Taking on WebRTC in an Enterprise

Alan Johnston, John Yoakum and Kundan Singh
Avaya Inc
Email: {abjohnston,yoakum,singh173}@avaya.com

***Abstract*—WebRTC, Web Real-Time Communications, will have a major impact on enterprise communications, as well as consumer communications and their interactions with enterprises. This article illustrates and discusses a number of issues that are specific to WebRTC enterprise usage. Some of these relate to security: firewall traversal, access control, and peer-to-peer data flows. Others relate to compliance: recording, logging, and enforcing enterprise policies. Additional enterprise considerations relate to integration and interoperation with existing communication infrastructure and session-centric telephony systems.**

Keywords-WebRTC; enterprise communication; firewall; media relay; SIP

I. Introduction

WebRTC [1][2][3], the industry effort to add real-time voice and video communication capabilities to browsers, is receiving much attention and hype these days. As browser vendors announce timelines for support and developers show off demos, it is clear that these new standards, Application Programming Interfaces (APIs), and protocols will have a major impact on the World Wide Web. Less often discussed is the impact on enterprises and comparisons to existing enterprise communications.

Typically, enterprise communication systems enable two-party phone calls or multiparty conference scenarios. The equipment enabling these session-centric client-server capabilities usually resides within the guarded enterprise network – e.g., behind the enterprise firewall or within a Virtual Private Network (VPN). In contrast, WebRTC opens up several compelling opportunities that go beyond the classical enterprise communication views of sessions and guarded networks. For example, team members visiting the same internal project webpage could auto-join a video conferencing application embedded in that page. As another example, customers visiting an enterprise website could initiate interactive conversations with its customer service agents – via voice or video – without asking the customers to call the enterprise using a phone. The agents – using WebRTC technology – participate in media flows from the browser rather than conventional session-centric telephony gear, making it easier for the agents to live outside the enterprise communication system boundaries. Numerous other examples related to collaboration or other enterprise value propositions that benefit from WebRTC are likely [4]. Some of these web-centric interactions will likely not have direct analogies in existing communications systems, given the ease of building rich user interfaces and experiences with HTML5.

Web developers working on consumer applications and websites will likely be able to use WebRTC directly as initially specified. Using fairly simple JavaScript code will result in new multimedia communication capabilities being embedded in their application or site in new and interesting ways. However, there are some open issues relating to enterprise adoption and use of WebRTC. This paper illustrates and discusses a number of these issues. Some of these relate to security: firewall traversal, access control, and peer-to-peer data flows. Others relate to compliance: recording and logging. Additional issues relate to integration and interoperation with existing communication infrastructure and session-centric telephony equipments.

Traditionally, the enterprise network enforces strict security requirements resulting in very restrictive firewalls and network border elements. On the other hand WebRTC strives to create an end-to-end secure media path between the two browser instances with little or no interference from any intermediate entity. This fundamental design principle poses several challenges to the traditional enterprise Information Technology (IT) mindset. Existing Voice-over-IP (VoIP) tools and techniques used by enterprises are not enough to deal with the challenges. Although, we propose potential directions to solve them, ultimately the industry will decide how it adapts this emerging technology in enterprises.

II. Enterprises and Firewalls

Enterprises use firewalls to enforce Internet Protocol (IP) access policies at the edge of their networks. These policies relate to who is allowed to access which sites and resources. Firewalls are often implemented using 5-tuple rules (source and destination IP address, source and destination ports, and transport protocol). Other firewalls utilize deep packet inspection to determine the application using the transport connection and its characteristics. Typically, firewall devices include both filtering and address mapping techniques – the latter is known as Network Address (and port) Translation (NAT). Conventional firewalls were developed mainly to handle client-server protocols such as web browsing, email, and file transfer. A client-server packet sent from inside the firewall to outside typically creates a pinhole at the firewall so that the packets in the reverse 5-tuple flow are not blocked. Peer-to-peer communication systems and protocols are a bigger challenge to firewalls and other policy enforcement devices.

