Network Working Group
Internet-Draft
Expires: April 16, 2012

W. Beck
Deutsche Telekom
October 14, 2011

Real-Time Web Communication Simplified Interconnection
draft-beck-rtcweb-alt-ic-00

Abstract

The Real-Time Communication Web (RTCWEB) approach uses browser-based scripts to minimize the number of network protocols between server and browser that need standardization. With traditional interconnection concepts, the success of this effort is limited as different RTCWEB applications still need to interoperate. This document proposes an alternative interconnection model where caller and callee always use the same RTCWEB client and server and avoid any interworking this way.

1. Problem Statement

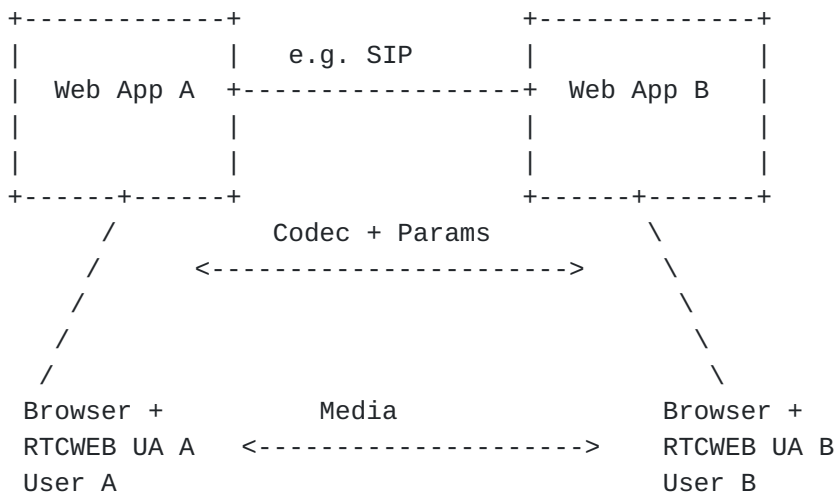

Figure 1

Figure 1 shows the original RTCWEB model for interconnecting users of web application A and web application B. Not only the server applications A and B have to agree on a common protocol, but to some degree also the browser-based RTCWEB clients.

This model requires that the signaling between RTCWEB server and client can be matched to the intra-server protocol. Codec parameters and other information that is only relevant to the clients must be exchanged between them.

The Session Initiation Protocol (SIP, [RFC3261]) and the Session Description Protocol (SDP, [RFC4566]) have traditionally been used to perform those functions. Both protocols suffer from the complexity that is the result of the combined number of requirements of all possible use cases. Re-using one or both of these protocols in RTCWEB clients offers little advantage over classical SIP clients.

The key benefit of RTCWEB is the possibility to tailor the client-server protocol to the actual use case. The software has only to support those parameters and protocol states that are required in a particular application. However, much of this advantage is lost when two different RTCWEB applications have to interoperate. The applications may have a different audiences and may have only a small subset of common functionalities. An example is the call from an RTCWEB application designed for mobile users to an RTCWEB application designed for high quality conferencing. The two applications will have very different requirements regarding the codec choice and call

control features.

The perceived requirement to have interoperable RTCWEB applications is rooted in the traditional way of interconnecting networks.

In traditional phone networks, the 'application providers' were also owners of the physical links that connected them. The architecture shown in Figure 1 was a technological necessity. The associated trust relationships were a welcome side effect as other ways of authentication and authorization were not feasible.

- The originating side (Web App A) authenticates and authorizes the user to make call. It trusts Web App B to forward the call to the correct user.

- The terminating side (Web App B, the application receiving a call) only authorizes the originating side as a whole, not its individual users. It trusts Web App A to authenticate the calling user.

- It is possible to have 'transit' entities between the originating and terminating side. The involved parties form a chain of trust.

Even with standardized protocols, this models often requires detailed agreements between interconnection partners to even out implementation differences.

2. Third Party Authentication

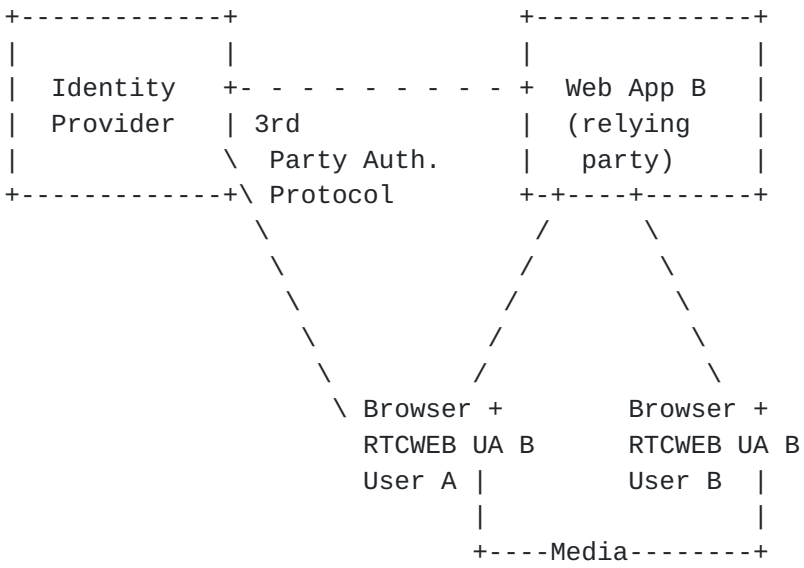

Figure 2

Figure 2 shows an alternative interconnection model:

- User A and user B use the same RTCWEB application and browser-based client (B). RTCWEB makes this very easy as user A can run the client instantly. As Web App B handles the call on its own, it does not need to interoperate with Web App A for call control.

