International Journal of Recent Trends in Engineering & Research
ISSN(ONLINE) : 2455-1457
IMPACT FACTOR : 3.344

Restricting Untrusted Browsers' Communication in Web-RTC

Pooja Birajdar[1], Sagar Soni[2], Nandkishor Surashe[3], Vishal Telsang[4]
[1,2,3,4] *Dept. of Information Technology, AISSMS IOIT*

Abstract— Communication between people is an important factor for the development of the masses. From using animals for carrying the message to using air as a medium, communication has evolved into many forms. Sending letters for communication is a passé as people can now talk through web using video conferencing which is much more preferred and convenient. The drawbacks of the traditional system of VoIP which needs to install plugin or application to perform the operation can be overcome with the help of open source technology called Web-RTC. It allows users to directly communicate with each other directly over browser without the need of plugin. However, security of Web-RTC is main concern. Web-RTC provides security mechanisms like encryption, authentication, and authorization to the data being exchanged over network. But any untrusted user can interrupt the system and can directly start the communication. This interruption can be prevented by providing restriction on the untrusted user. The user who has already registered can only start the communication with the system and its users.

Keywords— Authentication, Authorization, Browser, Certification Authority (CA), DTLS, Identity Provider (IdP), JavaScript Signaling, Peer-to-Peer (P2P), SRTP, Web-RTC (Web Real-Time Communication), Peer-to-Peer (P2P).

I. INTRODUCTION

Web-RTC (Web Real-Time Communication) is an Application Programming Interface (API) definition drafted by the World Wide Web Consortium (W3C). Imagine a world where your phone, TV and computer could all communicate on a common platform. Imagine it was easy to add video chat and peer-to-peer data sharing to your web application. That's the vision of Web-RTC. One of the last major challenges for the web is to enable human communication via voice and video: Real Time Communication, RTC for short. RTC should be as natural in a web application as entering text in a text input. Without it, we're limited in our ability to innovate and develop new ways for people to interact. Historically, RTC has been corporate and complex, requiring expensive audio and video technologies to be licensed or developed in house. Integrating RTC technology with existing content, data and services has been difficult and time consuming, particularly on the web. Gmail video chat became popular in 2008, and in 2011 Google introduced Hangouts, which use the Google Talk service (as does Gmail). Google bought GIPS, a company which had developed many components required for RTC, such as codecs and echo cancellation techniques. Google open sourced the technologies developed by GIPS and engaged with relevant standards bodies at the IETF and W3C to ensure industry consensus. In May 2011, Ericsson built the first implementation of Web-RTC. Web-RTC has now implemented open standards for real-time, plugin-free video, audio and data communication. The need is real:

- Many web services already use RTC, but need downloads, native apps or plugins. This includes Skype, Facebook (which uses Skype) and Google Hangouts (which use the Google Talk plugin).
- Downloading, installing and updating plugins can be complex, error prone and annoying.

- Plugins can be difficult to deploy, debug, troubleshoot, test and maintain—and may require licensing and integration with complex, expensive technology. It's often difficult to persuade people to install plugins in the first place!

The guiding principles of the Web-RTC project are that its APIs should be open source, free, standardized, built into web browsers and more efficient than existing technologies.

II. PEER-TO-PEER COMMUNICATION

Web-RTC can be used for multiple tasks, but real-time peer-to-peer audio and video (i.e., multimedia) communications is the primary benefit. In order to communicate with another person (i.e., peer) via a web browser, each person's web browser must agree to begin communication, know how to locate one another, bypass security and firewall protections, and transmit all multimedia communications in real-time.

One of the biggest challenges associated with browser-based peer-to-peer communications is, knowing that how to locate and establish a network socket connection with another computer's web browser in order to bi-directionally transmit multimedia data. When we visit a web site, we typically enter a web address or click a link to view the page. A request is made to a server that responds by providing the web page (HTML, CSS, and JavaScript). The key here is that we make an HTTP request to a known and easily locatable (via DNS) server and get back a response (i.e., the web page).

Now let's say we wanted to have a video chat with our mother. Our mother's computer is not a web server. Therefore, the problem is how do we make the request and actually receive her audio and video data directly, while also sending our audio and video data directly to her, but without going through an external server? Here, the Web-RTC comes into picture.