Real-time communication flows of Real-time Transport Protocol (RTP) [5] packets are typically described as peer-to-peer flows. That is, they are often established directly between the two communicating devices. Servers are often used to help establish these flows, but routing the resulting media session through these servers is often undesirable for a number of reasons:

1. Real-time media flows are extremely sensitive to latency or delay. Peer-to-peer flows typically result in the lowest possible latency.

2. Direct peer-to-peer media flows often have fewer IP hops than relayed traffic, which results in a lower chance of packet loss.

3. Servers used for signaling are often not distributed geographically nor have enough bandwidth to be used successfully as media relays.

Peer-to-peer flows have been historically difficult to get through enterprise firewalls. During the early days of Session Initiation Protocol (SIP) [6] VoIP deployment and testing, this was frequently encountered in the form of the "one way media problem" where media could be sent out from inside the firewall, but the reverse flow media was blocked. This occurred in cases where the signaling was able to traverse the firewall, due to its similarities to client-server protocols.

A number of approaches were developed to make firewall traversal easier. They include:

1. Symmetric RTP [7]. A bi-directional media session is actually two uni-directional RTP flows. In particular, the user agent uses the same User Datagram Protocol (UDP) port for sending and receiving the RTP stream. Using symmetric RTP, it is possible to make these two RTP flows seem more like a single bi-directional flow which more easily traverses firewalls.

2. Interactive Connectivity Establishment (ICE) [8] which formalizes the "hole-punching" approaches developed by peer-to-peer gamers. This approach uses test packets sent by both participants in a session to establish filter rules in firewalls and also Network Address Translation (NAT) devices.

For enterprise firewall traversal of communication, the most used approach today involves a Session Border Controller (SBC).

III. Session Border Controllers and Firewall Traversal

A Session Border Controller or SBC [9] is essentially an application layer firewall with a signaling and media application layer gateway (ALG) built in. The SBC is usually connected in an enterprise Demilitarized Zone (DMZ) as a trusted enterprise network element. It blocks all unauthorized signaling and media flows, and provides a point of policy enforcement as shown in Fig.1. Today's SBCs support SIP and RTP, including Secure RTP (SRTP). SIP is a signaling protocol used for VoIP and video communication to establish RTP (or SRTP) media sessions. By parsing SIP messages, the SBC is able to discover the transport addresses (5-tuple) to be used for the media session. If the SIP traffic is authenticated and authorized, the SBC either opens a pinhole (filter rule permitting the RTP traffic) or activates an RTP relay. In both cases, the resulting RTP media session is able to traverse the firewall.

In addition to firewall traversal, SBCs also provide a number of other services, including protocol normalization, media transcoding, and protection against malicious packets and payloads – these features are not directly related to the firewall traversal problem discussed here but highly useful in enterprises.

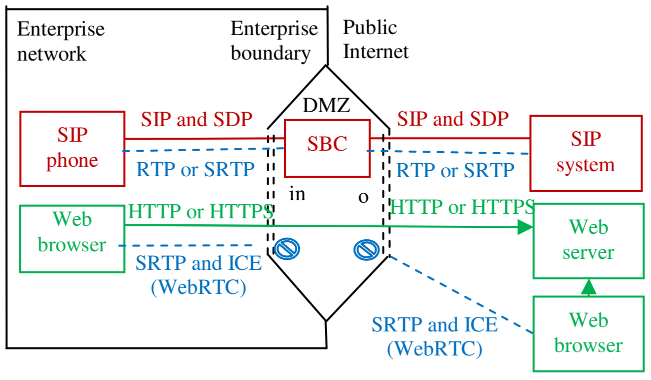

Figure 1. The SBC intercepts SIP/SDP signaling on port 5060, applies policies and opens firewall pinholes to allow RTP media path for enterprise communication, but WebRTC traffic uses Hypertext Transfer Protocol Secure (HTTPS) which is not intercepted and hence the media path is blocked.

