WebRTC（web real-time communication)
Security and pricvacy

Author:Yuhuai Luo
The Department of Information Technology,Faculty of Science -The university of Auckland,yluo044@aucklanduni.ac.nz

Additional Keywords and Phrases: WebRTC,security,privacy,authentication

Abstrct

The topic under discussion here is an open, royalty-free platform called webRTC (web real-time communication). The use of RTC products has increased dramatically in the last decade. As a result, the interest in webRTC is growing. In addition, webRTC security and privacy are becoming more and more important as people's perception of self-protection on the IT grows. This paper reviews the basic concepts and applications of webRTC, focusing on security and privacy aspects, analyzes some existing webRTC security limitations, and identify possible enhancement methods regarding some academic journals published in ACM and IEE

1. Introduction

webRTC is an API that supports web browsers for real-time voice conversations or video conversations without additional software packages.'This widely deployed communications platform powers audio/video calling, conferencing, and collaboration systems across all major browsers'[1].

'Over the past year, there has been a 100-fold increase of video minutes received via the WebRTC stack' [1].This is great proof that webRTC is becoming more and more known and accepted as a real-time web communication API, and this is probably due to some of its advantages such as convenience (no plugins or extensions needed), free of charge (integrates the best audio/video engines, very advanced codecs, but Google does not charge anything for these technologies, etc.). The large number of people using it ('by 2018, 4.7 billion devices will support WebRTC. '[3]) represents the value of his research, but he also has some security concerns, and in the current information technology environment, security and privacy vulnerabilities are of more significant concern.

In order to focus on webRTC research and the exploration of security and privacy, the research needs to understand the specific architecture of webRTC and analyze its possible pitfalls, based on exploring these aspects to find ways to improve it.

2. Methodology

First of all, for the accuracy and trustworthiness of references, the references used should come from professional libraries, use the university's library database to find professional information technology academic websites, such as ACM and IEEE, and use google scholar in order to understand more comprehensive webRTC information.

At the same time, the found literature should be explicitly evaluated. The first is the time of publication; the cited literature should not be too long, based on webRTC started in 2009 and google open-source webRTC started in 2011, this paper does not need to spend too much time on the timeliness.

Security and pricvacy

After that is the relevance, not all webRTC literature is relevant to research direction, need to find at least three to four pieces of literature on security and privacy and use it as primary research reference and study it is written for those people.

In terms of authority, the literature published in ACM and IEEE has a good guarantee in this regard; even google scholar is screened, but this is not absolute; also search the authors of these articles to ensure this and observe what professionals in the same field say about them.

It is also essential to evaluate the purpose of the literature, whether it is an opinion, scholarly reports, or widespread knowledge and whether their articles are biased. Most of the literature in the two digital libraries mentioned above is scholarly reports and some in-depth popular science, which did not take much time.

In order to quickly find high-quality literature with the above characteristics, usually start with an overview of the abstract to determine its relevance and purpose. If the abstract reflects the value of the review, read its introduction and conclusion and use this method to find literature that quickly meets the requirements.

3. Description of WebRTC

3.1 Basic webRTC Architecture

WebRTC (Web Real-Time Communication) is a framework that allows users to use their traffic to achieve real-time audio and video communication (APIs),currently also supports Android and ios.

The idea for WebRTC originated in late 2009; when the acquisition was completed in January 2011, 'the newly formed Chrome WebRTC team focused on integrating the code into Chrome and open-sourcing all the key components at webrtc.org' [1].

webRTC can act both as a user endpoint and as a transit point between two servers, although most webRTC services still rely on the client-server framework there are many other services deployed in p2p (peer-to-peer) (and webRTC is designated as a p2p technology by the original definition)

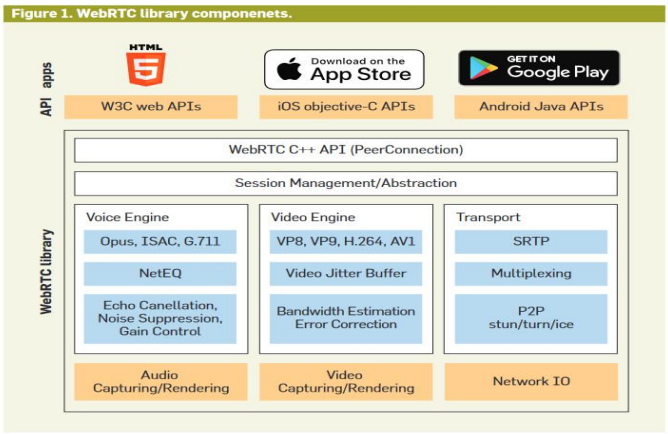


Figure1[1]

Security and pricvacy

As part pf figure1 shown, webRTC also brings together advanced real-time communication technologies, including but not limited to advanced audio and video codecs (Opus and VP8/9), mandatory encryption protocols (SRTP and DTLS), and network address converters (ICE & STUN).

