Internet Engineering Task Force
Internet-Draft
Intended status: Informational
Expires: December 29, 2013

T. Stach, Ed.
A. Hutton
Siemens Enterprise Communications
J. Uberti
Google

June 27, 2013

RTCWEB Considerations for NATs, Firewalls and HTTP proxies
draft-hutton-rtcweb-nat-firewall-considerations-01

Abstract

This document describes mechanism to enable media stream establishment for Real-Time Communication in WEB-browsers (RTCWEB) in the presence of network address translators, firewalls and HTTP proxies. HTTP proxy and firewall policies applied in many private network domains introduce obstacles to the successful establishment of media stream via RTCWEB. This document examines some of these policies and develops requirements on the web browsers designed to provide the best possible chance of media connectivity between RTCWEB peers.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

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 December 29, 2013.

Table of Contents

1. Introduction 2
1.1. Requirements Language 3
2. Considerations for NATs/Firewalls independent of HTTP proxies 3
2.1. NAT/Firewall open for outgoing UDP and TCP traffic . . . 3
2.2. NAT/Firewall open only for TCP traffic 4
2.3. NAT/Firewall open only for TCP-based HTTP(s) traffic . . 4
3. Considerations for NATs/Firewalls in presence of HTTP proxies 5
3.1. HTTP proxy with NAT/Firewall open for outgoing UDP and TCP traffic 5
3.2. HTTP proxy with NAT/Firewall open only for TCP traffic . 5
3.3. HTTP proxy assisted TURN server connection 5
3.3.1. TURN server connection via TCP 5
3.3.2. TURN server connection via UDP 7
4. Other Approaches 7
4.1. TURN server connection via WebSocket 7
4.2. Port Control Protocol 7
4.3. HTTP Fallback for RTP Media Streams 7
5. Requirements for RTCWEB-enabled browsers 8
6. Acknowledgements 8
7. IANA Considerations 8
8. Security Considerations 8
9. References 9
9.1. Normative References 9
9.2. Informative References 9
Authors' Addresses 10

1. Introduction

Many organizations, e.g. an enterprise, a public service agency or a university, deploy Network Address Translators (NAT) and firewalls (FW) at the border to the public internet. RTCWEB relies on ICE [RFC5245] in order to establish a media path between two RTCWEB peers in the presence of such NATs/FWs. As last resort in order to cater for NAT/FWs with address and port dependent filtering characteristics [RFC4787], the peers will introduce a TURN server [RFC5766] in the public internet as a media relay. Some use cases and requirements relating to RTCWEB NAT/FW traversal can be found in [draft-ietf-rtcweb-use-cases-and-requirements].

If an organization wants to support RTCWEB such a TURN server may be located in the DMZ of the private network of that organization where it is still under administrative control.

In certain environments with very restrictive FW policies a TURN server in the public internet may not be sufficient to establish connectivity towards the RTCWEB peer for RTP-based media [RFC3550]. Such policies can include blocking of all UDP based traffic and allowing only HTTP(S) traffic to the TCP ports 80/443. In addition access to the World Wide Web from inside an organization is often only possible via a HTTP proxy.

This document examines impact of NAT/FW policies in Section 2. Additional impacts due to the presence of a HTTP proxy are examined in Section 3.

1.1. Requirements Language

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].

2. Considerations for NATs/Firewalls independent of HTTP proxies

This section covers aspects of how NAT/FW characteristic influence the establishment of a media stream. Additional aspects introduced by the presence of a HTTP proxy are covered in Section 3.

If the NATs serving caller and callee both show port and address dependent filtering behavior the need for a TURN server arises in order to establish connectivity for media streams. The TURN server will relay the RTP packet to the RTCWEB peer using UDP. How the RTP packets can be transported from the RTCWEB client within the private network to the TURN server depends on what the firewall will let pass through.

Other types of NATs do not require using the TURN relay. Nevertheless, the FW rules and policies still affect how media streams can be established.