Since enterprises have widely deployed SBCs for communication firewall traversal, it would seem logical to re-use them for WebRTC. However, there are a number of problems in this approach. Firstly, it relies on using the signaling channel to authenticate the media channel. With WebRTC, there is no standard signaling channel. Secondly, SBCs rely on inserting themselves into the control path to learn when media flows are beginning and ending. With WebRTC, the control path is the Hypertext Transfer Protocol (HTTP) or WebSocket channel between the browser and web server and will be running over Transport Layer Security (TLS) and hence be encrypted and not available to observe as shown in Fig.1. Finally, SBCs use an identity determined from the signaling channel to authenticate the media channel. In WebRTC, there is no standard way to indicate identity. The few identity mechanisms that have been discussed in standards bodies are related to the identity in the media path, and this identity is of a different nature than what SBCs are accustomed to dealing with.

In addition, WebRTC has no concept of "sessions" – instead, it has a concept of "streams." Streams have media sources and sinks that generate and consume media flows. They are created and manipulated using JavaScript, resulting in Peer Connections being established. Streams can be created for media to flow point-to-point or between a browser and a media server or mixer. There is no direct correspondence between Peer Connections and participants in a multi-party session. Once a Peer Connection has been established between a browser and a media selector, additional participants can be added at any time. For example, media from multiple participants can be mixed or offered by the media mixer or selector.

One possible approach would be for an enterprise to attempt to convert every WebRTC session that crosses enterprise boundaries into a communication session that its existing infrastructure could handle in a session-centric manner. This could, for example, mean converting a WebRTC session into a SIP session, applying policy and authentication, then converting back again to provide to the other browser (Fig.2). This type of man-in-the-middle approach is not practical, especially since WebRTC capabilities can be embedded in many types of web applications and sites, and many of them follow a very different paradigm of real-time communication applications and services. In particular, a one-to-one translation from a WebRTC stream to a SIP session is not trivial without breaking existing SIP implementations. In addition, all control and media in WebRTC are encrypted.

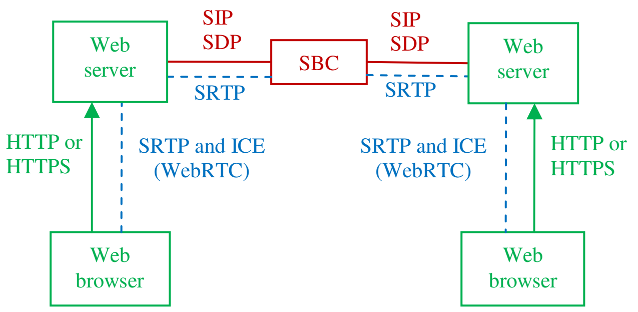

Figure 2. Converting a WebRTC media flow into a SIP/SRTP to allow policy enforcement by a SBC is possible for a simple call or conference but does not work in many web-centric scenarios where a one-to-one translation between a WebRTC media flow and SIP session is not trivial.

These problems do not mean that WebRTC will never cross enterprise firewalls, but rather that new approaches must be used. Some of these approaches will likely be novel, and discovered through actual deployment and rollout of WebRTC. However, there are some indications of directions this may take in today's standards and approaches. The rest of this paper presents ideas to answer three important questions: How can the enterprise firewall adapt to WebRTC? How can the enterprise detect and apply policy to WebRTC flows? And, how can the emerging WebRTC applications integrate and interoperate with existing enterprise communication equipment?

IV. WebRTC Firewall Traversal

From the previous discussion, it is clear that WebRTC firewall traversal must work without a standardized signaling protocol, without a conventional signaling identity, and without a concept of sessions that can be managed or controlled. This might seem like a daunting task, but there are some potential options.

It should be clear that the term Session Border Controller is not applicable for the element which assists WebRTC in traversing enterprise firewalls. While the border part may be accurate, the session part is not applicable, and any notion of controlling WebRTC streams without access to a signaling channel is also not realistic. In the following discussion, we will refer to the network element enabling enterprise edge traversal as a Secure Edge, just so we can have a label.

There are a number of ways in which this Secure Edge might be designed.

