UDC 004.455..004.738.5.057.4(045)

DOI: 10.18372/2310-5461.36.12222

E. V. Dubchak
National aviation university
e-mail: Dubchak.E.V@gmail.com
orcid.org/0000-0001-9739-3960

I. B. Petrash
National aviation university
orcid.org/0000-0003-3784-1252
e-mail: Petrashillia@gmail.com

COMPARATIVE ANALYSIS OF SIP-LIBRARIES. IMPROVEMENTS OF JSSIP LIBRARY AS THE MOST RELIABLE, FUNCTIONAL AND ACCESSIBLE

Introduction

WebRTC has become increasingly popular among different organizations, companies and universities over the past few years.

The security of mission critical systems is an advantage for the most prosperous organizations that are famous in the world. In order to communicate and share valuable information, WebRTC is one of the ways to do that [1].

WebRTC lets users make calls right from a web page without any plugin. This was made possible using the media APIs of a browser to fetch user media, WebSocket for transportation, and HTML5 to render the media on the web page. Thus, WebRTC is an evolved form of WebSocket communication. WebSocket is a Transport Layer protocol that carries data.

The WebSocket API is an Application Programming Interface (API) that enables web pages to use the WebSocket protocol for communication with a remote host [1].

Staging problems

WebRTC (Web Realtime Communications) enables peer-to-peer video-, audio-, and data communication between two web browsers. This allows for video calling, video chat, and peer-to-peer file sharing entirely in the web browser, with no plugins [1].

There are a lot of ways where real-time communication application may impose security risks, on both the carrier and the end user. Such security risks can be applicable to any application which deals with the transmission of real-time data and media. It is important to think of SIP-library as a secure way to transmit audio and video signals.

Nevertheless, SIP-libraries should be tested and evaluated by different criteria such as : cross-browser compatibility, accessibility, integrity, popularity, code quality, functionality and accessibility.

To begin with, the communication model between a client and remote host is based on the JSEP architecture, which differentiates the signaling and transacting of media into different layers (Fig. 1) [1].

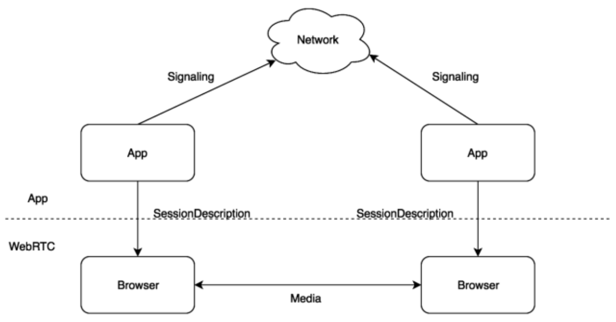

Fig. 1. JSEP Approach

Fig. 1 depicts two peers, A and B, where A initiates communication with B. A will have to call the create Offer function in order to begin the session. A also mentions details such as the codec, which sets up the local config. The remote party, B, reads the offer and stores it using the set Remote Description function. In order to generate an appropriate answer, party, B, calls the create Answer function, with the use of the set Local Description function, and sends an answer back to the initiator over the signaling channel. Nevertheless, when A gets the answer, it also stores it with the use of the set Remote Description function, and thus, the initial setup is complete. It may be repeated for multiple offers and answers [2].

Real-time Transport Protocol (RTP) is the way for media to flow between end points. RTP in WebRTC is enforced by Interactive Connectivity Establishment (ICE) protocol candidates, which could be STUN or TURN. ICE is required to establish the correct state of the firewalls in order to prevent the blocking of any of the UDP or TCP ports.

Signaling is an essential activity to establish any kind of network-based communication. It lets the endpoints share the session description and media information before setting up the path to actually exchange media. For a simple WebRTC client, there are Java Script-based WebSocket servers that can provide such signaling in a enduring, full duplex, real-time manner. Kamailio is one such server [1].

