Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Research Report

Max Jonas Werner

WebRTC Security in the context of a DHT implementation

Fakultät Technik und Informatik
Studiendepartment Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Max Jonas Werner

WebRTC Security in the context of a DHT implementation

Research Report submitted in the context of Seminar MINF-AW2

in the course Master of Science
at the Department of Computer Science
at the Faculty of Engineering and Computer Science
of Hamburg University of Applied Sciences

Submitted on: 28 July 2013

Max Jonas Werner

Title of the paper
WebRTC Security in the context of a DHT implementation

Keywords
Peer-to-peer, Web, Web Browser, Chord, DHT, JavaScript, DOM, HTML5, WebRTC, Security, Privacy, Encryption

Abstract
The WebRTC protocols and interfaces – as standardized jointly by the IETF and the W3C – permit Web applications to open data channel connections between browers. Such connections can be used to implement a Distributed Hash Table (DHT) running in the browser. This runtime environment however poses various security challenges that are analyzed in this paper.

Contents

1 Introduction

The WebRTC specification consists of an API definition (Bergkvist u. a., 2012) specified by the World Wide Web Consortium (W3C) and a set of underlying protocols specified by the Internet Engineering Task Force's (IETF) Rtcweb Working Group (Alvestrand, 2011). The goal of WebRTC is to enable web browsers to open direct peer-to-peer communication channels between each other, as outlined in Figure 1.1. The security aspects of WebRTC are specifically identified in Rescorla (2013b) and further outlined in Rescorla (2013a).

This paper analyzes the definitions found in these two specifications with a focus on applicability of WebRTC as a transport layer for a Distributed Hash Table (DHT) implementation.

1.1 Safety Objectives

My analysis of WebRTC security and impacts on a DHT implementation is based on a set of safety objectives generally known as the CIA triad. This acronym denotes the three objectives Confidentiality, Integrity and Availability. Since availability is an objective that fits more in the context of IT system evaluation I modified the CIA acronym so that the "A" stands for Authenticity which is a major concern in the design of a secure P2P application. The semantics of these three goals explained below are taken from Shirey (2007).

Confidentiality

This objective ensures that data which is transferred between two trusted peers does not reach an unauthorized third actor. An example of the need of confidentiality is the submission of credit card information from a buyer to a merchant. In an internet system the credit card number, name of the holder, expiration date, possibly a CVC check number and additional information usually pass a number of routers that lie between the buyer and the merchant. To make sure that no one on this way (e.g. with access to the intermediate routers) is able to read out the sensitive data it is encrypted and thus confidentiality is put in place.

