Ronnakorn Rattanakornphan (Ron)
Boston University
EC 601: Project1

WebRTC Security

What is WebRTC

General

WebRTC stands for Web Real-Time Communication. It is a group of APIs that allows web browsers and mobile applications to directly communicate with each other. With peer-to-peer communication provided by WebRTC, the users do not need to install additional plug-ins or extra applications (Skype, for example) since WebRTC comes as a part of the web browsers. WebRTC acquires and communicates the information by utilizing 3 APIs for establishing and managing the connection. “getUserMedia” is responsible for requesting permission and obtaining data from the local devices as MediaStream. RTCPeerConnection will handle peer connection, including generating SDP description to send to the “peer” during the signaling process. Lastly, RTCDataChannel will “enable peer-to-peer exchange of arbitrary data”, for example, files, video, audio, etc.

Signaling & Communication process

Signaling is a process for establishing communication between 2 “peers” by exchanging data before the session. The information exchange is handled via a signaling server and is made up of several steps such as “definition of a shared information channel, protocol negotiation, and IP address exchange” (Feher et al., 2018). The exchange uses the Session Description Protocol (SDP), which is a representation that can provide details regarding multimedia session such as media capabilities(video, audio), the codec used in the session, Network information (IP address, for example) (NTT Communications, 2016)(Feher et al., 2018).

The actual direct communication happens after the completion of contact information exchange, however, another validation to confirm both parties’ identities must be made, using the key that was previously exchanged in the signaling step.

Security impact

Security has always been a big issue in the computing world, especially when the word "connection" is involved. The advent of WebRTC technology, like any communication, has led to many possibilities. However, it is best to pay attention to such technologies with the security aspect involved. The following section will discuss the security measures/protection provided by WebRTC technology.

No Plug-Ins or extra setup required

WebRTC technology exists as a part of WebRTC-compatible browsers (for example Google Chrome, Safari, etc.). Thus, it is not a plug-in and requires no extra process of installation, which removes the chance that a harmful program can pretend to be the desired application and expose the user to malware and/or viruses.

Fast Patching

Although WebRTC's security still possesses a risk to be compromised in the future, its tie with the browser allows them to benefit from browsers' fast-pace development. With browsers having a high frequency of patch, any flaws exist in WebRTC are likely to be fixed quickly, along with the release of browsers' update. Moreover, the automatic updates in browsers enable the fix to be delivered without having to wait for users' input, leaving a shorter time for exposed vulnerabilities.

On the other hand, imagine some bugs in desktop applications that expose a user's computer. In order to have it fixed, a series of questions are needed to be answered. Does the user have a way to report the bug? Does the user know who they can report (the bug) to? How long till the developer would respond? When will the patch be distributed? When will that patch be installed? With the aforementioned considerations, a typical application would be slowed to patch up any vulnerabilities, leaving a large window for any malevolent software to take advantage of. Thus, the web browsers' rapid response to vulnerabilities offers WebRTC applications stronger security.

Media Access

One of the security concerns is access to the local device. Microphone, camera, screen, all these equipment can be used maliciously if the control is not managed well. However, WebRTC required applications to obtain explicit users' permissions to access such local devices. Not only that, WebRTC also requires that the interface needs to clearly display when a device is being used. The reason behind this design is so that the users would be able to make a conscious decision, which will provide a chance for the user to reconsider their privacy before allowing their devices to be used.

Encryption

One of the major security protection offered by WebRTC is encryption.
First of all, WebRTC forbids the communication of the raw/unprotected data, and, thus, all WebRTC components are required to use encryption. WebRTC uses DTLS-SRTP for encrypting media and DTLS for data (Reiter et al., 2017). Moreover, WebRTC requires that its application needs to be run on HTTPS (else it will not work), which guarantees an additional security layer for the users.

Vulnerabilities

Even though WebRTC has offered good countermeasures regarding security issues, it is still far for perfect. The following section will discuss potential threats to WebRTC applications.

