Securing WebRTC
Master's Thesis Defense Presentation

By Dennis McMeekan, School of Computer Sciences

Western Illinois University

---

1. Introduction
2. Importance
3. Motivation
4. Security Concerns
5. Related Research
6. Problem Statement
7. Confidentiality Violations
8. Covert Channels
9. WebRTC API
10. Research Methods
11. Unsecure/Secure Prototypes
12. Performance Results
13. IP Leaks
14. Distributed Hash Table
15. Conclusion
16. Future Work
17. References

WebRTC is a Real-Time Video Communication Specification

This allows for no added-plugins or installs being required.

Relatively new and not deeply explored.

How WebRTC Works

- Web Browser Specification
- Allows for Two or More Clients to Communicate in Real-Time
- Involves a Web Server in which Protocols are established to Send and Connect each Client's Request/API calls.

WebRTC holds importance due to the immense growth of media communication – virtual environments (especially with COVID-19)

Open-Source and Royalty-Free: Available for all Individuals, but Google (creator) holds no accountability.

Not all Security Protocols have been Examined.

Motivated to find a new and not fully examined specification.

Hold a lasting impact on Computer Science and current or future Real-Time communication products.

Increase of Virtual Environments, Seemed Quite Fitting

Further Security throughout the WebRTC Specification.

Mitigate Vulnerabilities: Which we will touch upon further in the presentation.

Security Concerns: Two Chosen by the Committee

Confidentiality Violations

IP Leaks

Allowed for Prototype Implementation with the Time Permitting.

Two Essential Papers Need to Be Discussed in Terms of Related Research

One being Constructivism in Computer Science Education

Discusses Knowledge Development in which students are able to use teachers, researchers, and leaders as a source for growth and new horizons. Instead of simply using prior knowledge with no further establishments being made.

Second being A Note on the Confinement Problem

Very classic paper, that establishes that there needs to be limitations of executions that exist outside the base program.

Problem Statement:

"WebRTC has the foundation to allow for a secure and simple connection to be made by two users without installing native apps or plugins."

This specification is relatively new and has immense potential, but with this, security vulnerabilities do exist, and with that, need explored.

With the problem statement being established, we can being to dive into the security concerns.

Confidentiality Violations relate to a Peer-To-Peer Connection being made where a transfer of real-time data is at the center.

The great benefit of having open-source API is great for development, but can be quite concering considering hackers and exploitations.

This leads to covert channels.

Covert Channels result in sending data unintentioanlity.

This can be due to unexpected results of the specfication, in our prototype and thesis this is due to WebRTC API.

This can be harmless, but can also be harmful due to possible sending Sensitve Data or Malware.

How it Works

Exploit Certain Elements (WebRTC API) to send/receive data

WebRTC API is extremely detailed and resourceful, many webpages including specification specific API by Google.

Allowed for research to hold an understanding.

Then could look into real-life examples.

To then prototype implementation.

Two main languages, JavaScript and HTML5.

Picture provides a brief overview of the type of API being used per Client and Server

---

Covert Channel Implementation

This is done by Image Filtering: JavaScript Method

We have two separate connections, local and remote (representing each client)

With image filtering, we add an extra step of a canvas element.

Allows for the data being sent from local to remote to be delayed.

Our main research method was determing the effectiveness of covert channels.

Two main statistics being looked at:

Error Rate: How many bits received match the bits being sent

Covert Channel Bandwidth: How much time does it take for each bit or data to be sent

Two Separate Prototypes:

Unsecure: Covert Channel Implementation

Secure: Mitigating Covert Channels

Two Separate "Video" Testing Procedures

Black Screen: Webcam covered and provides consistent data rates

Sample Presentation: Webcam not covered, with myself acting like I am giving a presentation. Real-life example.

For our demonstration, sample presentation will be used to ensure a real-life example is seen.

Unsecure Prototype

Implement Covert Channels: Send/Receive a Bit

Implement Delay with Image Filtering, Sense the Delay, Receive The bit

Couple main elements being looked at

Error rate, covert channel bandwidth, and

T or the Input Rate: used to determine covert channel bandwidth, how often a bit is being sent.

Unsecure Demo

We can see through the console log:

A bit is being sent, the delay will occur and effect the bitrate,

In which we can the determine what bit should be received.

Used the .25 covert channel bandwidth (about 4 seconds)

As we will see with results, this was the best effectiveness are for covert channel implementation.

Results Unsecure

These were examined without using any covert channel implementation or delays, provided a baseline for judgement.

Bitrate sits around 1450, and the framerate just slightly above 50.

Results Covert Channels

We do lose some performance, as the framerate does drop about 10-15 frames.

The error rate is the most important element, because this determines effectiveness.

We see that roughly 2-5 seconds is the ideal time to send/receive data, as this held the lowest error rate.

And when there is less or more time than this, error rate is especially high. This is believed to arise from there not being enough time to determine the delay, or too long that the delay actual stabilizes.

Secure Prototype – Mitigate Covert Channels with Delay Mechanisms

Constant – Consistent Delay to provide constant noise.

Random – Random Delay to provide random, yet constant noise.

Simulates control by the admin.

Can result in performance concerns.

Ideally, increases the error rate.

Secure Demo

Very similar to the unsecure version, but instead we see added noise between the bits being sent and received.

Can notice that the bitrate will sometimes remain lower.

But the error rate does increase in terms from going 0 to 40.

Results Constant Delay

The error rate does generally increase for each Covert Channel Bandwidth.

The video quality is a bit worse in comparison to the two prior results, but this is a small tradeoff for implementing mitigation mechanisms.

Results Random Delay

There are some rates in which the error rate actually decreases in comparison to the constant.

- This is believed that due to the “randomness” and only sensing and receiving two bits, it is actually somewhat easier and at a higher covert channel bandwidth to sense and receive data.

Adding a third or “baseline” bit to essentially “reset” what is being sent could mitigate this and possibly help both sides.

Video Quality is very similar to the Constant Delay.

IP Leaks

Results in sensitive data being obtained.

Due to the Public IP Address being required through Google’s STUN/TURN Protocols.

With the public IP Address, geographical location and other personal data can be found.

IP Leaks can be Mitigated with a Distributed Hash Table

The Current Approach: STUN/TURN Protocol for Server Implementation

The New Approach: Implement a similar server seen with the TOR browser.

DHT Implementation

We can see the clients are on both sides of the diagram, with a series of enter, relay, and exit nodes in the middle.

Each client will use the enter nodes, which then is sent to the relay nodes, and then sent to the exit nodes.

This allows for each client to have no direct communication between each other and can prevent the public IP address being needed.

Worked extremely successful with TOR browsing but is questionable in relation to real-time communication.

Could result in performance concerns, but proper implementation and testing would be required.

Conclusion

WebRTC is susceptible to Covert Channels.

Image Filtering allows fluctuating bit rates through delays.

From this, bit can be sent and received.

Mitigation method involves delay mechanisms.

Constant and Random Delays.

Random is obviously the better route, because being constant this can be easily detected, and then covert channel implementation can be altered accordingly.

WebRTC is susceptible to IP Leaks.

Distributed Hash Table implementation can mitigate this.

Future Work

Network Concerns

We simulate the network connection using WebRTC API but using one web browser and network to help testing and implementation.

Ideally, this would need to be seen done using two networks and two web browsers with completely separated clients.

Prototype Implementation of a DHT Server for WebRTC

Actual Development, Testing, and Analysis.