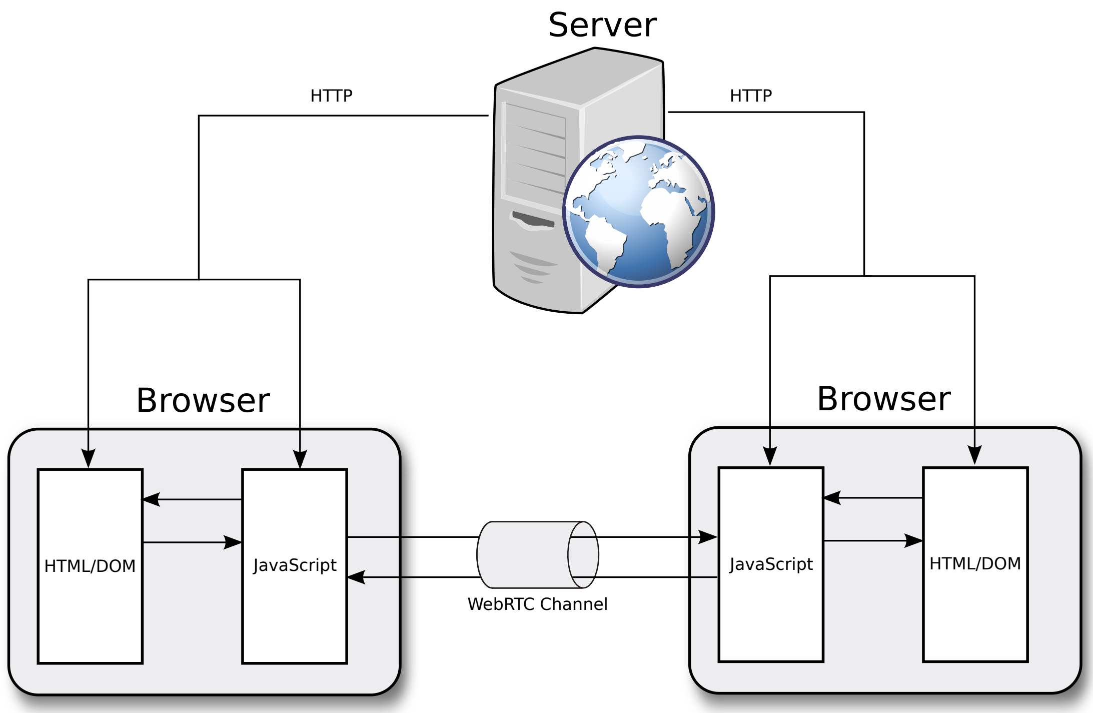


Figure 1.1: Schematic view of a WebRTC connection. The server delivers applications (consisting of HTML, CSS and JavaScript code as well as resources such as images) to all the client browsers. Additionally in this scenario the server handles the signaling of a WebRTC connection, therefore serving as a central connection establishment entity. When the signaling is done the WebRTC channel is established and no more signaling is required.

Integrity

Data integrity in an IT infrastructure makes sure that data originating from one node is not modified on the way to the receiver or the receiver is enabled to check whether data has been modified. Referring to the credit card example above this means that no person with access to intermediate routers can inject false credit card information. Since (deliberate or unintentional) modification of data in IP networks cannot be avoided one has to make sure that modifications are detected by the peers. In this scenario the integrity of the data en route is ensured using message authentication codes (MAC) and cryptographic signatures.

Additionally, integrity of data refers to the unmodifiability of data stored in an IT system. This can also be ensured using cryptographic signatures or checksums by using hash algorithms like MD5, SHA or RIPEMD and comparing the results. This is typically done when downloading files from a server.

Authenticity

The objective of authenticity implies the process of verifying the claim that data coming from a certain origin actually originated there. This is important in various contexts: A web server sending HTML to a browser via HTTPS authenticates itself using a TLS certificate that is cryptographically signed by a trusted certificate authority. On the other side a user authenticates to a server using her credentials (e.g. user name and password or TLS client certificate). More generally authentication may be applied by something a peer knows (e.g. a password), owns (e.g. a key card) or is (e.g. a biometric attribute).

Again referring to the example of a customer buying goods in an online shopping system the client authenticates itself to the shop provider by logging into the system. This ensures that the purchase can securely be tracked back to the customer.

There are, however, use cases where authenticity – in the sense of being able to track information down to a real person – is explicitly not desired. These include anonymous/pseudonymous conversations between two peers to maintain a certain level of privacy.

2 WebRTC: Proposed Security Mechanisms

The IETF Rtcweb WG is working on two Internet Drafts that specify the security requirements and implications of WebRTC implementations. Rescorla (2013b) introduces a general overview of threats that WebRTC implementors and standards bodies will have to deal with. The document – that is currently under development – explains the difference between the WebRTC threat model compared to threats that classic VoIP systems have to face. This includes problems that come with the nature of web applications in which the user is possibly connected to malicious servers that may use the WebRTC services in the browser against a user as well as cross-origin security where code from a malicious site is injected into insecure non-malicious sites. Rescorla (2013a) explains how the threats from the previous document are countered by the WebRTC protocols and API.

In the next sections I will analyze WebRTC security as proposed by the two mentioned specification documents. Section 2.1 goes into detail about the security needs for each of the components involved in a WebRTC conversation whereas Section 2.2 further analyzes how WebRTC deals with identities in different scenarios thus focussing especially on authenticity and anonymity/pseudonymity.

2.1 Component Security

The different components of a WebRTC conversation, namely the server, the browser, the path between server and browser as well as the path between two browsers all have to deal with different threats that the next sections explain in detail.

2.1.1 Server

The server as outlined in Figure 1.1 has two purposes:

1. Deliver the application that makes use of the WebRTC JavaScript API
2. Enable signaling between two browsers

Technically these two purposes may be handled by two different servers but since this separation does not have an impact on security considerations I will further assume a single-server scenario.

Delivering of the application is done in the classical way via HTTP or HTTPS and thus the same security considerations that apply to every web application delivery also apply here. Therefore these are not WebRTC-specific and not further investigated. Implications derived from the fact that using a central server may expose certain meta data about the users – which can be a threat to privacy – are outlined in Section 2.2. One important aspect to consider when deploying the server is that delivery of the application can be a first entry for attackers e.g. when the transfer is conducted via unencrypted HTTP. If this is the case, a man in the middle may be able to introduce malicious code to the user and perhaps fake a WebRTC connection. Thus application transfer should always be conducted using HTTPS with proper server certificates.