Kamailio is an Open Source SIP Server released under The General Public License, able to handle thousands of call setups per second. Kamailio can be used to build large platforms for VoIP and realtime communications — presence, WebRTC, Instant messaging and other types of applications. Furthermore, it can be easily used for scaling up media servers like Asterisk, FreeSWITCH or SEMS [3].

Goal — to provide comparative analysis of the most common SIP-libraries based on JavaScript. Moreover, it is essential to point out the advantages and disadvantages of these SIP-libraries in order to choose the most scalable, adaptable and most of all, the most secure library.

Considering the many SIP-libraries avaiable, it is important to include criterias, such as : cross-browser compatibility, accessibility, integrity, popularity, code quality, functionality.

Nevertheless, as comparisons are provided, it is important to analyze and highlight suggestions that show how to improve the given library in order to protect information from existing cyber attacks, namely XSS.

There are a lot of popular JavaScript libraries that offer easy-to-integrate support for WebRTC communication with the help of SIP signaling:

1. **SIP JS**: This is an SIP stack in JavaScript to implement SIP-based audio and video user agents in the browser. The demo application has the option to switch between WebRTC capabilities and Flash for browsers that support and do not support WebRTC [1; 4].

2. **JS SIP**: This is an SIP over WebSocket transport Application Program Interface (API) for audio/video calls and instant messaging. It works with all SIPWS-compatible SIP servers such as OverSIP, Kamailio, and Asterisk servers [1; 5].

3. **SipML5**: This is an open source JavaScript library with a provision for RTCWeb Breaker (audio- and video- transcoding when the endpoints do not support the same codecs or the remote server is not compatible with RTCWeb). For instance, features such as audio/video call, instant messaging, presence, call hold/resume, explicit call transfer, and Dual-tone multi-frequency (DTMF) signaling using SIP INFO are present in this library [1; 6].

4. **QuoffeSIP**: This is another WebRTC SIP library that provides establishment of real-time communication between browsers. This is developed in CoffeeScript (simple syntax). It features video/audio call capabilities using SIP over the Websockets protocol and also uses the SIP Outbound and Globally Routable UA URI (GRUU protocols) [1,7]. Comparisons of these libraries by different criteria can be seen in the following Tables 1, 2, 3.

Table 1

Reliability comparative analysis of SIP-libraries based on JavaScript

Reliability	SIP.js	JsSIP	sipML5	QuoffeSIP
SIP UA registration with the help of SIP Web socket	+	+	–	–
Code quality	–	+	–	–
SIP protocol through Websocket	+	+	+	+
SIP Outbound and GRUU protocols	+	+	–	+

Functionality comparative analysis of SIP-libraries based on JavaScript (Table 2) depicts which functionalities include different libraries.

Table 2

Functionality comparative analysis of SIP-libraries based on JavaScript

Functionality	SIP.js	JsSIP	sipML5	QuoffeSIP
SIP UA registration with the help of SIP Web socket	+	+	–	–
Code quality	+	+	–	–
Video / audio calls	+	+	+	+
Sending DTMF with SIP INFO	+	+	–	–

Next Table. 3 shows cross-browser compatibility of SIP-libraries based on JavaScript.

Table 3

Cross-browser compatibility of SIP-libraries based on JavaScript

Cross-browser compatibility	SIP.js	JsSIP	sipML5	QuoffeSIP
Firefox	+	+	–	–
Chrome	+	+	–	–
Android	+	+	–	+
Internet Explorer	–	–	–	+

In order to evaluate these libraries, it is essential to provide criterion comparative analysis of SIP-libraries based on JavaScript (Table 4).

Table 4

Criterion comparative analysis of SIP-libraries based on JavaScript

Criterion	SIP.js	JsSIP	sipML5	QuoffeSIP
Reliability	8	10	7	7
Functionality	7	9	7	7
Accessibility	9	9	8	6
Code quality	9	10	8	8
Popularity	9	10	8	6
Sum	**42**	**48**	**38**	**34**

As can be seen from the Table. 4, JS SIP tends to be the most reliable, functional, popular SIP-library based on JavaScript — 48 points out of 60.