A. *Detect ICE exchange*

For one, there is a type of standard signaling protocol used for establishing media flows defined as part of WebRTC. It is built into the Interactive Connectivity Establishment (ICE) protocol used for hole-punching. Before any media data flows, ICE will be run between the two browsers. The Secure Edge could detect when an ICE exchange is starting up across the enterprise border. This could be used to distinguish a WebRTC media flow from a random packet flow across the border, allowing policy to be applied.

In addition, there is a username/password fragment in ICE Session Traversal Utilities for NAT (STUN) messages. Normally, this information is randomly generated. If this information were generated in a particular way, or coordinated with an enterprise authentication system, the Secure Edge could authenticate the ICE exchange and hence the resulting media flow. One proposal for how to do this is described in [10]. In WebRTC, ICE is run by the browser. While the JavaScript has access to the username/password fragment, there is currently no standard way in the JavaScript to set this media flow identity. It is possible a browser plug-in, utilized by the enterprise, could be used to set this identity.

B. *SRTP key negotiation*

Another approach might be to use the SRTP key negotiation along with Datagram Transport Layer Security (DTLS) to authenticate the media flow. For example, if DTLS-SRTP is used for key management, the Secure Edge could act as a man-in-the-middle and hence validate the public key in the fingerprint. If the enterprise deployed a Public Key Infrastructure (PKI), then the certificate could be checked and validated. A self-signed certificate could also be uploaded from the browser and stored in an enterprise key server. This could allow the Secure Edge to authenticate the browser inside the enterprise and apply appropriate policy. However, if some form of end-to-end identity is used in WebRTC, then this man-in-the-middle approach would look like an attack and would be detected and the user alerted.

If the media is being relayed by a man-in-the-middle, then this also provides a place where recording could take place. Note, however, that the context of the media flow would not be available. That is, the identity of the remote party in the media flow or the application or website which enabled this media flow would not be known from this insight alone.

C. *Media relay*

Another approach would require a media relay to be used for WebRTC media sessions crossing the enterprise boundary. There are standards for this media relay, known as a Traversal Using Relays around NAT (TURN) [11]. ICE hole-punching begins with each browser gathering candidate addresses: addresses that might be useful in routing incoming media packets to the browser. Typically, this includes private IP addresses determined by reading the Network Interface Card (NIC) or equivalent on the user device, and public IP addresses determined by a STUN response packet from a STUN server. In addition, a TURN candidate address can also be provided to be used as a last resort (i.e. the lowest relative priority of all the candidates).

The enterprise firewall would be configured to block non-relayed WebRTC media flows. The enterprise would deploy a TURN server in the DMZ, and permit media flows which go through this server as shown in Fig.3. The enterprise would issue individual credentials to use the TURN server. A user would enter their TURN credentials into the browser, which would then use them to gain a TURN candidate address. During media flow setup, the TURN server could authenticate the user and also learn what type of media flow is to be setup based on the bandwidth requested. Some policy could then be applied to this media flow.

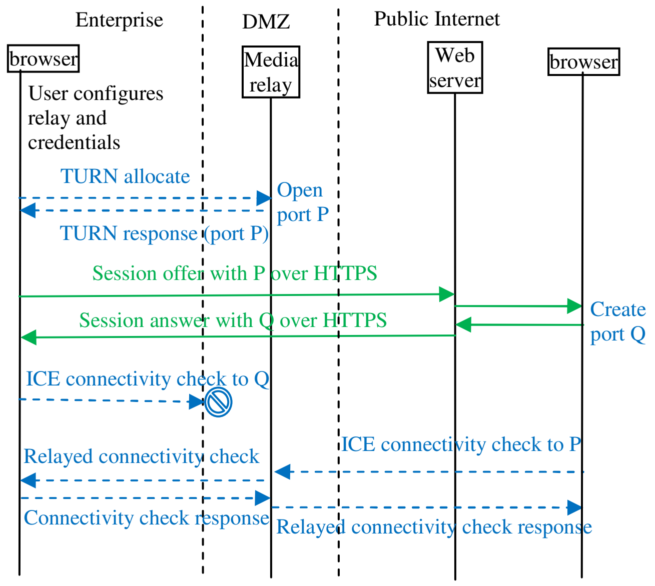
Enterprise
DMZ
Public Internet
browser
Media relay
Web server
browser
User configures relay and credentials
TURN allocate
Open port P
TURN response (port P)
Session offer with P over HTTPS
Create port Q
Session answer with Q over HTTPS
ICE connectivity check to Q
ICE connectivity check to P
Relayed connectivity check
Connectivity check response
Relayed connectivity check response

Figure 3. Flows using a media relay in the DMZ to establish a media path with WebRTC.

