKEAWAYS

• **TLS 1.3 is awesome**, but could lead to a vulnerable application if 0-RTT is being used.

• Your application (not just webapps) needs to be **0-RTT-aware** to prevent side effects

• You may need to change your application or server/CDN configuration to **protect against replay attacks**
Thanks!