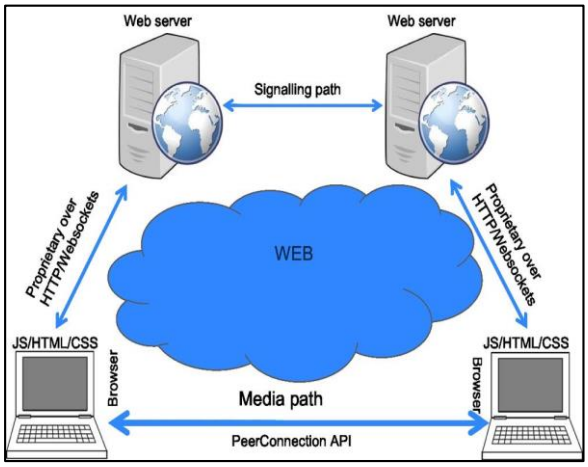

Figure 1: Working of Web-RTC model

III. SIGNALING, SESSIONS AND PROTOCOLS

The network information discovery process is one part of the larger topic of signaling, which is based on the JavaScript Session Establishment Protocol (JSEP) standard in the case of Web-RTC. Signaling involves network discovery and NAT traversal, session creation and management, communication security, media-capability metadata and coordination, and error handling. Signaling is not specified by the Web-RTC standard, nor implemented by its APIs in order to allow flexibility in the technologies and

protocols used. Signaling and the server that handles it is left to the Web-RTC application creator to sort out.

Assuming that the Web-RTC browser-based application is able to determine its public-facing IP address using STUN as described, the next step is to actually negotiate and establish the network session connection with the peer. This process is analogous to making a phone call. The initial session negotiation and establishment happens using a signaling/communication protocol specialized in multimedia communications. This protocol is also responsible for governing the rules by which the session is managed and terminated.

One such protocol is the Session Initiation Protocol (aka SIP). Due to the flexibility of Web-RTC signaling, SIP is not the only signaling protocol that can be used. The signaling protocol chosen must also work with an application layer protocol called the Session Description Protocol (SDP), which is used in the case of Web-RTC. All multimedia-specific metadata is passed using the SDP Protocol. Any peer (i.e., Web-RTC-leveraging application) that is attempting to communicate with another peer generates a set of ICE candidates, where ICE stands for the Interactive Connectivity Establishment protocol. The candidates represent a given combination of IP address, port, and transport protocol to be used. A single computer may have multiple network interfaces (wireless, wired, etc.), so can be assigned multiple IP addresses, one for each interface.