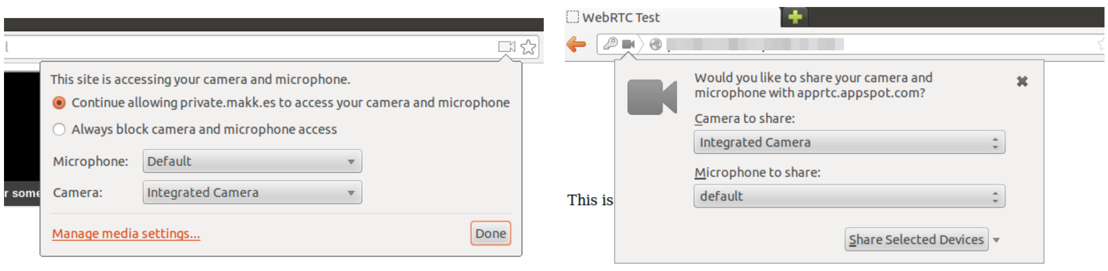
2.1.2 Browser

The browser acts as the runtime environment of all WebRTC code and therefore is a critical component in the WebRTC infrastructure. To enable use cases such as a audio/video conferencing the WebRTC application must be granted access to a microphone and/or camera attached to the user's computer. Additionally it may be possible that random users start connecting to the client's browser and other peers may get access to the client's IP address which poses threats to location privacy. Since any web application (malicious or not) a user points her browser to may be able to initiate WebRTC connections, the browser must also restrict applications' access to certain information. Rescorla (2013a) mentions several countermeasures to these threats which are elaborated below.

User Consent

The first countermeasure to the threat that arbitrary web applications may gain access to the user's camera/microphone and consequently transfer the captured image/sound stream to a malicious node is to ask the user for consent every time an applications indicates the desire to access one of these resources. The specification states that "allowing arbitrary sites to initiate calls violates the core Web security guarantee; without some access restrictions on local devices, any malicious site could simply bug a user." (Rescorla, 2013b)

Browsers are further obliged by the specification to "obtain explicit user consent prior to providing access to the camera and/or microphone". Figure 2.1 shows two examples of implementations of such a consent prompt; one for Chrome and one for Firefox.


This site is accessing your camera and microphone.

[✓ Continue allowing private.makk.es to access your camera and microphone]

Always block camera and microphone access

Microphone: Default

Camera: Integrated Camera

Manage media settings...

Done

WebRTC Test

Would you like to share your camera and microphone with apprtc.appspot.com?

Camera to share: Integrated Camera

Microphone to share: default

Share Selected Devices

This is

Figure 2.1: When a web application indicates the desire to access the computer's camera and/or microphone, browser's must aks the user for consent. The consent prompts shown here are those of Chrome (left) and Firefox (right).

Additional to asking for user's agreement to access hardware resources she must have the means of determining that a call is in progress. This is achieved by current implementations by the indicators show in Figure 2.2. A topic of recent discussion among browser developers is that of usability and general user experience with these consent mechanisms on mobile devices. There currently seems to be no final agreement on best practices e.g. what the browser should do when it is sent to the background on a mobile phone (options discussed right now are to completely stop sending data or to indicate hardware access in a notification area of the phone).

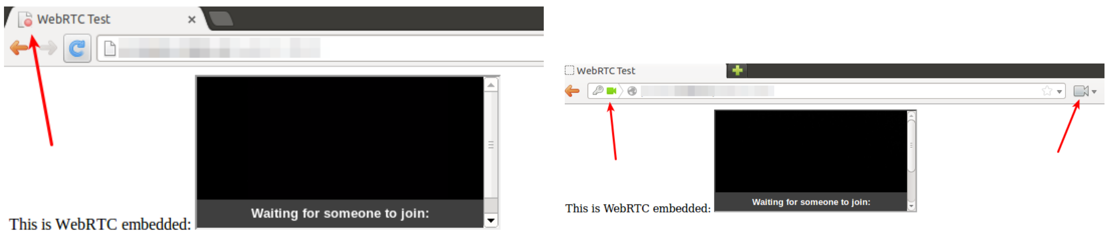


Figure 2.2: Once a WebRTC application has gained access to the camera/microphone browsers show an indicator of a call in progress so that the user knows of a possible image/sound transfer to another peer. Chrome (left) puts this indicator on the top of the tab while Firefox (right) shows it in the address bar as well as permanently right to the address bar so that even when the user is in another tab she knows of the call in progress.