- User A presents some 3rd party identity assertion to RTCWEB application B. RTCWEB application B can check this assertion either by asking the 3rd party, or by checking its cryptographic signature. Based on the issuer of the authentication assertion, application B can grant or restrict its functionality to user A.

The key idea is to separate the trust relationship from call signaling. Only the media stream between the two browsers needs to be standardized.

With this model the terminating side is always in charge of the whole call. The originating side takes only the role of an identity provider.

The following sections give an overview over some existing third party authentication methods that can be used with RTCWEB.

3. Client-Side Certificates

The generation of client certificates in the browser has been standardized with HTML 5.

- When a user signs up to the RTCWEB service, the server will generate a client certificate and install it in the browser. To do this, it uses an HTML form containing a <keygen> element. The browser will then generate a public key and send it to the server. The server will sign it and offer the resulting certificate back to the user for download.

- When a user makes a call, the called party's RTCWEB service will check the client certificate. If it trusts the issuer of the certificate, the user can make the call.

The main drawback of using client certificates is the poor user interface in current browsers. There have been attempts to improve this by implementing TLS in Javascript and to store certificates using a widely used proprietary browser plugin. This solution is not a suitable base for standardization.

With client certificates, the originating side has no control over the call at all. It does not even know that a call took place and can only distribute a certificate revocation list to restrict a user's actions.

To avoid the cost associated with properly authenticated client certificates, the WebID initiative has come up with a new way to verify them: the browser sends a client certificate as part of a https session setup. The certificate contains a URL that can be used for verification. The URL points to a resource which contains a public key. If this public key is identical to the one received in the client certificate, the web application can be sure of its authenticity. This way the user does not require an expensive certificate issued by a universally acknowledged authority.

The web application can choose to reject request carrying certificates issued by institutions it does not trust.

Example from the user's perspective:

1. Alice (https://atlanta.com/rtcweb/alice) wants to call Bob at https://biloxi.com/users/bob. She points her browser to Bob's URL.

2. Alice's browser opens a certificate selection box: 'The site has requested that you identify yourself with a certificate'. She

selects the one she wants to use for this call.

3. Alice's browser loads the RTCWEB client from biloxi.com and connects her to Bob.

4. OAuth

With OAuth an application can ask a user for access to a protected resource owned by the user. The OAuth roles can be mapped to RTCWEB interconnection roles:

- The terminating RTCWEB application is the OAuth client.
- The originating RTCWEB application provides the authorization and resource servers.
- The calling user is the resource owner.
- The authorization scope is the URL of the called user.
- The protected resource is information about the actions the caller's provider has granted to the calling user. To protect the caller's privacy, this could be restricted to a simple yes or no as a response to an access attempt carrying a specific authorization scope parameter.

When using OAuth, the calling user has to confirm the the access of the called RTCWEB provider to her own RTCWEB provider.

OAuth gives the originating RTCWEB application some control over its users' actions.

Example from the user's perspective:

1. Alice has logged into her RTCWEB application at atlanta.com.
2. Alice (https://atlanta.com/rtcweb/alice) wants to call Bob at https://biloxi.com/users/bob. She points her browser to Bob's URL.
3. biloxi.com redirects her to an OAuth dialog at atlanta.com, which recognizes her as logged in, eg by checking an HTTP cookie. It shows a dialog: 'biloxi.com wants to make a call to https://biloxi.com/users/bob for you, Alice. Allow / Don't Allow'
4. Alice allows the call and gets redirected back again. Her browser loads the RTCWEB client from biloxi.com and connects her to Bob.

5. OpenID

OpenID [OpenID] is an HTTP-based single sign-on system where a request to a relying party web site carries an authentication assertion which can be checked with an OpenID provider. If a request does not contain an authentication assertion, the relying party redirects the browser to the user's OpenID provider and adds a callback URL. Upon successful authentication, the identity provider uses the callback URL to redirect the user's browser back to the relying party.

In an RTCWEB scenario, the originating side would be the OpenID provider and the terminating side would be the relying party.

The originating side has no straightforward way to actually authorize a request. The openid.realm parameter and the return_to URL are the only hints about the nature of the service for which the authentication is intended.

Example from the user's perspective:

1. Alice (https://atlanta.com/rtcweb/alice) wants to call Bob at https://biloxi.com/users/bob. She points her browser to Bob's URL.

2. biloxi.com asks for her OpenID login; Alice enters it and her browser gets redirected to atlanta.com.

3. atlanta.com asks Alice for her password.

4. Alice enters her password. Her browser gets redirected back to biloxi.com and connects her with Bob.

6. Security Considerations

This document proposes to re-use existing third party authentication and authorization standards for RTCWEB. Their security is discussed in the respective standards documents. [TBD]

7. References

[RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.

[RFC4566] Handley, M., Jacobson, V., and C. Perkins, "SDP: Session Description Protocol", RFC 4566, July 2006.

[I-D.ietf-oauth-v2] Hammer-Lahav, E., Recordon, D., and D. Hardt, "The OAuth 2.0 Authorization Protocol", draft-ietf-oauth-v2-22 (work in progress), September 2011.

[OpenID] Openid.net, "OpenID Authentication 2.0 - Final", December 2007, <http://openid.net/specs/openid-authentication-2_0.html>.

Author's Address

Wolfgang Beck
Deutsche Telekom AG
Heinrich Hertz Strasse 3-7
Darmstadt
Germany

Email: beckw@telekom.de