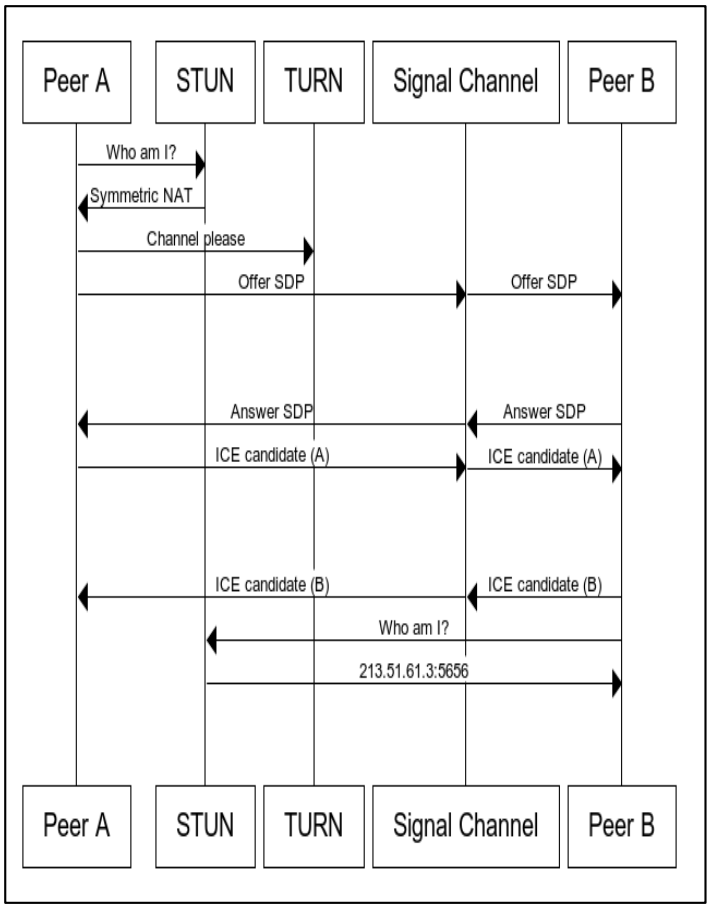

Figure 2: Exchange mechanism

IV. LITERATURE SURVEY

ACLs (Access Control Lists) and CAP (Capability-based Security) [1] are the authorization models. The permissions needed by users to obtain objects are controlled by ACLs. Whereas, specific permission on a given object are given using tokens using CAP. To compare the two proposed models' strengths and weaknesses, benchmark is generated for our Nubomedia prototype ACL and CAP implementations. Web Services can retrieve user identity information from a dedicated provider using Single-sign-on systems [2] (such as Facebook Connect). A most recent approach of SSO protocol like Browser-ID is used alongside OAuth2.0 and OpenID Connect. The development of browser-based peer-to-peer web applications is enabled using ufo.js [3], which is a novel architecture. Ufo.js makes use of the W3C Web-RTC data channel API. Security architecture to be integrated with federated Web identity systems is proposed by Web-RTC. The identity protocols that don't satisfy Web-RTC requirements are incompatible with the security architecture. Therefore, Li Li [4] proposed two alternative architectures, Web-of-Trust model and a mirror presence system, to fill these gaps. Web identity authentication and resolution mechanisms provide the basic means for communication and to collaborate safely and efficiently. A system to create video conferencing application that is as simple as possible for the end user is proposed by R. Vapenik [5]. The creation of a pilot web based videoconferencing system was the main goal of the work. The user should be able to share its resources, such as video, audio and data. In addition, the protection of the system should be granted. A number of issues that are specific to Web-RTC enterprise usage are illustrated and discussed by Alan Johnston [6]. This paper has begun to look at enterprise requirements for permitting Web-RTC media flows to cross enterprise boundaries. The existing state-of-the-art Session Border Controllers will not work with Web-RTC and many of their principles do not really apply to Web-RTC. The operations of outlining and discussion have been performed on a number of potential partial solution approaches. The analysis in this paper shows that while there are some promising potential approaches, the design of a Secure Edge to permit enterprise authorization and application of policy to Web-RTC traffic is far from solved today.

V. RESTRICTING UNTRUSTED USER

The existing system of Web-RTC focuses and overcomes three important security issues:

- Secure signaling to the server
- Security of the media
- End-to end authentication

The first issue of secure signaling to the server is look upon by HTTPS protocol. Whatever messages are exchanged between the web browser and web server is totally secured by HTTPS. The security mechanisms like encryption and authentication of the data is managed by HTTPS. The second security issue of security to the media like audio and video exchanged between to peers is handled by DTLS-SRTP protocol. DTLS protocol is used to encrypt data channel and is also used to exchange keys over network. These exchanged keys are then used for encryption purpose by SRTP protocol. This protocol uses the keys to encrypt media like audio and video using the AES encryption algorithm.

Identity Provider (IdP) handles the third security issue of end-to-end authentication. IdP can be thought of as a service which provides identities to the web browsers when required. Whenever any browser wants to communicate with other browser, it first issues Id from the IdP. This process is known as Identity Generation. Then this Id is sent to the other browser prior to the communication. The other browser verifies this Id from IdP. This process is known as Identity Verification. Once both the browsers are verified, the communication starts. Thus Web-RTC provides end-to-end authentication.

