RTC-Web Network Address Translation
draft-cbran-rtcweb-nat-00

Abstract

This document outlines the network address translation (NAT) mechanisms and requirements for RTC-Web client applications.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, and it may not be published except as an Internet-Draft.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 2, 2012.

Copyright Notice

Copyright (c) 2011 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.
Table of Contents

1. Introduction .................................................. 3
2. Terminology .................................................... 3
3. Connection Management Requirements .......................... 3
4. ICE for Web Browsers Via a JavaScript Library ............ 4
   4.1. ICE Timing and Pacing Requirements .................... 4
   4.2. Compatibility, Fixes and Update Rollout ............... 5
5. Negotiation Architecture ................................. 6
6. Legacy VoIP Interoperability ............................... 6
7. IANA Considerations ........................................ 6
8. Security Considerations .................................... 6
9. Acknowledgements .......................................... 7
10. Normative References ...................................... 7
Authors’ Addresses ........................................... 7
1.  Introduction

An integral part of the success and adoption of the Real-Time Communications Web (RTC-WE web) will be the ability for RTC-Web applications to have native, secure Network Address Translation (NAT) traversal capabilities. This specification proposes NAT traversal requirements and implementation specification for RTC-Web client applications.

The NAT requirements fit into a series of specifications have been created to address RTC-Web codec, security, data transmission, non-media data, signaling and negotiation and use case requirements. More information on the RTC-Web can be found here:

[TODO put links to supporting drafts here]

2.  Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

3.  Connection Management Requirements

It is quite probable that many RTC-WE client applications, such as web browsers will be deployed behind a NAT. To set up secure data plane sessions, all RTC-WE client application implementations are REQUIRED to implement ICE [RFC5245] or ICE-Lite Section 2.7 of [RFC5245]. Implicit to supporting ICE, all RTC-WE client applications are REQUIRED to implement Simple Traversal of User Datagram Protocol (UDP) Through Network Address Translators (NATs) (STUN) [RFC3489] and Traversal Using Relays around NAT (TURN) [RFC5766].

There are two deployment scenarios for RTC-WE client applications. The first scenario is when applications are deployed behind NAT and have to worry about NAT traversal. The second scenario is when the application is not behind a NAT, such as an RTC-WE application that is always connected to the public Internet. As stated in section 2.7 of [RFC5245], ICE requires that both endpoints to support it in order for ICE to be used on a call.

With regards to RTC-WE client applications that are deployed behind a NAT or do not have a public IP address are REQUIRED to support ICE [RFC5245], applications that are not behind a NAT and have a public IP address are REQUIRED to support ICE-Lite and MAY fully support...
ICE. RTC-WEB client applications that fully support ICE are REQUIRED to support AGGRESSIVE NOMINATION, and MAY support REGULAR NOMINATION.

[Open Issue: there is a strong interest to define a TURN-like protocol that looks like HTTP to intermediaries, so that media can be tunneled over HTTP. Should this be done?]

4. ICE for Web Browsers Via a JavaScript Library

There have been discussions regarding the responsibility of where ICE will be implemented this is best illustrated via a concrete example. Given that the dominant RTC-Web application will be the web browser, it has been proposed that the ICE implementation reside within a JavaScript library and not natively available within the web browser.

The reasoning behind requiring RTC-Web web apps to use a JavaScript library for ICE negotiation falls along two primary assumptions.

1. Modern JavaScript engines can handle the ICE timing and pacing requirements

2. JavaScript libraries provide the best deployment strategy for maintaining compatibility and versioning

4.1. ICE Timing and Pacing Requirements

The ICE pacing requirements have a lower bound of 20 ms [RFC5245, section B.1. Pacing of STUN Transactions]. At the writing of this document it is unclear if the resolution of modern JavaScript timers across the major operating systems could meet the lower boundary requirements for ICE. It has been suggested that the best way to determine if the ICE timing and pacing requirements were actually feasible was to create browser ready sample applications could prove or disprove the feasibility of ICE as a JavaScript library.

