Survey of WebRTC Security

WebRTC - Western Illinois University Graduate Thesis

Dennis McMeekan: November 2020

Abstract

This is a survey intended to allow for an introduction into security protocols that may be established when creating a WebRTC Peer-to-Peer application. Throughout group discussion, two certain security elements want to be attacked that were voted upon by committee members: myself, Dr. George, Dr. Sen, and Dr. Zhao. These two elements are data integrity and IP leak concerns.

Data Security

Going beyond server and client authenticity, it is vital to ensure that the communication stage, the main segment that is being used throughout this thesis implementation, holds secure. It is among my understanding, that the WebRTC application itself presents a very lightweight encryption among it's API calls and also throughout the implementation of the client and server architecture. This will be the basis for considering performance among our WebRTC application, and then further than a heavyweight encryption and decryption method is needed.

Real Time Image Filtering

For encryption and decryption, WebRTC Real Time Image Filtering can be implemented to prevent exploitations of accessing computer hardware of a separate client. With this, only output data will be accessible by the peer connection. This has been seen implemented using two different types of video input, "local" and "remote," but none truly testing and acting with a client-server infrastructure.

A very powerful example discovered is a WebRTC application that takes user input, and then applies different types of filters to the data being displayed [1]. This involves taking a user video stream, a canvas element, a buffer canvas, and then an output canvas. Expanding upon this, I believe it will be possible to these same aspects in manipulating data availability, but instead being able to access and alter the video being received on the remote side of the connection.

This manipulation can occur in many different ways, with the original possibility of inserting and decoding a bit encryption to the data. This was proposed by Dr. George, and through further research and analysis discovering different WebRTC API calls, I instead found a similar approach that instead focuses on altering the pixel Data that is being sent and received [2]. Although this may not be quite the extent to which a bit encryption may hold, it still will be suffice in determining a covert channel to allow for proper data integrity and in doing so allow for further statistics measurement.

With a proper bottom line for measurement in determing if the security protocols are efficient enough to be implemented at a much larger level, it is vital to conduct certain tests in doing. First, measurements must be taken on a very simple peer-to-peer connection being made with two video elements. Second, measurements must be taken by using real-time image filtering with an already set video manipulation. And third, measurements again must be taken using real-time image filtering, but instead of being pre-set, an added element of randomness must occur to allow for proper simulation of

control by the administrator of the WebRTC application. These will then be measured and averaged, and finally compared. These measurements will be primarily done through WebRTC API calls that allow for statistics to be calculated through peer-to-peer connections [3].

Sources

1. S. Donaldson, "Using WebRTC for Real-Time Image Filtering," *Sudo ISL Company* Accessed on: Nov 1, 2020. [Online] Available: https://sudo.isl.co/webrtc-real-time-image-filtering/

2. Mozilla, "Pixel Manipulation with Canvas," *Mozilla.* Accessed on: Nov. 10, 2020. [Online] Available: https://developer.mozilla.org/en-US/docs/Web/API/Canvas_API/Tutorial/Pixel_manipulation_with_canvas

3. K. Beihang, "Identifiers for WebRTC's Statistics API" *W3C.* Accessed on: Dec. 1, 2020. [Online] Available: https://w3c.github.io/webrtc-stats/

IP Leaks

A large issue with WebRTC applications, is IP leak concerns. This is due to the fact that a Peer-to-Peer connection requires that each client has each other's communication address. A similar issue was discussed and implemented with the creation of the TOR browser [4], but instead of using a simple WebSocket or HTTPS server, a Distributed Hash Table server was implemented that would allow for each client to still communicate directly with each other, but remain anonymous.

Time permitting, this will be further developed with a later research paper, because it has been determined that a majority of actual coding of project work will be spent establishing Covert Channels using Real Time Image Filtering.

Sources

4. D. Anon, "Everything you wanted to know about Tor but were afraid to ask," *Privacy.net.* Accessed on: Dec. 2, 2020. [Online] Available: https://privacy.net/what-is-tor/