2.1. NAT/Firewall open for outgoing UDP and TCP traffic

This scenario assumes that the NAT/FW is transparent for all outgoing traffic independent of using UDP or TCP as transport protocol. This case is used as starting point for introduction of more restrictive firewall policies. It presents the least critical example with respect to the establishment of the media streams.

The TURN server can be reached directly from within the private network via the NAT/FW and the ICE procedures will reveal that media can be sent via the TURN server. The TURN client will send its media to the allocated resources at the TURN server via UDP.

Dependent on the port range that is used for RTCWEB media streams, the same statement would be true if the NAT/Firewall would allow UDP traffic for a restricted UDP port range only.

2.2. NAT/Firewall open only for TCP traffic

This scenario assumes that the NAT/FW is transparent for outgoing traffic only using TCP as transport protocol. This gives two options for media stream establishment dependent on the NAT's filtering characteristics. Either transport RTP over TCP or contacting the TURN server via TCP.

In the first case the browser needs to use ICE-TCP [RFC6544] and provide active, passive and/or simultaneous-open TCP candidates. Assuming the peer also provides TCP candidates, a connectivity check for a TCP connection between the two peers should be successful.

In the second case the browser needs to contact the TURN server via TCP for allocation of an UDP-based relay address at the TURN server. The ICE procedures will reveal that RTP media can be sent via the TURN relay using the TCP connection between TURN client and TURN server. The TURN server would then relay the RTP packets using UDP, as well as other UDP-based protocols. ICE-TCP is not needed in this context.

Note that the second case is not to be mixed up with TURN/TCP [RFC6062], which deals with how to establish a TCP connection to the peer. For this document we assume that the TURN server can reach the peer always via UDP, possibly via a second TURN server.

2.3. NAT/Firewall open only for TCP-based HTTP(s) traffic

In this case the firewall blocks all outgoing traffic except for TCP traffic to port 80 for HTTP or 443 for HTTPS. A TURN server listening to its default ports (3478 for TCP/UDP, 5349 for TLS) would not be reachable in this case.

However, the TURN server can still be reached when it is configured to listen to the HTTP(S) ports as well. In addition the RTCWEB clients need to be configured to contact the TURN server over the HTTP(S) ports and/or needs to be able to tell the browser accordingly.

3. Considerations for NATs/Firewalls in presence of HTTP proxies

This section considers a scenario where all HTTP(S) traffic is routed via an HTTP proxy. Note: If both RTCWEB clients are located behind the same HTTP proxies, we, of course, assume that ICE would give us a direct media connection within the private network. We consider this case as out of the scope of this document.

3.1. HTTP proxy with NAT/Firewall open for outgoing UDP and TCP traffic

As in Section 2.1 we assume that the NAT/FW is transparent for all outgoing traffic independent of using UDP or TCP as transport protocol. The HTTP proxy has no impact on the transport of media streams in this case. Consequently, the same considerations as in Section 2.1 apply with respect to the traversal of the NAT/FW.

3.2. HTTP proxy with NAT/Firewall open only for TCP traffic

As in Section 2.2 we assume that the NAT/FW is transparent only for outgoing TCP traffic. The HTTP proxy has no impact on the transport of media streams in this case. Consequently, the same considerations as in Section 2.2 apply with respect to the traversal of the NAT/FW.

3.3. HTTP proxy assisted TURN server connection

3.3.1. TURN server connection via TCP

Different from the previous scenarios, we assume that the NAT/FW accepts outgoing traffic only via a TCP connection that is initiated from the HTTP proxy. Consequently, a RTCWEB client would have to use the HTTP CONNECT method [RFC2616] in order to get access to the TURN server via the HTTP proxy. The HTTP CONNECT request needs to convey the TURN Server URI or transport address. As a result the HTTP Proxy will establish a TCP connection to the TURN server, i.e. the TURN server only has to handle a standard TCP connection and an update to the TURN protocol or the TURN software is not needed.

