Tin Can

Motivation

Today people spend more time in their web browsers than ever before. A wide range of activities, from media consumption to word processing, that were formerly provided by desktop applications are increasingly shifting toward web applications. In addition to client-server applications, the extensibility and strong community support of modern browsers has provided developers with the ability to create rich services that execute inside the browser.

At the same time, current events have resulted in a public more cognizant of the debates surrounding internet privacy than ever before. Projects such as Tor have reached a level of notoriety unthinkable until recent times. In response to these two contemporaneous trends, we set out to create a completely in-browser peer-to-peer communication app that would allow users to control their own data without sacrificing convenience or usability.

Methods

Tin Can relies on the relatively recent WebRTC technology to accomplish its goals. Through exchanging offer and answer messages, WebRTC allows two remote peers to form a direct connection. This connection can provide both raw data transport (DataChannel) and media streaming without any plugins. While these offfer and answer messages are typically conveyed via a signaling server, no server is actually required for a connection. We allow two users to form an initial connection through copy-and-pasting offer and answer messages after sending them through some other sort of secure channel. After this initial setup, the two users can introduce each other to their other peers by providing offer and answer forwarding.

Running the project

While Tin Can can technically run as a purely static page, we use a simple Node.js server for convenience and because of issues with Chrome's getUserMedia implementation.

To try out Tin Can:

1. Clone the repository
2. Install Node.js
3. Navigate to the project directory
4. Start the server by typing `node app.js`
5. Point your browser to `localhost:3000`

Acknowledgments

We'd like to thank Chris Ball for providing the inspiration for Tin Can with his serverless-webrtc project.