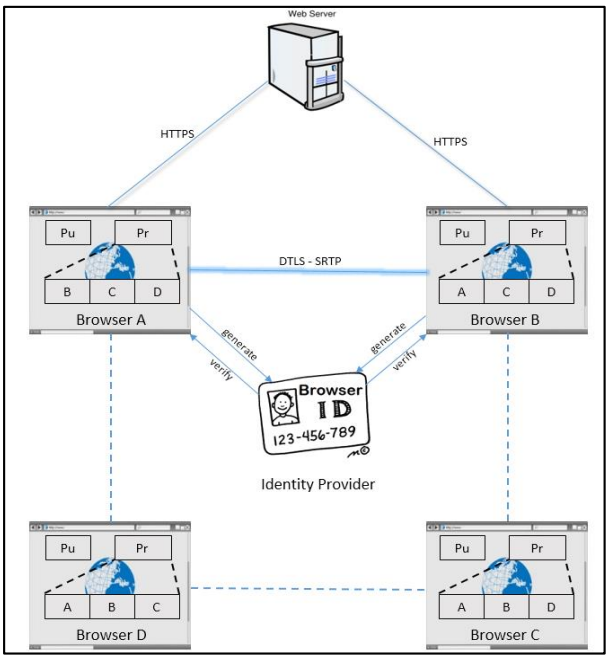

Figure 3: Concept of Public and Private Ids

There is another major problem in existing Web-RTC model. Any user can directly start communication with other users in the network. Consider a scenario where a video conferencing technology is used by a teacher to deliver the lecture. But only those students should be allowed to join the conference who are students learning under the particular teacher. Web-RTC doesn't provide any mechanism as such. This issue is handled in the proposed system.

As shown in figure 2, each browser is slightly modified to contain public and private field (not to be confused with public and private keys). These fields perform certain functionality. Public field contains the Id of the self-browser. Whenever there is a request for Id of the browser from other browser, the public is accessed and Id is sent. On the other hand, private field contains the Ids of the other browsers in the network. So if any other browser wants to communicate with it, this browser will first scan the private field contained in it. If it finds an entry in the private field, then it allows the further communication, else it ignores the other browser. Thus, with the introduction of public and private fields, untrusted user can be restricted from communication.

Approach for achieving target:

1. Firstly, the browsers who want to communicate requests the server for a channel through the HTTPS protocol.
2. The server verifies the browsers.
3. Each browser then asks the IdPs for the keys of each other and also asks for the permission.
4. The IdP then verifies the browser and checks whether they are available for the communication in the format of peer-to-peer network.
5. If available, the Identity Provider generates and passes on the key to the browser.
6. The browsers seek the key and send it to each other for individual identification process.
7. After the verification process a secure link is established.
8. This secure link is made up by merging the DTLS and SRTP protocols.

VI. RESULTS

As shown in the following pictures, an unknown user (figure 3.A) is trying to call in some another network who is not in the contact list of the other user (figure 3.B). The other user is restricting the call and is shown a message regarding the same.

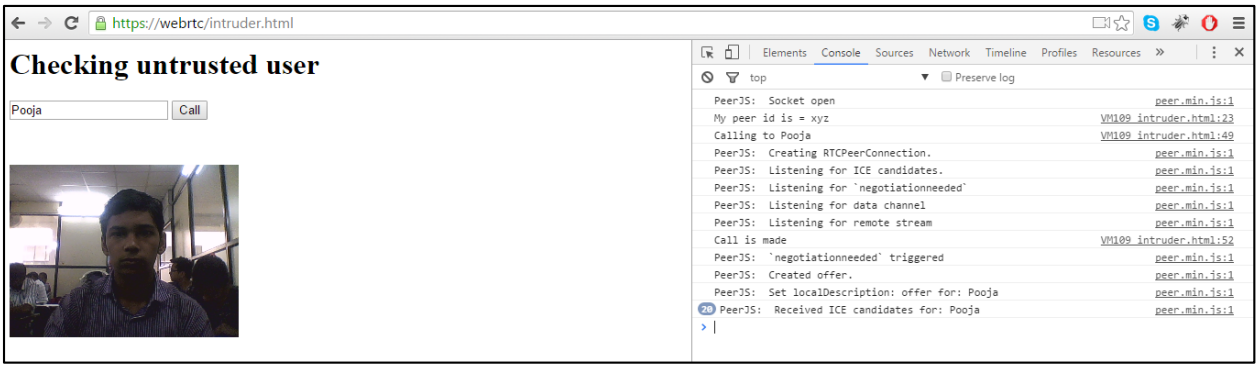

Figure 3.A: Untrusted user trying to call in the system.