If and when the testing is performed, there are several factors that have to be taken into consideration. The first being the testing environment. The testing environment must represent a real world user’s environment as close as possible. A partial listing of user environments to consider for JavaScript/ICE testing would be the operating system, virtualization, browser vendor selection, hardware platform (notebook, desktop, tablet, netbook, smart phone, etc) and network connectivity.

In addition to the underlying hardware, operating system, browser software and network, a crucial piece of testing JavaScript ICE MUST include testing performance under real-world web page conditions.
Inline advertisements in web pages are a commonplace on the web. A page with advertisements may also include long-running JavaScripts that prevent web application timers from firing in correctly. Handling long running JavaScripts while meeting the ICE pacing requirements should be part of the evaluation criteria.

[TODO - is someone going to write sample code that can prove, disprove the timer issue?]

4.2. Compatibility, Fixes and Update Rollout

It has been proposed that JavaScript ICE libraries would be easier to manage with regards to compatibility and updates when compared to ICE native within the web browser. While JavaScript libraries would make it easy to add fixes and enhancements to an ICE implementation this approach will not scale when it comes to interoperability and rapid deployment. With ICE as a JavaScript library, there can literally be a copy of the library on a per website basis, given that there are over 250 million individual websites on the internet, in addition to the millions of intranet hosted sites, upgrading a JavaScript library will simply not scale in a time friendly manner.

With ICE native within the browser, there are fewer than a dozen implementations world wide that have to interoperate with each other, which means that enhancements to ICE can be coordinated between browser vendors. When it comes time to enhance or fix a defect with the browser’s native ICE implementation, updates to browsers can be deployed, at scale, to hundreds of millions of users in the span of a few weeks. The rapid updates have proven effective and most if not all the major browser vendors have short term update mechanisms. Given that web browsers will be the dominant RTC-Web endpoint and that a native implementation of ICE within the browser will significantly narrow the complexities of ICE interoperability, defect fixes and enhancements at scale it is RECOMMENDED that ICE be implemented natively within all RTC-Web client applications.

A question may arise regarding the above recommendation if a JavaScript ICE library could meet the ICE performance requirements. While such a library may meet the ICE performance requirements, until a deployment solution is proposed to propagate bug fixes and enhancements to the JavaScript library at internet scale, a JavaScript library approach would be an inferior recommendation compared to the native in the browser approach.

[NOTE: This recommendation is based on current criteria and is subject to change should new criteria or techniques be discovered during the working of this draft]
5. Negotiation Architecture

[WORK IN PROGRESS] An example of this will be showing how a RTC-Web capable web browser that natively supports ICE does signaling and negotiation to set up a DTLS [REF] connection. Once the DTLS connection has been established, the RTC-Web client application will use the secure channel for SIP signaling and media transmission.

[OPEN issue - add architecture diagram and content]

6. Legacy VoIP Interoperability

There is no way to meet all the security requirements and maintain comparability with all legacy VoIP equipment. This draft tries to minimize the impedance mismatch. The requirements here would allow interoperability with legacy VoIP equipment as long as that equipment either directly supported, or was fronted by an SBC that supported ICE or ICE-Lite.

Support for ICE-Lite has historically been lacking in VoIP equipment, this is changing and ICE-Lite becoming increasingly prevalent, particularly on devices designed to sit on the edge of a domain and connect to remote user agents that may be behind NATs. Given the increasing adoption of ICE-Lite, it could be conjectured that a substantial fraction of VoIP equipment meets the RTC-WEB interoperability list.

7. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

8. Security Considerations

Because there are a number of security issues, considerations and requirements for RTC-WEB client applications there is a draft that specifically addresses the RTC-WEB application security considerations. This draft defers its security considerations and requirements to the security considerations for RTC-Web draft [I-D.ekr-security-considerations-for-rtc-web].
9. Acknowledgements

This draft incorporates ideas and text from the IETF mailing list. In particularly we would like to acknowledge, and say thanks for, work we incorporated from Timothy Terriberry, Jonathan Rosenberg and Christopher Blizzard.

10. Normative References

[I-D.ekr-security-considerations-for-rtc-web]


Authors' Addresses

Cary Bran
Cisco
170 West Tasman Drive
San Jose, CA 95134
USA

Phone: +1 206 256-3502
Email: cbran@cisco.com