Seriously Secure WebRTC

Danilo Bargen (@dbrgn), Threema GmbH

26. September 2016

Coredump Rapperswil

WebRTC?

Was ist WebRTC?

> *«WebRTC is a free, open project that provides browsers and mobile applications with Real-Time Communications (RTC) capabilities via simple APIs.»*
>
> — webrtc.org

WebRTC ermöglicht Rechner-zu-Rechner-Kommunikation in Echtzeit.

Use Cases

Mögliche Anwendungsfälle:

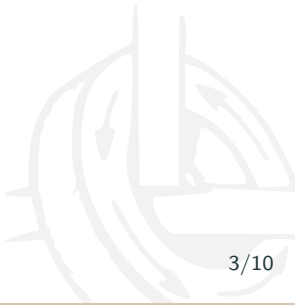
- Videokonferenzen
- Dateitransfer
- Chat
- Desktopsharing
- ...

Coole Demo: `https://appear.in/`

Wieso WebRTC?

WebRTC löst folgende Probleme der Echtzeit-P2P-Komunikation:

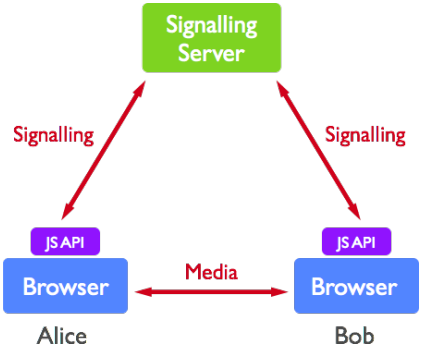
- Einheitliches Protokoll
- Aufbauen einer Netzwerkverbindung (NAT Traversal etc)
- Einheitliche APIs
- Lizenzfreie Codecs
- Integration in Web Browser



Serverless?

WebRTC bietet Peer-to-Peer-Kommunikation, ist aber nicht ganz unabängig von Servern.

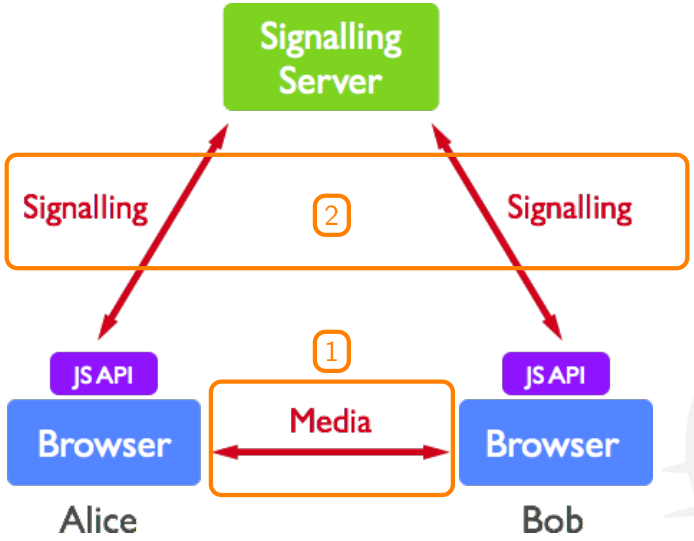
Signaling-Server wird benötigt um Netzwerkverbindung aufzubauen Dieser übernimmt IP-Detection / NAT-Traversal (STUN), und falls nötig Relaying (TURN).



Source: https://webrtc-security.github.io/

Security?

Angriffsvektoren: Daten (1) und Signalisierung (2).



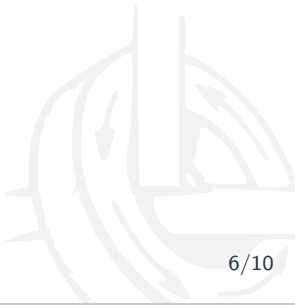
Security: Daten (1)

WebRTC verschlüsselt Data Streams mit DTLS und Media Streams mit SRTP.

Probleme:

- SRTP verschlüsselt nur Inhalte, nicht jedoch Header, darin enthalten ist z.B. die Voice-Lautstärke
- Beide Protokolle basieren auf dem TLS Trust-Modell
- TLS Parameter sind nicht konfigurierbar
- ...

Eigene End-zu-End-Verschlüsselung wäre wünschenswert.



Security: Signaling (2)

Signalisierung läuft über einen Server. Das Protokoll ist von WebRTC nicht spezifiziert.

Probleme:

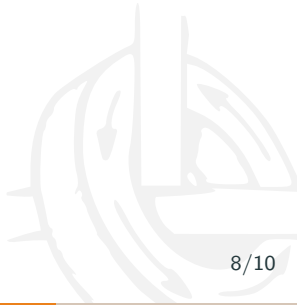
- Standard-Implementierung in-the-wild ist mit Websockets, häufig unverschlüsselt
- Signalisierung enthält sensitive Netzwerkinformationen und andere Metadaten
- Secure Websockets (wss) basieren auf dem TLS Trust-Modell und bieten nur Transportverschlüsselung
- ...

Ein end-zu-end verschlüsseltes Signaling-Protokoll, bei dem man dem Server nicht vertrauen muss, wäre wünschenswert.

SaltyRTC

SaltyRTC will diese Probleme lösen.

- Ende-zu-Ende verschlüsseltes Signalisierungs-Protokoll
- Authentisierte Verschlüsselung basierend auf dem NaCl Box-Modell
- Sicherheit auch mit kaputtem TLS
- Kein Server-Trust notwendig



SaltyRTC: Status

Aktueller Status:

- Protokoll ist beinahe fertig spezifiziert
- Server-Implementierung in Python
- Client-Implementierungen in TypeScript und Java
- Grundlage für Threema Webclient

Links & Thanks

SaltyRTC ist Open Source Software:

https://github.com/saltyrtc/

Danke an:

- Lennart Grahl (Protokoll-Entwicklung und Server-Implementation)
- Threema GmbH (Finanzierung der Entwicklung)

Thank you!

www.coredump.ch