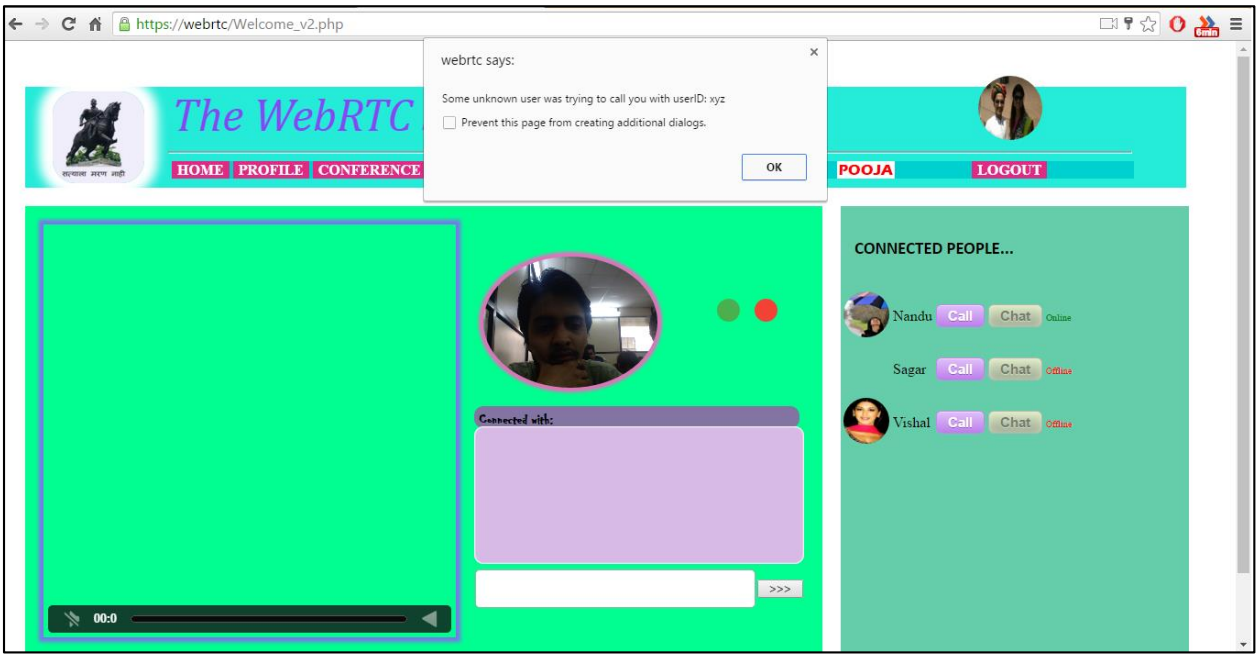

Figure 3.B: Message from the server informing about the intruder in the system

VII. REFERENCES

1. Luis López-Fernández, Micael Gallego, and Boni García-Universidad Rey Juan Carlos, David Fernández-López and Francisco Javier López-NaevaTec – "Authentication, Authorization, and Accounting in Web-RTC PaaS Infrastructures", IEEE Computer Society, November/December 2014.
2. Victoria Beltran and Emmanuel Bertin-Orange Labs, Noël Crespi-Institut Mines-Telecom – "User Identity for Web-RTC Services: A Matter of Trust", IEEE Computer Society, November/December 2014.
3. Bevilacqua, P. Boemio, S.P. Romano – "Introducing ufo.js: a browser-oriented p2p network", International Conference on Computing, Networking and Communications, Internet Services and Applications Symposium, 2014.
4. Li Li, Wu Chou, Zhihong Qiu, and Tao Cai-Huawei Shannon (IT) Lab – "Who Is Calling Which Page on the Web?", IEEE Computer Society, November/December 2014.

5. R. Vápeník, M. Michalko, J. Janitor and F. Jakab-Department of Computer and Informatics – "Secured Web Oriented Videoconferencing System for Educational Purposes Using Web-RTC Technology", 12th IEEE International Conference on Emerging eLearning Technologies and Applications, December 4-5, 2014.
6. Alan Johnston, John Yoakum, and Kundan Singh, Avaya Inc – "Taking on Web-RTC in an Enterprise"- IEEE Communications Magazine, April 2013.
7. Wajdi Elleuch Dep. Computer Science and Applied Mathematics- "Models for Multimedia Conference between Browsers based on Web-RTC", 2013 Sixth International Workshop on Selected Topics in Mobile and Wireless Computing.
8. Kundan Singh and Venkatesh Krishnaswamy, Avaya Labs- "A Case for SIP in JavaScript" IEEE Communications Magazine, April 2013.

