Table 1. Comparison of leaked IP addresses' types by the main web browsers for different WebRTC IP handling modes.

		Browser	**Mozilla Firefox**					**Chrome - Edge Opera - Brave**				**Safari**
		Configuration	Default	Forced				Default		Forced		Default
		User consent	Yes	No	Yes	No^{b}	Both	Yes	No	Both	Both	Both
		Mode	1	2.2	2	$2 \rightarrow 2.2$	3	1	2.2	2	3	2
Observation	**Retrieved address(es)**											
	Source	***Type***										
SDP offers	**Local**^{a}	mDNS	.	X	.	X	.	.	X	.	.	.
		Priv. IPv4	X^{c}	.	X^{e}	.	.	X^{c}	.	X^{e}	.	X^{e}
		Priv. IPv6	X^{d}	.	X^{d}	.	.	X^{d}	.	X^{d}	.	X^{f}
		Pub. IPv4	X	.	X^{e}	.	.	X	.	X^{e}	.	X^{e}
		Pub. IPv6	X	.	X	.	.	X	.	X	.	X
Wireshark	**STUN**	Pub. IPv4	X	X	X	X	X	X	X	X	X	X
		Pub. IPv6	X	X	X	X	X	X	X	X	X	X
	TURN	Pub. IPv4	X	X	X	X	X	X	X	X	X	X
		Pub. IPv6	X	X	X	X	X	X	X	X	X	X

Table 2. Comparison of leaked addresses' types by a **vanilla** Firefox for different WebRTC IP handling modes and configurations. *Docker only supports IPv6 on Linux.*

		Browser	**MF only MF + SOCKS/HTTP(S)**					**MF + VPNs**					**MF dockerised MF dockerised + SOCKS/HTTP(S)**					**MF dockerised + VPNs**				
		Configuration	Default		Forced			Default		Forced			Default		Forced			Default		Forced		
		User consent	Yes	No	Yes	No^{b}	Both	Yes	No	Yes	No^{b}	Both	Yes	No	Yes	No^{b}	Both	Yes	No	Yes	No^{b}	Both
		Mode	1	2.2	2	2.2	3	1	2.2	2	2.2	3	1	2.2	2	2.2	3	1	2.2	2	2.2	3
Obs.	**Retrieved addr.**																					
	Source	***Type***																				
SDP offers	**Local**^{a}	mDNS	.	X	.	X	.	.	X	.	X	.	.	X	.	X	.	.	X	.	X	.
		Priv. IPv4	X^{c}	.	X^{e}	.	.	X^{c}	.	VPN local	.	.	Doc. priv.	.	Doc. priv.	.	.	Doc. priv.	.	Doc. priv.	.	.
		Priv. IPv6	X^{d}	.	X^{d}	.	.	X^{d}	.	.^{h}	.	.	Doc. ULA	.	Doc. ULA	.	.	Doc. ULA	.	Doc. ULA	.	.
		Pub. IPv4	X	.	X^{e}	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.
		Pub. IPv6	X	.	X	.	.	X	.	.	.	.	.	.	.	.	.	.	.	.	.	.
Wire shark	**STUN**	Pub. IPv4	X	X	X	X	X	VPN	VPN	VPN	VPN	VPN	X	X	X	X	X	VPN	VPN	VPN	VPN	VPN
		Pub. IPv6	X	X	X	X	X	.^{g}	.^{h}	.^{h}	.^{h}	.^{h}	X	X	X	X	X	VPN	VPN	VPN	VPN	VPN
	TURN	Pub. IPv4	X	X	X	X	X	VPN	VPN	VPN	VPN	VPN	X	X	X	X	X	VPN	VPN	VPN	VPN	VPN
		Pub. IPv6	X	X	X	X	X	.^{g}	.^{h}	.^{h}	.^{h}	.^{h}	X	X	X	X	X	VPN	VPN	VPN	VPN	VPN

Software information. The various software and versions used in our experiments are available in the Anonymous GitHub repository.

Tables notations. The notation in Tables 1, 2, and 3 are as follows. "X" means that one (or more) client's address(es) were retrieved (depending on the WebRTC IP handling mode). "." means that no address was retrieved. The means used to observe the host addresses retrieved and the addresses discovered by the STUN/TURN servers are in the "Obs." column for Observation. No crosses in 1, 2, and 3 correspond to TURN *relayed addresses* as they do not identify the client. *VPN local, Doc. priv., Doc. ULA, VPN* mean *VPN private IP address, Docker private IPv4 address, Docker ULA IPv6 address, VPN's public IP address*, respectively. *MF* means Mozilla Firefox, *lo* means loopback IP address(es), and *ll* means private link-local address(es).