Arbitrary incoming connections

Currently a consent is granted for the whole site and not individual incoming calls. This may be convenient for the user but poses the threat that an arbitrary user uses the calling service to call any other user currently connected to the service. One way to handle this would be that the application code asks for consent when an incoming call is to be accepted. In a VoIP application for example the user may be prompted by the application that another user is calling. Since this is left to the application there may be applications that leave users open to being tapped by other users. On the other hand when implementing a DHT it is even desirable that connections may be opened without the user's consent.

Location privacy

Rescorla (2013b) and Rescorla (2013a) both contain a dedicated section dealing with location privacy concerns. These arise when negotiating Interactive Connectivity Establishment (ICE) parameters between two browsers prior to establishing a WebRTC connection which may leave the user open to revealing her IP address to another peer without her having ways to suppress this. The specification documents therefore mandate implementations to supply JavaScript applications with a means to suppress ICE negotiation until the user has explicitly granted the connection initialization. Guaranteeing location privacy hence is a task left up to any individual application.

2.1.3 Signaling Path

The act of signaling a WebRTC connection also is commonly achieved using known transport mechanisms, namely HTTP(S) or WebSockets. Alternatively signaling may be conducted via another channel – independent of the application server – e.g. using XMPP; the WebRTC specification does not dictate any specific transport for signaling. The standard case is that the application uses HTTPS or encrypted WebSockets (WSS) which then inherits the security considerations immanent in the protocol used. In the case of HTTPS or WSS server authentication (via SSL/TLS certificates), confidentiality (via encryption) and data integrity are guaranteed.

One potential privacy threat with using a central server for signaling is that the server provider gets to know who communicates with whom and when. A possible approach to mitigate this threat can be to signal connections in-band via existing WebRTC connections; this, though, implies that peers are able to reach one another via some sort of WebRTC mesh network and is anything but trivial to implement. In a DHT implementation, though, this may be a useful countermeasure.

2.1.4 Media Path

The media path between two browsers is at the core of the WebRTC specification. Figure 2.3 outlines the different protocols being used for audio/video and generic data transfer. The specification mandates that the "Extended Secure RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/SAVPF) [RFC5124] as extended by [I-D.ietf-avtcore-avp-codecs] MUST be implemented" for A/V data (Perkins u. a., 2013). For exchanging encryption keys browsers "MUST support DTLS-SRTP [RFC5764] for key-management" (Perkins u. a., 2013). This ensures that users' A/V data is encrypted end-to-end without even leaving a signaling server with the possibility to decrypt the traffic. There's ongoing passionate discussion about mandating browsers to also implement SDES (Andreasen u. a., 2006) which would make it easier for current media gateways to serve WebRTC end points. Since SDES would not ensure that the signaling server cannot decrypt the SRTP traffic this would probably pose privacy threats. A final decision in the working group has not been made, yet.

For Data Channels the specification states that the "encapsulation of SCTP over DTLS (see [I-D.ietf-tsvwg-sctp-dtls-encaps]) over ICE/UDP (see [RFC5245]) provides a NAT traversal solution together with confidentiality, source authentication, and integrity protected transfers" (Jesup u. a., 2013).

Both A/V data as well as Data Channel traffic cannot be sent unencrypted between WebRTC peers which meets the safety objectives stated in Section 1.1.

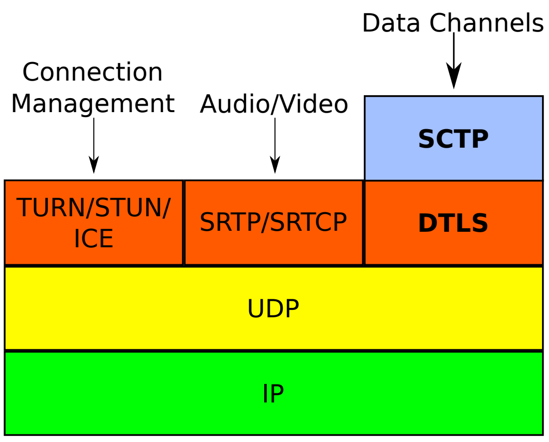


Figure 2.3: The protocol stack of WebRTC is divided into a connection management component for establishing and maintaining connections, an A/V component and a Data Channel component. Both audio/video data as well as Data Channel streams are encrypted end-to-end using SRTP/SRTCP and DTLS, respectively.