It should be noted that webRTC p2p is divided into four steps: signaling, connection management, security, and media transfer (this review will focus on security, but in fact, each step covers security-related issues in the middle, which will be mentioned below). These four steps are composed of many other protocols, webRTC as an API and as a protocol applicable and only applicable to javascript.

3.2 Application aspects of webrtc

'Most modern services that use voice or video are either based on the WebRTC protocols or have the ability to use them in addition to the native protocols the service originally deployed with. ' [1]

In addition to the advantages covered by webRTC itself, the covid-19 virus that will be popular in 2020 is also an essential factor in creating this situation. Because of the general environment, people worldwide are starting to educate, communicate, work through video and voice, the demand for webRTC has grown exponentially compared to the past，thus webRTC becoming one of the most important information technologies. Also, webRTC has shown an enormous capacity beyond what was initially predicted, from the most straightforward peer-to-peer video and voice communication to providing high-quality multi-user conversations through machine learning (webRTC's open framework allows them to use machine learning and artificial intelligence), covering everything from games, AR/VR and more.

WebRTC is now used globally in various fields such as teleconferencing, online education, insurance claims, telemedicine, online sales, and video customer service. Customers can innovate business scenarios based on Full-time WebRTC to make any business process with communication capability and significantly improve business efficiency.

4. Implementation

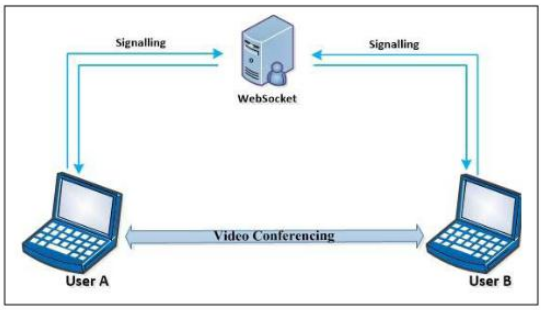
'Google Chrome builds Node.JS as a server platform and is Google's open source high-performance JavaScript engine. Node.JS is an appropriate platform for building network web application'[2].

There are two main steps covered in implementing WebRTC video conferencing between browsers, namely the client application and the signaling mechanism.

4.1. Client application

The use of SDP (Session Description Protocol) for exchanging signaling and providing answers is used in the client application. In simple terms, user Peter sets the local description and creates the offer method (), which is then sent to user Mike through the signaling server, and all Mike has to do is set Peter's description to the remote description. Similarly, user peter sets the local description and sends the answer to user peter, and when Peter receives the description, it also sets it to the remote description

Security and pricvacy



Figure[2]

4.2.Signaling

Signaling implementation can use the WebSocket mechanism, as shown in figure [2] for the server architecture via WebSocket. After the user Peter mentioned in the client request sends a request to the server, it generates the user media from the server after getting the control information, and after the user Mike responds, a response signaling message is sent, after which both parties can start establishing a peer-to-peer connection. The provisioning and response messages are transmitted by the SDP format mentioned above.

4.3 Difficulties in implementation

Inevitably, some difficulties may also be encountered in practice. First of all, 'WebRTC has not yet specified signaling management to allow developers to make modifications.' [2].

After that, the transmission quality is difficult to guarantee.

WebRTC's transmission design is based on P2P, making it challenging to guarantee transmission quality, and the optimization means are limited, so only some end-to-end optimization can be done, which is challenging to cope with the complex Internet environment.

However, there is no need to worry too much about the point in this simple peer-to-peer experiment.

To provide optimization and compatibility, it is necessary to use MCUs, but the problem is that MCUs are very expensive.

5. Security

5.1Signaling Security

According to the part mentioned in section 4, for the application and practice of webRTC, signaling (handshake) is an indispensable part. Based on the signaling mechanism of webRTC, as long as the security of the signaling channel and the server can ensure that there will be no other malicious attacks in the middle, which is one of the advantages of webRTC. But also, as mentioned above, webRTC lacks signaling management, so 'the signalling channel needs to be super secure and trusted to not endanger user's privacy'[3].

At the same time, webRTC's media communication is encrypted. 'DTLS is used for key exchange, and both endpoints generate a new self-signed certificate for each connection'[3], ensuring that the channel information is encrypted.

5.2 Safety hazards

Security and pricvacy

5.2.1 Signaling channel

For webRTC executed in the browser (the primary application method), the server may serve the web application and act as a signaling server, so the server operator has a significant degree of control over your signaling mechanism and source code. Although the webrtc signaling mechanism has security guarantees, it is easier for the operator to modify the source code.