Note, however, that the exact context of the media flow is inherently unknown. This approach also relies upon the browser being able to be configured with the user's TURN credentials. In some ways, this is similar to the configuration of a web proxy, used by some enterprises to monitor and control web browsing.

The TURN server is also an ideal place to perform recording, according to an enterprise policy. Again, without the cooperation of the website, no media flow context could be determined. Also, since the media is encrypted end-to-end, the browser or the web application would need to share the encryption key in order for the media to be played back.

D. *Firewall conscious applications*

The previous approaches deal with WebRTC transparently to the application. Another path of evolution might be that the enterprise web applications consciously work in conjunction with the Secure Edge. This is similar to how some gaming applications use Universal Plug and Play (uPnP) to configure firewall holes. The Secure Edge might have a web interface, which is used by the enterprise application developer to explicitly authenticate the end user and request permission from the Secure Edge. Thus, the recording is handled by the application developer in the browser, and later uploaded to the Secure Edge or other enterprise-specific storage, instead of transparently recording the conversation at the border device. The application developer might also deliver all the control messages of JavaScript to the Secure Edge device so that it can authenticate, install session context and open pinholes as shown in Fig.4. Such a device would block all WebRTC traffic unless the application has explicitly used its API to install a media flow context.

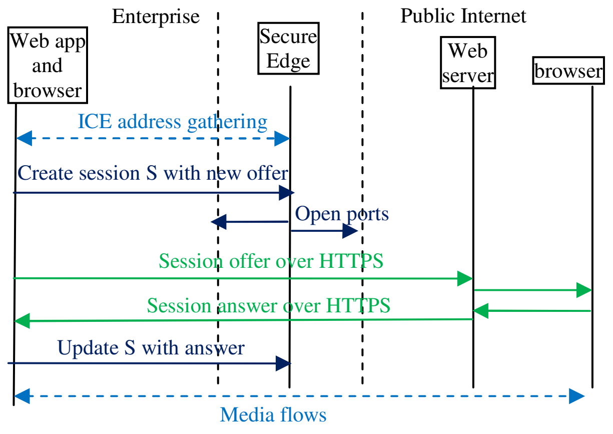

Figure 4. A firewall conscious web application voluntarily delivers the media flow context to the Secure Edge which opens the firewall pinholes.

The main advantage of this approach is that the application developer voluntarily delivers the media flow context to the Secure Edge. This approach would require standardization of a Secure Edge API.

V. Policy Compliance

Enterprises often require policy compliance such as recording and logging of all conversations. Some require selective authorization of certain call destinations or websites based on who is involved in the interactions. In particular, records of past emails, memos, instant messaging, calls, and even logs of visited websites are useful during auditing and sometimes in legal proceedings. The usefulness of such insight often requires enterprises to co-relate records with the person who sent or received a piece of information or placed or received a call.

Various examples of where policy could be applied have been mentioned in the examples above. For all of the approaches described in the previous section except the last one, there is no known way for the Secure Edge to know the full context of a media flow to apply policy. For example, even knowing which web site or application is originating the flow is difficult, as the user may have multiple open tabs and browsers running, and any of the sites could be WebRTC enabled. If per-site WebRTC blocking is desired, the entire site might need to be blocked using, for example, the enterprise proxy.

A. *Transparent at border*

From the previous discussion, it is clear that due to the end-to-end encryption and security of a WebRTC media path, transparently recording media conversations at the Secure Edge is not feasible without acting as a man-in-the-middle.

Unfortunately, this could prompt the end user with security warnings and cause annoyance.

