Paper title: Enterprise WebRTC powered by browser extensions.
Speaker: Kundan Singh. Track: IPTComm.

Hello everyone. My name is Kundan Singh. I will start my presentation with this quote.

"A technology is helpful only if it can be used"

This applies particularly to WebRTC when used in enterprises with restricted communication policies.

My presentation is about the challenges of using WebRTC in enterprises and how they can be addressed.

What is WebRTC?

Plugin-free browser-to-browser communication

Encrypted media flows and data channel

Unlike SIP, signaling is outside the specification

Status: not all browsers, not all devices

WebRTC or Web Real Time Communication is a technology to enable plugin-free browser-to-browser communication.

The audio, video media flows or data channels are peer-to-peer and encrypted. Unlike SIP systems, the signaling in WebRTC is outside the scope of the browser, and every website can implement its own signaling.

Unfortunately, the technology is not available on all browsers or devices yet.

When using WebRTC in enterprises, there are some important problems to solve.

Enterprise + WebRTC

Problem #1

Competes with existing communication systems

"Do I have to learn yet another system?"

First, any new enterprise communication system tends to compete with existing systems, and users are often reluctant to adopt the new one unless it can seamlessly integrate with existing systems.

Secondly, enterprises often restrict communication across their border, particularly those which cannot be understood or analyzed, or are peer-to-peer.

Problem #2

Traversal through enterprise firewalls
for encrypted + peer-to-peer flows

A WebRTC flow cannot be distinguished from other malicious or unwanted communication by the border element. So, restricted enterprises end up blocking all such UDP traffic.

However, the real problem is that such WebRTC flows cannot be associated with the identity of an internal user.

Real problem #2

Policy enforcement per user's identity
But which identity?

"My name is John Locke. I am called the Smoke Monster on Facebook."

If the flow can be associated with a user, then better policy enforcement can be applied - for example to allow WebRTC flows for some previleged users, or on some approved external web sites, or only during certain hours.

In this paper presentation, I will show how a browser extension can solve these problems, and why is it needed?

My co-authors are John and Alan. Alan is actually present here in this conference.

We use browser extensions (1) to integrate WebRTC-centric communication with existing enterprise systems such as click-to-call from corporate directory, and (2) to traverse media flows across network border and to apply policies.

I will show demonstrations of two systems that I implemented. The first is called Vclick. It is a web-based video collaboration application that allows the users to quickly and easily initiate conversation. It also integrates with our existing enterprise systems such as corporate directory and voice-over-IP systems.

Let me show a video demonstration of Vclick browser extension.

Vclick

Click to see a demo video

Once installed, user can configure its options. When configured, it can register this user id to receive call as long as the

browser is open. The blue extension icon indicates online.

Incoming call is shown using desktop notification. It is clicked to answer or closed to decline.

I don't have write access to this corporate directory website, but the browser extension can modify the displayed page to inject a click-to-call icon whenever it detects an email address.

The color of this icon indicates that this user is online. Right click menu shows various ways to reach the user. Or clicking the icon initiates the default video call.

This is a two-party Vclick call between the two machines.

The click-to-call icon with presence is very generic and can apply to any web page.

I will invite another user in my existing call. And make it a three party video call between three of my machines.

The browser extension can also inject click-to-call for phone numbers. You can select a telephony gateway in Vclick configuration. Then the extension detects and modifies any phone number in third-party web pages.

This icon can be clicked to initiate a voice call to that phone number via the selected gateway.

A browser extension is required in Vclick for two things - for user presence so that the user can receive a call whenever the browser is open - and for modifying third-party websites.

Why browser extension?

1. User presence when the browser is open
2. Inject click-to-call without help from website

Many people confuse between a browser plugin and a browser extension. A browser extension can modify the browser, and any visited web page. It can also intercept browser's JavaScript API.

This is used in the next system called secure edge. It allows secure and approved traversal of media flows across network border.

It has two parts - a secure media relay that sits on the border or in the public Internet, and a browser extension that injects the relay on all media flows, even if happening on third-party websites.

The enterprise firewall is configured to block all unwanted UDP traffic unless through this relay.

Enforcing policies on media flows of third-party websites requires injecting an intermediate element such as the relay in all such flows. But that requires changing the signaling data exchanged between the two browsers.

Earlier I mentioned that every website can implement its own signaling. If the signaling data is encrypted, existing border elements such as VoIP SBCs cannot modify it and hence cannot inject the media relay as pointed out in 1.

However, the browser extension in 2 that sits between the web application and the browser can intercept the JavaScript APIs, and alter those SDP and transport address values in the API. It can then inform those values to the border element for secure and approved border traversal of such media flows.

In this video demonstration of the secure edge browser extension, I will show how the browser extension can prevent leaking private IP address and how it can inject our enterprise media relay in WebRTC flow on third-party websites.

The options by default allow all types of ICE candidates or transport addresses as if the extension is not active.

In that case websites like ipleak.net can detect private IP address 192 range using WebRTC APIs.

Let us see what happens to a WebRTC call on third-party website such as Google's apprtc demo page. The webrtc-internals page shows that the WebRTC peer connection uses local ICE candidate addresses.