There are a few components, that a typical WebRTC SIP-based client include [4]:

1. Cascading Style Sheets fot styling a page.
2. SIP stack(JavaScript library form).
3. WebRTC media API is an advantage to render a peer-to-peer connection.
4. An HTML5-based graphical interface to provide inputs such as registration parameters, self-URI, URI of the party to be called.

The components that must be deployed on the network side are as follows[4]:

1. The WebRTC gateway.
2. The SIP server to embed the SIP application / proxy logic.

As can be seen from Fig. 2, WebRTC client communicating with SIP client through SIP server that also acts as SIP-WebSockets to SIP convertor [1]:

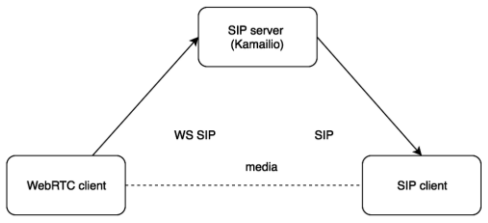

Fig. 1.2. WebRTC client communicating with SIP client through SIP server

Solutions and improvements

The browser's job is to enable access to the internet, while providing appropriate security protections for the user.

The security requirements of WebRTC are built directly on this requirement; the browser is the portal through which the user accesses all WebRTC applications and content. The level of trust provided to the user by WebRTC is directly influenced by the user's trust in the browser. Nonetheless, Browser Trust Model is a crucial moment in establishing appropriate protections in order to prevent injections and different types of attacks.

The W3C Cross-Origin Resource Sharing (CORS) allows the browser to contact the script's target server to determine whether it is willing to participate in a given type of transaction. As such, cross-origin requests can be safely allowed, by giving the target server the option to specifically opt-in to certain requests and decline all others [8].

WebSockets is another option that allows alternative functionality, but on transparent channels rather than isolated HTTP requests. Once such a connection has been established, the script can transfer traffic and resources as it likes, with the necessity of framing as a series of HTTP request/response transactions [8].

In order to prevent any types of cyber attack such as XSS, it is better to write the client part in a more proficient way, namely with the help of React&REDUX JavaScript library. It is important that all user inputs should have HTML entities escaped.

Conclusion

With all things considered, the obvious conclusion to be drawn here is that, in the modern age of smartphones and mobile devices people are communicating more than ever, and in even more personal ways than we have known before.

WebRTC has a big advantage over most VoIP services in the security area. Until now, most services have typically treated security as optional, meaning most end users use VoIP calls without encryption.

Nonetheless, security and integrity of different SIP-libraries are become invaluable.

Based on the widespreadness of different SIP-libraries that were mentioned above, this article has considered the security, accessibility and reliability of the most common SIP-libraries based on JavaScript. Furthermore, it has provided a cross-browser compatibility comparative analysis of the most common JsSIP libraries. Ultimately, the JsSIP library was considered the most reliable and accessible library among others, thus we suggested some improvements by considering the different types of cyber attacks, namely XSS.

Needless to say, the JsSIP library makes use of the WebRTC stack present in modern web browsers for enabling audio/video realtime communication. In the near future, it is expected to see more communication services providing largely increased security for their users. But for now, JsSIP is one of those who are leading the charge.

LITERATURE REVIEW

1. **Blake J**. WebRTC Integrator's Guide / J. Blake, S. Kopestake. — Packt Publishing Ltd. Livery Place — US-2014. — 382 p.

2. Datatracker [Electronic resource]: JavaScript Session Establishment Protocol. — (1 file, 42 notes). — US-Access mode: https://datatracker.ietf.org/doc/draft-ietf-rtcweb-jsep/

3. IEFT [Electronic resource]: Requests for Comments: SIP. — (1 file, 1 note) — US-Access mode: https://tools.ietf.org/html/rfc3261.