B. Mangle web JavaScript

A web proxy could intercept and filter out any WebRTC-related JavaScript code, or enforce use of a media recording API. This would likely interfere with proper operation of the website, and could perhaps render the entire site unusable or unpredictable in behavior. Secondly, this approach is either too difficult or impossible to implement in practice, as it involves detecting related code in obfuscated or dynamically generated JavaScript, and intercepting web pages delivered over TLS.

VI. Integration and Interoperation

WebRTC deviates from traditional enterprise communication in two ways – it opens up communications from every web application instead of controlled software pieces installed by the IT department, and it bypasses the existing enterprise communication infrastructure typically enabled in its modern form by the SIP family of standards. The integration and interoperation of WebRTC with legacy SIP devices could happen in the end-user's browser (client) or via a translation gateway (server).

A. Interoperation

To connect with a SIP device from the browser, one could implement the SIP stack in JavaScript running in the client, use SIP-over-WebSocket transport, and use an end-to-end media path enabled by WebRTC. However, WebRTC includes several new profiles and extensions in the media path that are typically not implemented in existing SIP devices, e.g., multiplexing RTP and Real-time Transport Control Protocol (RTCP), audio and video streams on the same UDP port, mandatory ICE negotiation attributes, mandatory SRTP, and the proposed mandatory audio and video codecs. Interoperating with existing SIP devices that do not support some or all of these features requires an intermediate gateway that has access to the signaling as well as media paths. Alternatively, the SIP devices need to be "upgraded" to support all the new features.

Upgrading existing SIP phones, both hardware devices and softphones, is not trivial. There are several potential paths this could evolve into. For example, existing SIP phones could be reprogrammed to be able to participate in WebRTC media flows, remain unchanged but interoperate with WebRTC via a gateway, or be superseded by a new generation of phones that employ various web-centric enablements such as HTML5 and WebRTC. The benefit and cost analysis for such transitions is an open issue for many enterprises and device vendors that are considering WebRTC adoption. Besides the technical differences listed above, there is a crucial behavior difference in the way a SIP phone or a WebRTC enabled application is expected to work. In particular, people primarily expect a phone to make or receive calls, which is a single primary application. However, people expect to use WebRTC to enable real-time communications inside whatever they are already doing or want to do on the web. It remains to be seen how the industry shapes the future of existing SIP devices while adopting WebRTC.

Translating between SIP/RTP and WebRTC at a gateway appears to be a viable short term solution for enterprises and device vendors alike. Moreover, such a gateway server can work with the proposed Secure Edge to integrate authentication and firewall traversal between the two protocols. Due to differences in offer-answer state machines, it is not trivial to blindly forward session description between SIP and WebRTC endpoints. Nevertheless, the gateway can terminate and originate media flows on each side to perform translation as necessary similar in spirit to a back-to-back user agent.

B. Integration

Integration of WebRTC enabled applications in an enterprise is a broader issue – of which interoperation with existing communication system is only one part. The larger perspective deals with how WebRTC will change the way enterprise does business – either within the organization or outside with customers or other organizations. The emerging trend to bring your own device to work has opened the enterprise network to some extent, and such devices may not be subject the same set of strict policy enforcement. As the trend of moving everything to the cloud continues, WebRTC can bring the communication primitives to these cloud hosted web applications, e.g., problem tracking and resolution systems, information repositories, customer relations tools, corporate directories, social media interactions, blogs, and so on. Initially, enterprises will likely take measures to primarily interoperate between existing communications systems and WebRTC-centric applications, while keeping core communications functionality in SIP. Eventually, a few powerful web-centric enterprise applications will likely emerge that have the potential to move WebRTC technology to a more central position in enterprises.

WebRTC not only brings voice and video flows to web applications but also allows sharing generic data interactively. For example, a browser could expose the user's desktop as a local media stream so that any web application can enable desktop sharing. These applications diminish the differences between a communication specific device and a generic web and computing device. Such interactive data sharing brings new threats in relation to enterprises.

