Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 21, 2016

G. Shieh
J. Uberti
Google

March 20, 2016

WebRTC IP Address Handling Recommendations
draft-ietf-rtcweb-ip-handling-00

Abstract

This document provides best practices for how IP addresses should be handled by WebRTC applications.

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 September 21, 2016.

Copyright Notice

Table of Contents

1. Introduction 2
2. Problem Statement 2
3. Goals 3
4. Detailed Design 4
5. Application Guidance 5
6. Security Considerations 6
7. IANA Considerations 6
8. Acknowledgements 6
9. Informative References 6
Appendix A. Change log 7
Authors' Addresses 7

1. Introduction

As a technology that supports peer-to-peer connections, WebRTC may send data over different network paths than the path used for HTTP traffic. This may allow a web application to learn additional information about the user, which may be problematic in certain cases. This document summarizes the concerns, and makes recommendations on how best to handle the tradeoff between privacy and media performance.

2. Problem Statement

WebRTC enables real-time peer-to-peer communications by enumerating network interfaces and discovering the best route through the ICE protocol. During the ICE process, the peers involved in a session gather and exchange all the IP addresses they can discover, so that the connectivity of each IP pair can be checked, and the best path chosen. The addresses that are gathered usually consist of an endpoint's private physical/virtual addresses, and its public Internet addresses.

These addresses are exposed upwards to the web application, so that they can be communicated to the remote endpoint. This allows the application to learn more about the local network configuration than it would from a typical HTTP scenario, in which the web server would only see a single public Internet address, i.e. the address from which the HTTP request was sent.

The information revealed falls into three categories:

(1) If the client is behind a NAT, the client's private IP addresses, typically [RFC1918] addresses, can be learned.

(2) If the client tries to hide its physical location through a VPN, and the VPN and local OS supports routing over multiple interfaces, WebRTC will discover the public address associated with both the VPN as well as the ISP public address over that the VPN runs over.

(3) If the client is behind a proxy, but direct access to the Internet is also supported, WebRTC's STUN checks will bypass the proxy and reveal the public address of the client.

Of these three concerns, #2 is the most significant concern, since for some users, the purpose of using a VPN is for anonymity. However, different VPN users will have different needs, and some VPN users (e.g. corporate VPN users) may in fact prefer WebRTC to send media traffic directly, i.e. not through the VPN.

#3 is a less common concern, as proxy administrators can control this behavior through local firewall policy if desired, coupled with the fact that forcing WebRTC traffic through a proxy will have negative effects on both the proxy and on media quality. For situations where this is an important consideration, use of a RETURN proxy, as described below, can be an effective solution.

#1 is considered to be the least significant concern, given that the local address values often contain minimal information (e.g. 192.168.0.2), or have built-in privacy protection (e.g. [RFC4941] IPv6 addresses).

Note also that these concerns predate WebRTC; Adobe Flash Player has provided similar functionality since the introduction of RTMFP in 2008.

3. Goals

Being peer-to-peer, WebRTC represents a privacy-enabling technology, and therefore we want to avoid solutions that disable WebRTC or make

it harder to use. This means that WebRTC should be configured by default to only reveal the minimum amount of information needed to establish a performant WebRTC session, while providing options to reveal additional information upon user consent, or further limit this information if the user has specifically requested this. Specifically, WebRTC should:

- Provide a privacy-friendly default behavior which strikes the right balance between privacy and media performance for most users and use cases.

- For users who care more about one versus the other, provide a means to customize the experience.

4. Detailed Design

The main ideas for the design are the following:

- By default, WebRTC should follow the route for HTTP traffic, when this is easy to determine (i.e. not considering proxies). This is accomplished by binding local sockets to the wildcard addresses (0.0.0.0 for IPv4, :: for IPv6), which allows the OS to route WebRTC traffic the same way as normal HTTP traffic, and allows only the 'typical' public addresses to be discovered.

- By default, support for host-host connections should be maintained. Even when binding to the wildcard addresses, the local IPv4 and IPv6 addresses of the interface used for outgoing STUN traffic should still be surfaced as candidates; this is necessary for certain peer-to-peer data channel apps to function correctly. The appropriate addresses here can be discovered by binding sockets to the wildcard addresses, connect()ing those sockets to a public destination (e.g. "8.8.8.8"), and then reading the bound local addresses via getsockname().