Table 3. Comparison of leaked addresses' types by a **compromised** MF for different WebRTC IP handling modes and configurations. *Docker only supports IPv6 on Linux.*

		Browser	**compr. MF compr. MF + SOCKS/HTTP(S)**		**compr. MF + VPNs**		**compr. MF dockerised & compr. MF dockerised + SOCKS-HTTP(S)**		**compromised MF dockerised + VPNs**	
		Configuration	Forced		Forced		Forced		Forced	
		User consent	Yes	No	Yes	No	Yes	No	Yes	No
		Mode	1	2	1	2	1	2	1	2
Obs.	**Retrieved addr.**									
	Source	***Type***								
SDP Offers	**Local**[a]	mDNS	.	.	.	.	.	.	.	.
		Priv. IPv4	X[c]	X[e]	X[c]	VPN local	Doc. priv. + lo	Doc. priv.	Doc. priv. + lo	Doc. priv.
		Priv. IPv6	X[d]	X[d]	X[d]	.[g]	Doc. ULA + lo + ll	Doc. ULA	Doc. ULA + lo + ll	Doc. ULA
		Pub. IPv4	X	X[e]	X	.	.	.	.	.
		Pub. IPv6	X	X	X	.	.	.	.	.
Wire Shark	**STUN**	Pub. IPv4	X	X	VPN	VPN	X	X	VPN	VPN
		Pub. IPv6	X	X	.[g]	.[h]	X	X	VPN	VPN
	TURN	Pub. IPv4	X	X	VPN	VPN	X	X	VPN	VPN
		Pub. IPv6	X	X	.[g]	.[h]	X	X	VPN	VPN

Tables footnotes. The footnotes referenced in Tables 1, 2, and 3 are as follows:

[a] For IP addresses, it is subject to have these addresses' types associated with the local interface(s) used according to the different modes (RFC 8828 [3]).

[b] Even when forcing the use mode 2 on Firefox, the default mode is 2.1 without user consent (mDNS protection of the preferred interface's local addresses).

[c] Firefox filters local IPv4 addresses associated with all interfaces and skips link-local and loop-back addresses [1]. Chromium-based browsers: [2].

[d] A filtering of IPv6 addresses associated with the interface(s) chosen is done. One or more IPv6 addr. preferred by this filtering will be chosen [1,2].

[e] It is either a public or a private IPv4, as mode 2 selects the default route interface and as an interface can only get one IPv4.

[f] Our observation shows that if the default interface does not have a public IPv6 addr. but only a ULA IPv6 addr. (routable to the internet via a NAT), it will be included in an ICE candidate (tested with OpenVPN UDP assigning a local ULA addr.). It is very like having a filter similar to the one in Firefox/Chromium.

[g] See [d]; In our tests, Wi-Fi and Ethernet interfaces attach temporary routable public IPv6 addresses. Thus, the VPN local ULA address is eliminated. There are indeed attempts to connect to the TURN and STUN servers via the public addresses (seen on Wireshark) but blocked by the VPN.

[h] See [d]; It seems that filtering is done on all interfaces before selecting the default interface used in modes 2, 2.1 and 3 (here VPN one). Thus as there is at least one public IPv6 address, the VPN ULA is removed. Firefox then appears to select only remaining IPv4 addresses associated with the VPN interface (since the ULA has been removed). Thus, there can be no IPv6 ULA (private) address, routable to the Internet via the VPN NAT, in the ICE candidate list, and therefore no requests to the STUN/TURN servers.

References

1. Mozilla: addrs.c (Apr 2024), https://hg.mozilla.org/releases/mozilla-release/file/FIREFOX_125_0_3_RELEASE/dom/media/webrtc/transport/third_party/nICEr/src/stun/addrs.c#l66, accessed on 2024-07-12.
2. The Chromium Project: network.cc (May 2024), https://webrtc.googlesource.com/src.git/+/refs/branch-heads/6478/rtc_base/network.cc#620, accessed on 2024-07-12.
3. Uberti, J.: WebRTC IP Address Handling Requirements. Request for Comments RFC 8828, Internet Engineering Task Force (Jan 2021). https://doi.org/10.17487/RFC8828