C. Peer-to-peer data flows and file transfers

WebRTC also includes enablers for interactively exchanging data and files in conjunction with real-time media flows. This capability is highly attractive to gaming applications and possibly many web applications. Such transfers will likely be perceived by enterprises as introducing significant threats. In contrast to media flows that are processed by codec technology that has defined expectations and can ignore or discard unexpected content, data flows can be unstructured and used in many ways. In general, threats related to media flows involve exploiting some flaw in the media processing technology. In contrast, interactive data flows can more easily contain viruses or other malware since they are not processed in a specific way.

Requests for media flows are polite in that they always ask the user for permission to use a resource like the microphone or camera in a flow. The current WebRTC specifications do not ask or alert the users in any way that a data flow is going to happen. This characteristic alone likely makes such data flows as part of WebRTC interactions appear as more of a threat in an enterprise context than the media flow itself for both network protection as well as intellectual property reasons. Beyond network threat vectors, enterprises have concerns about what intellectual property leaves or enters the enterprise. Undetectable leakage of valuable intellectual property owned by an enterprise is an easy to understand consideration. Less obvious but also important are possible allegations or claims that may arise over intellectual property owned by someone else that enters an enterprise, especially if the entry is not well documented.

It is likely interactive data flows will have to be detected and subjected to enterprise policy, possibly in a different manner than interactive media flows. At least initially, until it is clear exactly how to handle interactive data flows across enterprise network boundaries, some enterprises may want to simply restrict WebRTC to media flows only. At this point in the maturity of WebRTC, detailed discussion of this topic is beyond the scope of this introductory article.

VII. Conclusion and Future Work

This paper has begun to look at enterprise requirements for permitting WebRTC media flows to cross enterprise boundaries. The existing state-of-the-art Session Border Controllers will not work with WebRTC, and many of their principles do not really apply to WebRTC. A number of potential partial solution approaches have been outlined and discussed. The analysis in this paper shows that while there are some promising potential approaches, the design of a Secure Edge to permit enterprise authorization and application of policy to WebRTC traffic is far from solved today.

In the future, if an enterprise treats web-based rich media interactions differently than existing VoIP, it may not apply the same set of strict policies to WebRTC. In such cases, web site level filtering may be enough. On the other hand, the IT department may treat it as a threat to enterprise security and compliance. Ultimately, the benefits and desirability of WebRTC interactions across the enterprise boundary will result in solutions to this problem.

We are currently investigating various ideas in the application of enterprise policy in a WebRTC context in ways that are significantly different from how policy is applied in session-centric communications technologies. Publication of the concepts related to our current enterprise policy research is presently under consideration, placing discussion of that research beyond the scope of this article.

References

[1] A.B. Johnston and D.C. Burnett, WebRTC: APIs and RTCWEB Protocols of the HTML5 Real-Time Web, Digital Codex, 2012, ISBN 978-0985978808

[2] WebRTC 1.0: Real-Time Communication Between Browsers, W3C Working Draft, Aug 2012, http://www.w3.org/TR/webrtc/

[3] Real-Time Communication in WEB-browsers (RTCWEB) IETF working group, http://tools.ietf.org/wg/rtcweb

[4] A.Lepofsky, "Social Business 2013: Less Talking, More Doing", Constellation Research blog, Dec 2012, http://www.constellationrg.com/blog

[5] H. Schulzrinne et al, “RTP: A Transport Protocol for Real-Time Applications,” RFC 3550, 2003

[6] J. Rosenberg et al, “SIP: Session Initiation Protocol,” RFC 3261, 2002

[7] D.Wing, "Symmetric RTP / RTP Control Protocol (RTCP)," RFC 4961, 2007

[8] J. Rosenberg, “Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator Traversal for Offer/Answer Protocols,” RFC 5245, 2010

[9] J. Hautakorpi et al, “Requirements from Session Initiation Protocol (SIP) Session Border Control (SBC) Deployments,” RFC 5853, 2010

[10] T.Reddy et al., "STUN Extensions for Authenticated Firewall Traversal", IETF, work in progress, draft-reddy-rtcweb-stun-auth-fw-traversal

[11] R. Mahy et al, “Traversal using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN),” RFC 5766, 2010