- WebRTC incorporates an explicit permission grant for access to local audio and video, which are typically much more sensitive than the aforementioned IP address information. If the user has consented to media access, this should also allow WebRTC to gather all possible candidates and determine the absolute best route for

media traffic.

- Determining whether a web proxy is in use is a complex process, as the answer can depend on the exact site or address being contacted. Furthermore, web proxies that support UDP are not widely deployed today. Therefore, the only way to ensure that WebRTC traffic traverses a proxy is to force WebRTC to use ICE-TCP or TURN-over-TCP, and always try to make the TCP connection through the proxy, if one exists. Naturally, this will have attendant costs on media quality and also proxy performance.

- RETURN [I-D.ietf-rtcweb-return] is a new proposal for explicit proxying of WebRTC media traffic. When RETURN proxies are deployed, media and STUN checks will go through the proxy, but without the performance issues associated with sending through a web proxy.

Based on these ideas, we define four modes of WebRTC behavior, reflecting different privacy/media tradeoffs:

Mode 1 Enumerate all addresses: WebRTC will bind to all interfaces individually and use them all to ping STUN servers or peers. This will converge on the best media path, and is ideal when media performance is the highest priority, but it discloses the most information. As such, this should only be performed when the user has explicitly given consent for local media access, as indicated in design idea #3 above.

Mode 2 Default route + the single associated local address: By binding solely to the wildcard address, media packets will flow through the same route as normal HTTP traffic. In addition, the associated private address is discovered through getsockname, as mentioned above. This ensures that direct connections can still be established even when local media access is not granted, e.g. for data channel applications.

Mode 3 Default route only: This is the the same as Mode 2, except that the associated private address is not provided, which may cause traffic to hairpin through NAT or fall back to the application TURN server, with resulting quality implications.

Mode 4 Force TCP and proxy: This disables any use of UDP and forces use of TCP to connect to the TURN server or peer. If a web proxy server is configured, the TCP traffic will be sent through the proxy, with resulting quality implications.

We recommend Mode 1 as the default behavior only if cam/mic permission has been granted, or Mode 2 if this is not the case.

Users who prefer Mode 3 or 4 should be able to select a preference or install an extension to force their browser to operate in the specified mode. For example, Chrome users can install the WebRTC Network Limiter extension for this configuration.

Note that when a RETURN proxy is configured for the interface associated with the default route, Mode 2 and 3 will cause any external media traffic to go through the RETURN proxy. This provides an effective solution to the proxy concern mentioned in the problem statement, but without the performance issues associated with Mode 4.

5. Application Guidance

The recommendations mentioned in this document may cause breakage to certain WebRTC applications. In order to be robust in all scenarios, applications should follow the following guidelines:

- Applications should deploy a TURN server with support for both UDP and TCP connections to the server. This ensures that connectivity can still be established, even when Mode 3 or 4 are in use.

- Applications can detect when they don't have access to the full set of ICE candidates by checking for the presence of host candidates. If no host candidates are present, Mode 3 or 4 above is in use.

- Future versions of browsers may present an indicator to signify that the page is using WebRTC to set up a peer-to-peer connection. Applications should be careful to only use WebRTC in a fashion that is consistent with user expectations.

6. Security Considerations

This document is entirely devoted to security considerations.

7. IANA Considerations

This document requires no actions from IANA.

8. Acknowledgements

Several people provided input into this document, including Harald Alvestrand, Ted Hardie, Matthew Kaufmann, and Eric Rescorla.

9. Informative References

[I-D.ietf-rtcweb-return]
Schwartz, B. and J. Uberti, "Recursively Encapsulated TURN (RETURN) for Connectivity and Privacy in WebRTC", draft-ietf-rtcweb-return-01 (work in progress), January 2016.

[RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <http://www.rfc-editor.org/info/rfc1918>.

[RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <http://www.rfc-editor.org/info/rfc4941>.

Appendix A. Change log

Changes in draft -00:

- Published as WG draft.

Authors' Addresses

Guo-wei Shieh
Google
747 6th St S
Kirkland, WA 98033
USA

Email: guoweis@google.com

Justin Uberti
Google
747 6th St S
Kirkland, WA 98033
USA

Email: justin@uberti.name