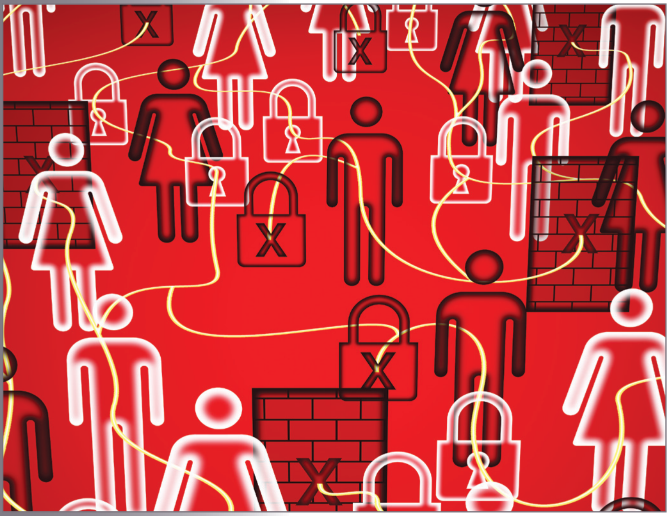

Real-Time Communications Security on the Web

Lieven Desmet
University of Leuven

Martin Johns
SAP AG

Web Real-Time Communications (WebRTC) brings an extremely powerful technology close to users while opening up the path for innovative adoptions within the communications landscape. Key to its success will be how we secure Web RTC at both the network and application layers.

To make a WebRTC connection, a user typically visits a service provider's website, or *calling site*. By calling the appropriate JavaScript APIs in the user's browser, the calling page can set up the WebRTC connection with a remote party.

WebRTC uses two distinct paths for communication: the *signaling path* goes from the call initiator to the call receiver via the existing HTTP(S) connections to the calling sites; the *media path* is set up in a peer-to-peer fashion between the call initiator and the receiver using network protocols such as Session Traversal Utilities for NAT (Network Address Translation) and Datagram Transport Layer Security-Secure Real-Time Transport Protocol (DTLS-SRTP) for RTC. For identity and authentication, both communicating parties employ an *identity provider* (IdP).

This considerable flexibility in deployment is simultaneously one of the most exciting and most challenging aspects of WebRTC technology. As several articles in this issue illustrate, we can apply WebRTC in multiple deployment settings, ranging from a simple scenario with one user contacting the helpdesk from a single calling site to more complex scenarios with multiple calling sites, multiple identity providers, and media gateway services working together to set up video conferencing facilities. WebRTC specifications are deliberately open to allow multiple scenarios by explicitly decoupling the signaling and media planes, calling pages, and IdPs. However, the variety in trust models also means that no single security solution will fit all.

From a security viewpoint, the combination of peer-to-peer RTC (for example, across multiple firewalls and NAT boxes) as well as the availability of a rich set of JavaScript APIs (such as video capture and streaming) makes this an interesting target for a broad set of attacks. Adversaries can, for instance, target the communication on the network, abuse the JavaScript APIs, hijack the service provider's website, or create identity confusion — to name only a subset of potential resulting attack scenarios.

Furthermore, WebRTC is the first browser-based technology that breaks with the Web's strict client-server architecture by enabling direct, serverless browser-to-browser communication. This has significant security consequences because the Web's primary security policy (same-origin) is based on assurance of server identity using the global Secure Sockets Layer-Public-Key Infrastructure (SSL-PKI). For users, however, no established method exists to ensure peer-to-peer authenticity and user identity — a fact that's reflected in this issue's articles.

In This Issue

This issue's theme focuses on the provisioned security characteristics and remaining challenges of emerging WebRTC technology. Although it presents only a small selection of security topics within the WebRTC field, it adequately covers the major issues: the overall security architecture, the provisioning of user identities in the browser environment, and securing media gateway infrastructures.

In "Browser-to-Browser Security Assurances for WebRTC," Richard L. Barnes and Martin Thomson assess end-to-end security assurances based on the WebRTC architectural model. They discuss step by step WebRTC's relevant architectural characteristics, such as the signaling path via the calling sites, peer-to-peer media path, signaling code running in the user's browser, and IdP-based identity provisioning. In their assessment, the authors start by observing that the calling site might not (always) be trusted. In this security setting, they identify a crucial role for browsers as trusted elements in the end-to-end setup. In addition to a set of security improvements for the browser environment, the authors discuss the open problem of securing media that's shown in the browser.