2.2 Identity

Rescorla (2013a) deals very specifically with handling user identities. With regards to the authenticity safety objective mentioned in Section 1.1 there are two general concepts available to deal with identities:

1. Anonymity/Pseudonymity
2. Identity Providers

Anonymity/pseudonymity are useful when it is not desirable or necessary to identify a peer. An example mentioned in the security architecture specification is that of a "click to call support" button on a company's website. Here, the company's call center agent must not necessarily know the real identity of the caller only to deal with general product questions.

Identity providers are outlined in Rescorla (2013a) as third entities that mutually ensure users' identities so that every user can be guaranteed that the identity she claims to obtain is proved by a trusted third party. This implies, of course, that user A trusts the identity provider of user B to securely prove her identity.

In detail, an application may ask an identity provider to generate a cryptographically secured identity assertion that is then carried over the signaling layer together with the Offer/Answer packages needed for connection establishment. Such an assertion is then extracted from the package by the peer on the other side and sent to the identity provider for validation.

A current concern with the identity provider approach laid out in the specification is that no browser has implemented even parts of that mechanism. Thus, usable implementations may come to users rather late in the WebRTC rollout process.

3 Conclusion/Outlook

Currently not all security measurements specified in Rescorla (2013b) and Rescorla (2013a) are implemented in the browsers (e.g. Identity Provider support) and some problems are unsolved (e.g. the problem of letting the user know that a web application is accessing the camera on a mobile phone when the browser is sent to the background).

From a security point of view, though, even current implementations of WebRTC provide a very solid transport mechanism for running a Distributed Hash Table application. An implementor can rely on a solid foundation that's needed for securing peer communication by WebRTC media channel encryption (which would otherwise have to be implemented in the DHT application which is probably more prone to bugs).

The specification of an identity assertion may additionally be useful to identify and/or exclude malicious nodes from the DHT and probably to detect e.g. sybil attacks (Steinmetz und Wehrle, 2005). Still, though, the general security problems of DHTs also exist with WebRTC as transport. The DHT implementation must actively deal with malicious nodes (probably asserting identities using WebRTC mechanisms) and force some sort of trust or reputation system for nodes and ensure the usage of secure identifiers.

Bibliography

[Alvestrand 2011] Alvestrand, Harald: Overview: Real Time Protocols for Brower-based Applications / IETF. June 2011 (01). – Internet-Draft – work in progress

[Andreasen u. a. 2006] Andreasen, F. ; Baugher, M. ; Wing, D.: Session Description Protocol (SDP) Security Descriptions for Media Streams / IETF. July 2006 (4568). – RFC

[Bergkvist u. a. 2012] Bergkvist, Adam ; Burnett, Daniel C. ; Jennings, Cullen ; Narayanan, Anant: WebRTC 1.0: Real-time Communication Between Browsers / W3C. URL http://www.w3.org/TR/2012/WD-webrtc-20120209/, Oktober 2012. – W3C Working Draft

[Jesup u. a. 2013] Jesup, Randell ; Loreto, Salvatore ; Tuexen, Michael: RTCWeb Data Channels / IETF. February 2013 (04). – Internet-Draft – work in progress

[Perkins u. a. 2013] Perkins, Colin ; Westerlund, Magnus ; Ott, Joerg: Web Real-Time Communication (WebRTC): Media Transport and Use of RTP / IETF. February 2013 (06). – Internet-Draft – work in progress

[Rescorla 2013a] Rescorla, Eric: RTCWEB Security Architecture / IETF. January 2013 (06). – Internet-Draft – work in progress

[Rescorla 2013b] Rescorla, Eric: Security Considerations for RTC-Web / IETF. January 2013 (04). – Internet-Draft – work in progress

[Shirey 2007] Shirey, R.: Internet Security Glossary, Version 2 / IETF. August 2007 (4949). – RFC

[Steinmetz und Wehrle 2005] Steinmetz, Ralf (Hrsg.) ; Wehrle, Klaus (Hrsg.): *LNCS*. Bd. 3485: *Peer-to-Peer Systems and Applications*. Berlin Heidelberg : Springer-Verlag, 2005