Users

One of the security consideration lies with the users. Not matter how complex an encryption is, or how ingenious the communication methods are, the users' privacy may still be at risk, caused the hand of the users themselves. Some privacy breaches may come from the silliest issues like forgetting to turn off the recording, or not knowing how much of the screen they are actually sharing.

Some possible fixes can be implemented on the clients side, for example, implement a time limit on how long a user can be away from the camera before it is shut down automatically, or mute the microphone when a certain period has passed without receiving audio input. In such cases, more studying might be required in order to adjust the appropriate time limit without interfering with the usage.

Signaling information

As mentioned above, the signaling process will initiate the exchange of information in order to establish a peer-to-peer connection. This information also contains some sensitive information like internal IP address. Consequently, the following scenario might lead to users' privacy being compromised.

Untrusted server

As the information needs to pass through the signaling server, the service provider may have a chance to intercept the message and figure out the user's address. Since the nature of WebRTC is that of API, a malicious website can leverage WebRTC's capabilities and use it in other contexts in order to obtain user's information such as IP address. For example, a malicious website can trigger WebRTC API to obtain data as part of its application like pretending to offer free storage space and service to record audio but actually establishing the connection using WebRTC, then obtain user's information such as internal IP address.

Signaling server is compromised

As the signaling server is responsible for forwarding communication requests and various other information. A security breach on the server can lead to many harmful consequences such as server failure, faking additional users participating in the conference, or performing man-in-the-middle attacks by making both clients communicating with a fake user and having fake user forwarding conversation to each client.

Assuming that the signaling server is compromised, a man-in-the-middle attack can be conducted easily by redirecting the "call" request from user A to the middle man who will initiate another "call" from himself to the user B. In this case, user A and user B are connected with the middle man who can just forward the information between user A and B to fake a direct connection between A and B.

Crashing the server

If the server didn't vigorously validate the input and provide a fault solution, an attack could be made by sending a wrong/mismatched input to the server which will make the server enable to parse the information and possibly crash as a result.

Harmful file transfer

Normally, the transferring of the file is done via server, providing an opportunity to sanitize the files and guarantee the safety of the receiver. However, with WebRTC, it is possible to send an arbitrary file directly to the receiver without passing the server. As a consequence, any harmful file (trojan horse, for example) can be sent to another person without being checked by the server. (Feher et al., 2018)

Although it is possible to design a checking mechanism on the clients' side to determine any harmful files, doing so might expose the method to the client who might try to circumvent the mechanism

Ref

Dutton, S. U. (2012, July 23). Getting Started with WebRTC - HTML5 Rocks. Retrieved September 6, 2020, from https://www.html5rocks.com/en/tutorials/webrtc/basics/

Dutton, S. U. (2013, November 4). WebRTC in the real world: STUN, TURN and signaling - HTML5 Rocks. Retrieved September 9, 2020, from https://www.html5rocks.com/en/tutorials/webrtc/infrastructure/

Feher, B., Sidi, L., Shabtai, A., Puzis, R., & Marozas, L. (2018). WebRTC security measures and weaknesses. *International Journal of Internet Technology and Secured Transactions, 8*(1), 78. doi:10.1504/ijitst.2018.092138

Google. (n.d.). Real time communication with WebRTC. Retrieved from https://codelabs.developers.google.com/codelabs/webrtc-web/

NTT Communications. (2016, October 29). A Study of WebRTC Security. Retrieved September 7, 2020, from https://webrtc-security.github.io/

Reis, C., Barth, A., & Pizano, C. (2009). Browser Security: Lessons from Google Chrome. *Queue, 7*(5), 3-8. doi:10.1145/1551644.1556050

Reiter, A., & Marsalek, A. (2017). WebRTC: Your Privacy is at Risk. *Proceedings of the Symposium on Applied Computing - SAC 17,* 664-669. doi:10.1145/3019612.3019844 https://webrtc.org/