If webRTC is not executed in the browser but used in the Application, the application logic and signaling mechanism are separated, ‘a compromised signaling server could insert man-in-the-middle nodes without other nodes noticing’ [3].

The existing solution to this problem is to seek a trusted third party who will intervene (during the handshake), but this increases the external dependency and cost of webRTC. Another possible workaround is to have the webRTC generate a self-signed certificate before establishing the connection.

5.2.2 IP address leakage and local networks

Whether for companies or individual users using webRTC, IP address leakage is always unwanted (and helps attackers launch attacks), and webRTC has a fatal drawback in this regard.

To facilitate this, most browsers that support webRTC today have an environment for accessing webRTC from JavaScript. This results in all available interfaces' IP addresses being exposed to the javascript console, mainly because webRTC always finds the best path between two nodes, using local IPs for the same network and traversal techniques for hosts after NAT.

Like what Google Maps can also do, the public IP address does not require additional permissions to locate it roughly.

The solution to this is that the WebRTC API should be ‘extended with fine-grained permission controls’[3].

5.2.3 webRTC fingerprint recognition

Fingerprint recognition frameworks can track browser users, and simple methods such as privacy mode do not help them clear this hidden problem.

As mentioned in 5.2.2, webRTC can compromise IP addresses, it is difficult to pinpoint and identify a device with a single IP address, but many devices have multiple IP addresses such as the IP address to access the local network and the IP address assigned to the wifi adapter. ‘With every address, entropy is added and strengthens the uniqueness of the result’[3]. If the public address of two devices is in the same range on the public network, they are likely to be on the same network, ‘after which analysis can be validated by instructing the devices to establish a direct WebRTC connection to the identified private IP address ‘[3].

The solution is similar to the one mentioned in 5.2.2.

5.2.4 Broadband

With the trend of more and more users connecting to broadband, the security and non-usability of broadband are becoming more and more critical.

The WebRTC connection-initiation mechanism relies on the

‘participating nodes to send protocol, IP, and port combinations - as possible connection endpoints’ [3]. Thus, the webRTC environment on the target side and the modification of IceCandiates using JAVAscript allows the infected client to force other clients in its signaling domain to connect to any address, and client connection checking can cause problems with outgoing broadband and in the most severe cases can lead to server shutdown.

A possible way to solve this problem is to warn the user when an exception is detected and have webRTC block incoming requests.

5.3 Improve webRTC security without third party

In the paper, Yilmaz, Brak, Ozdemir, and others in 2020 [4], boldly and creatively proposed using blockchain technology to solve some security problems in webRTC. The security of peer-to-peer communication without relying on third parties can also be improved by blockchain technology.

First, the DTLS security method based on webRTC is fingerprinted, and note that a self-signed certificate needs to be generated here to construct a secure DTLS channel. After that, prepare the smart contract, describe a struct definition in Solidity, ‘the fields of this struct are critical; they are identity (id), name, surname, address that identifies the user in the Ethereum Word, and fingerprint information of the user's certificate’[4], these methods are defined according to the smart contract on the blockchain network. Still, they can only be invoked by the target user due to the authorization feature. After this, the communication group needs to be specified, and in fact, an administrator is still required for authorization in this method, but the administrator can still be the standard user in the system. The administrator is also responsible for deploying the prepared canonicals to the blockchain network. Finally, a smart contract is used as the identity provider, i.e., the fingerprints of these certificates are written into the blockchain's smart contract after matching (performed by the administrator) and making changes to permissions and specific operations, after which normal webRTC operations are performed.

This increases the security of webRTC communication without relying on third parties.

6. Authentication

Another problem brought about by the rise of webRTC is authentication. As mentioned in the previous section, the information transmission of webRTC is DTLS and encrypted, so it can be said that the security of information is guaranteed to a certain extent. Still, if the user is not sure whether the person receiving the message is really the one he wants, information security will be meaningless, so this chapter will explore the webRTC authentication mechanism.

6.1 webRTC Authentication system

First of all, it should be clear that web identity is not the same as the communication industry; for example, in the telephony domain, identity is managed as shared between providers, while for the website, 'user identity is basically an immutable link between profile authorization rules, and credentials ' [5]. Their main purpose is still to allow users to log in rather than to verify their identity. Of course, the retrieval of user identity information in web services cannot always be absent, and the SSO system used by Facebook, for example, accepts identity assertions, which ameliorates this problem to some extent (it should be noted that SSO is still primarily used for user login rather than identity confirmation). webRTC does use some of the underlying SSO protocols ('mainly BrowserId, OAuth2.0, or OIDC' [5]). Identity authorization in webRTC relies heavily

on CP, the authentication process that authorizes users to log in, and the ability to accomplish these privacy-related aspects relies on identity provisioning models, which in webRTC are trust, partial trust, and full trust models. The performance and work of cp can vary in different models. However, the concept of trust or distrust of cp is still unclear in the domain of webRTC, and webRTC's application of the SSO underlay is slightly inadequate, which makes webRTC not well defined for user identity.