The most classic deployment model for WebRTC is two-party communication via a shared communication provider (that is, calling site), with a separate identity provider used for user identification. In "User Identity for WebRTC Services: A Matter of Trust," Victoria Beltran, Emmanuel Bertin, and Noël Crespi discuss various trust models based on this classic deployment and assess the impact each model has on user privacy. In particular, the article examines the use of existing IdP solutions such as BrowserID, OAuth 2.0, and OpenID Connect in the WebRTC context.

"Who Is Calling Which Page on the Web?" by Li Li, Wu Chou, Zhihong Qiu, and Tao Cai, investigates identity provisioning in more complex deployments where multiple calling sites and IdPs are involved. To support different IdPs, WebRTC uses proxies for interactions between the browser environment and the IdP. The authors propose and compare several architectural approaches on top of these proxies to support identity provisioning, user lookup, and authentication in a uniform way. Moreover, the article suggests how users of Web-of-trust-based identity models could overcome the limitations of hierarchical identity systems.

WebRTC is the first browser-based technology that breaks with the Web's client-server architecture by enabling direct browser-to-browser communication.

To exploit WebRTC technology's full potential, we can realize more complex usage scenarios by deploying additional WebRTC media servers. Such servers can, for instance, transcode media, record and archive media streams, or set up group communications. In "Authentication, Authorization, and Accounting in WebRTC PaaS Infrastructures: The Case of Kurento," Luis López-Fernández, Micael Gallego, Boni García, David Fernández-López, and Francisco Javier López set up several experiments to offer these WebRTC media intermediaries as cloud services. The authors discuss how to expand the WebRTC model to accommodate these sophisticated infrastructures. Moreover, they evaluate various authorization schemes for granting users access to media services without exposing their WebRTC identities.

Outlook

In this issue, we intentionally focused on the complex problem of identity provisioning in WebRTC-based applications because it's a major hurdle to the broad adoption of this high-potential technology. However, as mentioned, this is only a subset of emerging security-related topics that require further investigation.

An important aspect in securing the client-side WebRTC environment is the ability to enforce security policies in the JavaScript execution context. For now, the JavaScript code that controls the whole WebRTC communication runs in the user's browser (and executes on his or her behalf); at the same time, it's under the calling page's control and will likely include (and delegate control to) third-party libraries. Put together, this opens an interesting vector for various attacks (including injection attacks) while receiving and interpreting data via the signaling or media path and running code from the calling site and third-party script providers.

Another aspect is the complexity inherent in setting up the peer-to-peer connections between browsers and the combination of two rather distinct worlds, each with its own security model: end-to-end networking and JavaScript/Web. In particular, we expect the Web's origin-based security model to conflict with RTC's connection-based one. For instance, questions arise about how to set up a connection across different JavaScript origins, or how to manage multiple connections and identities from within a single origin and JavaScript execution context.

Moreover, several technical solutions on the identity provisioning side as well as on the JavaScript permission side propose enrolling users in making security-sensitive decisions, such as whether to let the browser set up a call with user@idp (once or permanently), whether to share video streams or desktops, or whether to trust site A in calling external parties. This will undoubtedly expose (some of) the technology's complexity to users, but even more critically, it implies that WebRTC's end-to-end security applications will strongly depend on the judgment calls of novice users, potentially "promoting" such users to the weakest link of WebRTC.

WebRTC's considerable security considerations will keep security-minded professionals occupied for the next few years. Given the complexity and many facets of the underlying communication and application models, this is hardly surprising. The WebRTC case also clearly illustrates how well the Web is maturing: its technological advances open up a completely new area of applications, but (even more importantly) security is no longer an afterthought — it's already under consideration early in the standardization process.

Lieven Desmet is a research manager of secure software within the iMinds-DistriNet research group at the University of Leuven, Belgium. He outlines and implements research strategy on software security within the university, coaches junior researchers in (Web) application security, and participates in dissemination and valorization activities. Desmet received a PhD in computer science from the University of Leuven. He's a board member of the Open Web Application Security Project's Belgium chapter, and program director of the yearly SecAppDev training courses on secure application development. He has been investigating the security of WebRTC as part of the ongoing EU-FP7 project STREWS. Contact him at lieven.desmet@cs.kuleuven.be.

Martin Johns is a research expert in the Security and Trust group within SAP AG, where he leads the Web application security team. His research interests include software security, static analysis, and exploring the Web ecosystem. Johns received a PhD in computer science from the University of Passau. He's a board member of the Open Web Application Security Project's German chapter. Within the EU-FP7 project STREWS, he is leading the roadmapping activities for Web security research in Europe. Contact him at martin.johns@sap.com.