4. DEMO JsSIP [Electronic resource]: JsSIP. — (1 file, 1 note) — US-Access mode: http://theintencity.com/sipjs/phone.html?network_type =WebRTC.

5. DEMO TryIt [Electronic resource]: TryIt. — Access mode: http://tryit.jssip.net.

6. DEMO sipML5 [Electronic resource]: sipML5 — Access mode: http://sipml5.org/call.htm.

7. DEMO Quobis [Electronic resource]: Quobis — US — Access mode: http://talksetup.quobis.com.

8. A Study of WebRTC Security [Electronic resource]: WebRTC-Security — US-Access mode: http://webrtc-security.github.io.

Дубчак О. В., Петраш І. Б.

ПОРІВНЯЛЬНИЙ АНАЛІЗ SIP-БІБЛІОТЕК. ШЛЯХИ УДОСКОНАЛЕННЯ БІБЛІОТЕКИ JSSIP

У статті розглянуто безпеку, доступність та надійність широко використовуваних SIP-бібліотек, написаних мовою JavaScript, та надано порівняльний аналіз кросбраузерності SIP-бібліотек. Також описана модель клієнт-віддалений хост. Розлянута JsSIP-бібліотека як найбільш надійна та доступна; наведено можливі шляхи вдосконалення щодо підвищення протидії певним видам кіберзагроз.

Ключові слова: протокол VoIP; SIP; зв’язок; WebRTC; RTP; Kamailio; STUN/TURN сервери; JavaScript; JSEP; ICE; API; QuoffeSIP; sipML5; JsSIP; SIP.js; OverSIP; SIP Outbound / GRUUпротоколи; UDP;TCP; HTTP; XSS; GPL.

Дубчак Е. В., Петраш И. Б.

СРАВНИТЕЛЬНЫЙ АНАЛИЗ SIP-БИБЛИОТЕК. ПУТИ УСОВЕРШЕНСТВОВАНИЯ БИБЛИОТЕКИ JSSIP

В статье рассмотрены безопасность, доступность и целостность широко используемых SIP-библиотек, созданых на языке JavaScrip, и приведен сравнительный аналіз кроссбраузерности SIP-библиотек. Описана модель клиент — удаленный хост. Рассмотрена JsSIP-библиотека как наиболее надежная и доступная; приведены возможные пути усовершенствования для повышения противодействия некоторым видам киберугроз.

Ключевые слова: протокол VoIP; SIP; связь; WebRTC; RTP; Kamailio; STUN / TURN серверы; JavaScript; JSEP; ICE; API; QuoffeSIP; sipML5; JsSIP; SIP.js; OverSIP; SIP Outbound / GRUUпротоколы; UDP;TCP; HTTP; XSS; GPL.

Dubchak O. V., Petrash I. B.

COMPARATIVE ANALYSIS OF SIP-LIBRARIES. IMPROVEMENTS OF JSSIP LIBRARY

In this article, are considered the security, accessibility and reliability of the most common SIP-libraries based on JavaScript. It describes of the communication model between a client and a remote host. This article provides a cross-browser comparative analysis of the most common SIP-libraries. The JsSIP library has been considered the most reliable and readily available, thus it is suggested some improvements by considering different types of cyber attacks

Keywords: VoIP – protocol; SIP; signaling; WebRTC; RTP; Kamailio; STUN / TURN servers; JavaScript; JSEP; ICE; API; QuoffeSIP; sipML5; JsSIP; SIP.js; OverSIP; SIP Outbound / GRUU protocols; UDP; TCP; HTTP; XSS; GPL.

Стаття надійшла до редакції 11.11.2017 р.
Прийнято до друку 15.11.2017 р.
Рецензент — д-р техн. наук, Корнієнко Б. Я.