6.2 Web based A/V communication scenario

In Bostani and Grégoire's experiments and report [6], they experimented with six different application authentication scenarios, namely, guest, an unregistered user with CAPTCHA, Isolated, and Federated, Isolated or Federated, Isolated with federated as support, they conducted authentication-related experiments for these six scenarios covering speed, efficiency, ease of learning, memorability, preference, and accessibility. They concluded that the usability of web A/V base on webRTC authentication. The top three rankings are Guest, Isolated or Federated, and Federated, and it is also necessary to use different authentication scenarios depending on the situation.

7. Limitation

In the above article, it is easy to find that even though webRTC is based on a solid security foundation, there are still many limitations in security. webRTC's authentication has never been well resolved, and the safety of the signaling channel is also based on having a very trusted channel and a service provider that can be trusted. There are many different solutions mentioned above, most of them based on trusted third parties, but as the reliance on third parties increases, so do the third-party-oriented security risks. Some of the ideas that do not require third-party dependency are very innovative and creative and have some feasibility, but unfortunately, some need a lot of time and money to practice and apply. Some of them have defeated the purpose of webRTC as an open-source and free modifier for developers, making it difficult to achieve a natural balance.

8. conclusion

In general, before WebRTC, there was no unified industry standard to describe a device's real-time communication capabilities and connection establishment process. This was very confusing in applications where the need for real-time communication capabilities was particularly acute. Based on the demand and the general environment, driven by the demand for interaction between web applications and mobile terminal applications, more and more mobile applications for audio and video services use or use the technical specifications of WebRTC.

WebRTC part of the real-time communication solution is also the closest to the user part, represents the standardization of terminal technology specification, and is a good start.

WebRTC will significantly activate the information bond between people, remove the barriers of time and space between communication terminals, and become a powerful technical guarantee for the innovation of application scenarios.

In terms of security and privacy, in general, webRTC has been designed based on security considerations. As mentioned above, webRTC has a robust security system in general; even if the authentication system is not perfect, it also uses the underlying protocol and trust model of SSOand is not defenseless. The existing security risks and vulnerabilities of webRTC do have room for improvement, but it must be said that some issues are difficult to achieve a natural balance. In

the open-source and its own characteristics of the limitations of the current can only try to optimize his shortcomings, and feasible solutions, if it will harm the advantages of webRTC itself or is too challenging to implement, will also take time to improve.

References

[1] Niklas Blum,Serge Lachapelle， Harald Alvestrand.2021.WebRTC: real-time communication for the open web platform.Communications of the ACMVolume 64Issue .pages 50-54. https://dl-acm-org.ezproxy.auckland.ac.nz/doi/10.1145/3453182.

[2] Naktal Moaid Edan; Ali Al-Sherbaz; Scott Turner.2017.Design and evaluation of browser-to-browser video conferencing in WebRTC.2017 Global Information Infrastructure and Networking Symposium (GIIS).https://ieeexplore-ieee-org.ezproxy.auckland.ac.nz/document/8169813

[3] Andreas Reiter,Alexander Marsalek,2017.WebRTC: your privacy is at risk.SAC '17: Proceedings of the Symposium on Applied Computing.Pages 664–669.https://dl-acm-org.ezproxy.auckland.ac.nz/doi/10.1145/3019612.3019844

[4] Berat Yilmaz; Ertugrul Barak; Suat Ozdemir.2020.Improving WebRTC Security via Blockchain Based Smart Contracts.2020 International Symposium on Networks, Computers and Communications (ISNCC).https://ieeexplore-ieee-org.ezproxy.auckland.ac.nz/document/9297333

[5] Victoria Beltran; Emmanuel Bertin; Noël Crespi.2014.User Identity for WebRTC Services: A Matter of Trust.IEEE Internet Computing (Volume: 18, Issue: 6, Nov.-Dec. 2014) .https://ieeexplore-ieee-org.ezproxy.auckland.ac.nz/document/6939599

[6] Hassan Bostani; Jean-Charles Grégoire.2017.Usable authentication systems for real time web-based audio/video communications. 2017 20th Conference on Innovations in Clouds, Internet and Networks (ICIN).https://ieeexplore-ieee-org.ezproxy.auckland.ac.nz/document/7899400