Formal Analysis of Widevine DRM/EME

Stéphanie Delaune, Joseph Lallemand, Gwendal Patat, Florian Roudot, Mohamed Sabt
3 April 2024
DRM Systems

- **Digital Rights Management:**
  restrict uses of digital content – prevent copy, etc.

- Used for music, books, video games, video streaming...
DRM Systems

Over-the-top provider (OTT)
Licence server

Proprietary mechanisms

W3C EME

Content Decryption Module (CDM)

Protected media
Clear media playback

Protected media
Clear media playback

Protected media
Clear media playback

DRM System A

DRM System B

DRM System C

Over-the-top provider (OTT):
- Netflix
- Prime Video
- Disney+

Licence server:

Proprietary mechanisms:
- W3C EME
Encrypted Media Extension (EME)

- **EME**: A standard defined by the World Wide Web Consortium (W3C)
- An API to make DRM use in browsers more uniform
- Integrated into all major browsers
- An “opaque” specification
Contribution

- **Our goal:** formally study the security of EME instantiated by Widevine
- **Reverse engineer** the proprietary Widevine protocol
- Define **security properties** (not present in EME spec)
- Model and prove in **Tamarin**
EME Specification

- Standard defines EME workflow and messages:
  - Initiate session
  - Initial Licence request/response
  - Licence Renewal request/response
  - Using a licence to decrypt

- Does not specify the content of messages or internal behaviour of the CDM → they are proprietary and implementation-specific
### Reverse engineering of Widevine EME messages – Initial Licence

<table>
<thead>
<tr>
<th>EME Message</th>
<th>Widevine EME Message Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Licence Request</td>
<td>(\langle \text{reqID}, \text{nonce}, {\text{clientID}}<em>\text{privacyK}, {\text{privacyK}}</em>\text{serviceCert}, \text{keyID}, t_1 \rangle = \text{req} + \text{signature of req with deviceK} )</td>
</tr>
<tr>
<td>Licence Response</td>
<td>({\text{sessionK}}<em>\text{deviceK}, \langle \text{reqID}, t_1, \Delta t, \text{keyID}, {\text{contentK}}</em>\text{assetK}, {\text{nonce}}_\text{contentK}, \text{policy} \rangle = \text{resp} + \text{MAC of resp with macK}_S )</td>
</tr>
</tbody>
</table>

- **Hierarchy of keys:**

  - deviceK \(\rightarrow\) sessionK \(\rightarrow\) assetK, macK_S, macK_C \(\rightarrow\) contentK
  - assetK, macK_S, macK_C = \(KDF(tags, req, sessionK)\)

- **nonce** to ensure freshness
- **Timestamps** and **time-to-live** to control licence expiration
Reverse engineering of Widevine EME messages – Licence Renewal

<table>
<thead>
<tr>
<th>EME Message</th>
<th>Widevine EME Message Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Renewal Request</td>
<td>⟨reqID, {clientID} _privacyK', {privacyK'} _serviceCert, t_1, t_2, ctr, nonce'⟩ + MAC with macK_C</td>
</tr>
<tr>
<td>Renewal Response</td>
<td>⟨reqID, t_1, t_2, ctr + 1, policy', Δt, {nonce'} _contentK⟩ + MAC with macK_S</td>
</tr>
</tbody>
</table>

- Counter `ctr` to ensure correspondence between request/response
- Two slightly different versions:
  - `nonce'` in renewal is present on Android, but absent on desktop
Specifications are not public, EME gives no security guarantees. No standard security definitions for such DRM systems 😱
Specifications are **not public**, EME gives **no security guarantees**
No standard security definitions for such DRM systems 😞
⇒ We propose our own definitions for the security of Widevine 😊

**Attacker scenario:**
trusted CDM and OTT, untrusted network and API user (browser/client)
Specifications are not public, EME gives no security guarantees
No standard security definitions for such DRM systems 😞
⇒ We propose our own definitions for the security of Widevine 😊

Attacker scenario:
trusted CDM and OTT, untrusted network and API user (browser/client)

We introduce seven security goals, split into three groups
Specifications are not public, EME gives no security guarantees
No standard security definitions for such DRM systems 😨
⇒ We propose our own definitions for the security of Widevine 😏

Attacker scenario:
trusted CDM and OTT, untrusted network and API user (browser/client)

We introduce seven security goals, split into three groups

Security Goal 1: Key Confidentiality
Content decryption keys remain secret.
Security goals for initial licences

Security Goal 2: Integrity
The CDM must load initial licence responses as they were generated by the OTT.

Security Goal 3: Freshness
A given licence response can be loaded at most once, and only by the CDM generating the corresponding request.

Security Goal 4: Expiration
In the initial phase, the CDM can use a decryption key at time $t$ only if the OTT granted a licence for it expiring at time $t_0 + \Delta t \geq t$. 
Security goals for licence renewal

**Security Goal 5: Integrity**
The CDM must load renewal responses as they were generated by the OTT.

**Security Goal 6: Freshness**
A given renewal response can be loaded at most once, and only by the right CDM, and a CDM loads at most one response per renewal event.

**Security Goal 7: Expiration**
In the renewal phase, the CDM can use a decryption key at time $t$ only if the OTT granted a (renewed) licence for it expiring at time $t_0 + \Delta t \geq t$. 
Formal analysis of Widevine/EME

- We analyse **Widevine/EME** using the **Tamarin** prover (both Android and desktop version)

- A fairly **complex** protocol to model:
  - two roles (CDM & Licence Server), unbounded sessions
  - a stateful API
  - with **loops**
  - and **counters**
  - and **timers**...
Modelling of time in Tamarin

- Widevine messages contain timestamps, and Goals 4 and 7 explicitly mention time.
- No built-in support for time-based properties in Tamarin.

\[ t \] each protocol rule receives it: \( GTime(t) \).

It can appear in protocol messages and lemmas: \( GTime(t_1 @ i & State(t_2, \ldots)) @ i \Rightarrow t_1 << t_2 \).

Attacker chooses the time each rule is executed, a restriction forces consistent choices:

\[ #i < #j & GTime(t_1 @ i) & GTime(t_2 @ j) \Rightarrow t_1 << t_2 \]
Modelling of time in Tamarin

- Widevine messages contain *timestamps*, and Goals 4 and 7 explicitly mention *time*
- *No built-in* support for time-based properties in Tamarin
- We propose *our own encoding* of time
  - Time as an integer \( t \)
  - Each *protocol rule* receives it \( \text{ln}(t) \), has event \( \text{GTime}(t) \)
  - It can appear in *protocol messages and lemmas*:
    
    \[
    \text{GTime}(t_1)@i \land \text{State}(t_2, \ldots)@i \Rightarrow t_1 < t_2 \ldots
    \]
  - *Attacker* chooses the time each rule is executed, a *restriction* forces consistent choices:
    
    \[
    \#i < \#j \land \text{GTime}(t_1)@i \land \text{GTime}(t_2)@j \Rightarrow t_1 < t_2
    \]
Conclusion & future work

- We reverse-engineered the Widevine DRM protocol
- We propose definitions for the security of Widevine as an EME instance
- We formally analyse the protocol in Tamarin

Future work

- Privacy properties
- Model for dishonest Licence server
- Other DRM systems
Questions?