Afterwards, the RTCWEB client could upgrade the connection to use TLS, forward STUN/TURN traffic via the HTTP proxy and use the TURN server as media relay. Note that upgrading in this case is not to be misunderstood as usage of the HTTP UPGRADE method as specified in [RFC2817] as this would require the TURN server to support HTTP. We rather envisage the following sequence:

- the browser opens a TCP connection to the HTTP proxy,
- the browser issues a HTTP CONNECT request to the HTTP proxy with the TURN server address in the Request URI,

- the HTTP proxy opens a TCP connection to the TURN server and "bridges" the incoming and outgoing TCP connections together, forming a virtual end-to-end TCP connection,

- the browser can do a TLS handshake over the virtual end-to-end TCP connection with the TURN server.

If it is not possible to use HTTP CONNECT in this way it will not be possible to establish connectivity between the RTCWEB peers and the ICE connectivity checks will fail.

Strictly speaking the TLS upgrade is not necessary, but using TLS would also prevent the HTTP proxy from sniffing into the data stream and provides the same flow as HTTPS and might improve interoperability with proxy servers. Some tests (done a while ago) indicated that there are proxies performing Deep Packet inspection (DPI) that expect to see at least a SSL handshake and, possibly, valid SSL records. The application has the ability to control whether SSL is used by the parameters it supplies to the TURN URI (e.g. turns: vs. turn:), so the decision to do TURN/TCP to port 443 versus TURN/TLS to port 443 could be left up to the application or possibly the browser configuration script.

In contrast to using UDP or TCP for transporting the STUN messages, the browser would now need to first establish a HTTP over TCP connection to the HTTP proxy, upgrade to using TLS and then switch to using this TLS connection for transport of STUN messages. It is also desirable that the browser detects the need to connect to the TURN server through a HTTP proxy automatically in order to achieve seamless deployment and interoperability. The browser should use the same proxy selection procedure for TURN as currently done for HTTP. The user or network administrator should not be required to change browser or proxy script configuration.

Further considerations apply to the default connection timeout of the HTTP proxy connection to the TURN server and the timeout of the TURN server allocation. Whereas [RFC5766] specifies a 10 minutes default lifetime of the TURN allocation, typical proxy connection lifetimes are in the range of 60 seconds if no activity is detected. Thus, if the RTCWEB client wants to pre-allocate TURN ressources it needs to refresh TURN allocations more frequently in order to keep the TCP connection to its TURN server alive.

3.3.2. TURN server connection via UDP

If a local TURN server under administrative control of the organization is deployed it is desirable to reach this TURN server via UDP. The TURN server could be specified in the proxy configuration script, giving the browser the possibility to learn how to access it. Then, when gathering candidates, this TURN server would always be used such that the RTCWEB client application could get UDP traffic out to the internet.

4. Other Approaches

4.1. TURN server connection via WebSocket

The RTCWEB client could connect to a TURN server via WebSocket [RFC6455] as described in [draft-chenxin-behave-turn-WebSocket]. This might have benefits in very restrictive environments where HTTPS is not permitted through the proxy. However, such environments are also likely to deploy DPI boxes which would eventually complain against usage of WebSocket or block RTCWEB traffic based on other heuristic means. It is also to be expected that an environment that does not allow HTTPS will also forbid usage of WebSocket over TLS.

In addition, usage of TURN over WebSocket puts an additional burden on existing TURN server implementation to support HTTP and WebSocket. The resulting benefit seems rather small, thus TURN over WebSocket is left for further study.

4.2. Port Control Protocol

As a further alternative, the Port Control Protocol (PCP) [RFC6887] allows to configure how incoming IPv6 or IPv4 packets are translated and forwarded by a NAT/FW. However, this document does not examine benefits of PCP for the management of the local NAT/FW, but leaves this for further study until PCP is deployed more widely.

4.3. HTTP Fallback for RTP Media Streams

As an alternative to using a TURN server it was proposed to send RTP directly over HTTP [draft-miniero-rtcweb-http-fallback]. This approach bears some similarities with TURN as it also uses a RTP relay. However, it uses HTTP GET and POST requests to receive and send RTP packets.