In secure edge configuration, I will now configure my local candidate to be some dummy IP address like 10.1.1.3 to hide my actual private address in 192 range. I will also remove any STUN servers, and I will force use my TURN relay server.

Now ipleak.net cannot detect the real 192 range private IP address but only the faked one in 10 range, and the relay address.

For the WebRTC call on apprtc page, webrtc internals page now shows that the peer connection uses the injected relay address. Note that this is done without help from the apprtc website.

The secure edge extension can also be used to inject user identity. The list of all peer connections on any browser tab is captured and shown here along with the remote identity if available.

Additionally, media devices can be disabled. Now any third-party website will not be able to use the camera in its getUserMedia API.

There are many other things possible with the extension, e.g., to add a fallback relay server or restrict the local host addresses range. This demonstrations showed a flavor of how the extension can modify the APIs to alter behavior independent of the website.

Secure Edge

Click to see a demo video

A browser extension is required in the secure edge system to intercept and modify WebRTC APIs on any third-party website. It

also allows such customization in the endpoint rather than at a server.

Why browser extension?

1. Intercept and change WebRTC APIs of any website
2. Endpoint driven customization and control

There are many other examples shown in the paper that benefit from a browser extension. It can fill the gap of missing user identity implementations in browsers. It can distinguish between the user identity supplied by the web site vs. his enterprise identity which is usually different from third-party website account such as your Facebook identity vs. enterprise email address.

An enterprise may want to allow WebRTC on some websites but not all. The extension can detect the web page origin and send it to the border element for such access control.

Although WebRTC does not have a notion of a call, one can use heuristics to co-relate WebRTC connections on the same page or pages of the same web site to be in the same call. This helps in call logging and accounting requirements of enterprises.

Recording of all conversations is particularly important in some corporate sectors such as banks. Furthermore, server side recording is desired for all such flows.

The paper has the details on how this is done using a browser extension.

The basic idea is to intercept various WebRTC APIs such as create offer or set local description. And modify the SDP and ICE candidates. In this example, the two web applications see their sessions as X and B, whereas the two browsers use A and Y, and X and B respectively.

In a way the browser extension injects the media server as the man-in-the-middle of the media flow, without the knowledge of the JavaScript web applications or the web server.

Unlike the previous demo that injected a media relay, this one injects a media proxy server that can decrypt and re-encrypt the media flow, and apply any kind of media processing to the flow including recording.

I have shown that a browser extension is needed to intercept APIs on third-party web pages. "Third-party" is the key. It is not needed if you can modify the web pages at the website. For a contact center if you own the contact center website, you can modify the web pages to connect user's browser to your own media server for recording, without a browser extension.

Why is a browser extension needed?

To intercept APIs on webpages of third-party websites

The paper also lists several other senarios that require a browser extension. Although Google Chrome has screen or app sharing in WebRTC, it can only be enabled from a native app or using a browser extension, but not in a pure web page.

When a user navigates away from a web page, its WebRTC peer connection is terminated, but can be preserved by moving the peer connection to a persistent browser extension.

An extension can force certain device or media parameters, e.g., if you have an HD camera, you can force HD capture attributes.

1. Screen or app sharing
2. Persistent peer connection on navigation/reload
3. Alter media (HD, G.711) irrespective of webpage

You can change the display size or pop-out the video to separate tab or window.

An extension can also improve privacy, e.g., by disabling data channel APIs or hiding VPN IP addresses.

4. Pop-out video streams to separate tab/device, any size
5. Disable data channel, hide VPN IP address, etc.

However, there are limitations to what a browser extension can do. Basically the extension runs on top of JavaScript. So it can only alter behavior that can be done in JavaScript. But it cannot do things that cannot be done in JavaScript. For example, it cannot do speech recognition or alter video frames of a media flow.

Limitations of browser extensions

Can alter only if available in JavaScript

No speech recognition, synthesis or video frame alteration

I have shown two systems based on browser extensions and several other examples of how the extension can help adopt WebRTC in enterprises.

The first one, named Vclick, allows seamless integration of WebRTC with existing systems.

Vclick's design principles

1. Separation of call initiation from conversation
2. Application logic in the endpoint

But the focus of the paper is on the second system, secure edge.

Note however that secure edge must work together with firewall rules to block any other unwanted UDP traffic at the border, otherwise a user can easily by-pass the the policies.

SecureEdge design

1. Requires all flows through approved edge element
2. Inject this edge element for all websites

Although browser extension can solve certain problems, the bigger questions are as follows.

How can browser vendors provide necessary hooks for enterprise control of WebRTC.

And does it even matter if the user is bringing or customizing her own device for enterprise use.

Although WebRTC is missing of some mobile browsers, does it matter, given that many services prefer to have mobile apps rather than mobile web pages.

The bigger questions!

1. Should browser vendors provide hooks for enterprise control of WebRTC?
2. Does it even matter with BYOD and CYOD?
3. WebRTC on mobile - on browser or apps?

That concludes my presentation, on why and how to use browser extensions for enterprise WebRTC?

Enterprise + WebRTC
+ Browser Extension
Why? How?