Despite a number of open issues, the proposal addreses some corner cases. However, the expected benefit in form of an increased success rate for establishment of a media stream seems rather small, thus HTTP fallback is left for further study.

5. Requirements for RTCWEB-enabled browsers

For the purpose of relaying RTCWEB media streams or data channels a browser needs to be able to

- connect to a TURN server via UDP, TCP and TLS,
- connect to a TURN server via a HTTP proxy using the HTTP connect method,
- connect to a TURN server via the HTTP(s) ports 80/443 instead of the default STUN ports 3478/5349,
- upgrade the HTTP proxy-relayed connection to the TURN server to use TLS,
- use the same proxy selection procedure for TURN as currently done for HTTP,
- switch the usage of the HTTP proxy-relayed connection with the TURN server from HTTP to STUN/TURN,
- use a preconfigured or standardized port range for UDP-based media streams or data channels,
- learn from the proxy configuration script about the presence of a local TURN server and use it for sending UDP traffic to the internet,
- support ICE-TCP for TCP-based direct media connection to the RTCWEB peer.

6. Acknowledgements

The authors want to thank Heinrich Haager for all his input during many valuable discussions.

Furthermore, the authors want to thank for comments and suggestions received from ...

7. IANA Considerations

This memo includes no request to IANA.

8. Security Considerations

TBD

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

9.2. Informative References

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC2817] Khare, R. and S. Lawrence, "Upgrading to TLS Within HTTP/1.1", RFC 2817, May 2000.

[RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[RFC4787] Audet, F. and C. Jennings, "Network Address Translation (NAT) Behavioral Requirements for Unicast UDP", BCP 127, RFC 4787, January 2007.

[RFC5245] Rosenberg, J., "Interactive Connectivity Establishment (ICE): A Protocol for Network Address Translator (NAT) Traversal for Offer/Answer Protocols", RFC 5245, April 2010.

[RFC5766] Mahy, R., Matthews, P., and J. Rosenberg, "Traversal Using Relays around NAT (TURN): Relay Extensions to Session Traversal Utilities for NAT (STUN)", RFC 5766, April 2010.

[RFC6062] Perreault, S. and J. Rosenberg, "Traversal Using Relays around NAT (TURN) Extensions for TCP Allocations", RFC 6062, November 2010.

[RFC6455] Fette, I. and A. Melnikov, "The WebSocket Protocol", RFC 6455, December 2011.

[RFC6544] Rosenberg, J., Keranen, A., Lowekamp, B., and A. Roach, "TCP Candidates with Interactive Connectivity Establishment (ICE)", RFC 6544, March 2012.

[RFC6887] Wing, D., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", RFC 6887, April 2013.

[draft-chenxin-behave-turn-WebSocket]
Xin. Chen , "Traversal Using Relays around NAT (TURN) Extensions for WebSocket Allocations ", 2013, <http://tools.ietf.org/html/draft-chenxin-behave-turn-WebSocket>.

[draft-ietf-rtcweb-use-cases-and-requirements]
C. Holmberg, S. Hakansson, G. Eriksson , "Web Real-Time Communication Use-cases and Requirements ", 2012, <http://tools.ietf.org/html/draft-ietf-rtcweb-use-cases-and-requirements>.

[draft-miniero-rtcweb-http-fallback]
L. Miniero , "HTTP Fallback for RTP Media Streams ", 2012, <http://tools.ietf.org/html/draft-miniero-rtcweb-http-fallback>.

Authors' Addresses

Thomas Stach (editor)
Siemens Enterprise Communications
Dietrichgasse 27-29
Vienna 1030
AT

Email: thomas.stach@siemens-enterprise.com

Andrew Hutton
Siemens Enterprise Communications
Technology Drive
Nottingham NG9 1LA
UK

Email: andrew.hutton@siemens-enterprise.com

Justin Uberti
Google
747 6th Ave S
Kirkland, WA 98033
US

Email: justin@uberti.name