

Smoothing Rough Edges of IPv6 in VPNs

Yejin Cho

University of Southern California
Los Angeles, USA
yejincho@usc.edu

John Heidemann

Information Sciences Institute
University of Southern California
Los Angeles, USA
johnh@isi.edu

Abstract

How do commercial VPNs interact with IPv6? We show two “rough edges” in how commercial VPNs handle IPv6. First, we show that many *IPv4-only VPNs leak IPv6 traffic to the ISP*. Individual use VPNs in part to conceal their local IP addresses, so such leaks reduce user privacy. While prior work has studied VPNs in testbeds, we use a new dataset of 129k VPN-using daily visitors to WhatIsMyIPAddress.com that quantifies these leaks and show 12 VPNs previously considered safe still leak for at least 5% of their users. We show native IPv6 addresses leak most commonly in VPNs that claim only IPv4 support, with 5% to 57% of visitors of v4-only VPNs having their native IPv6 address exposed. Second, we show that *most dual-stack VPNs users actually select IPv4 instead of IPv6*. We observe this problem in our visitor data, and we identify the root cause arises because when user’s computer follows standard address-selection rules, VPN-assigned addresses are often de-preferenced. Testing six VPNs on Android, we show that five consistently de-prioritize IPv6. Finally, *we suggest a solution to IPv6 de-preferencing*: we define a new IPv6 address range for VPNs that is not de-preferenced by address selection. We prototype this solution on Linux. Our findings help identify and address rough edges in the addition of IPv6 support to VPNs.

1 Introduction

Virtual private networks (VPNs) and IPv6 are important parts of today’s Internet. VPNs improve privacy and protect against censorship by encrypting data and hiding end-user IP addresses from eavesdropping. IPv6 offers enough addresses to enable true peer-to-peer, end-to-end connectivity. VPN providers increasing advertise native IPv6 support as a key feature [3, 6, 7].

Our paper explores **how commercial VPNs actually interact with IPv6**. We find two “rough edges” in practice: *IPv4-only VPNs often leak IPv6 traffic*, failing to protect user privacy, *VPNs advertise IPv6 support as feature but actually connects over IPv4*, even though IPv6 is available and working. We explore these rough edges with a novel data source, WhatIsMyIPAddress.com.

Our first contribution is to quantify **how often VPNs leak IPv6 traffic**, a rough edge where VPNs fail to provide

the privacy they promise users. VPNs are designed to hide a user’s local identity and IP addresses, rewriting them with VPN addresses when traffic leaves the VPN to its destination. If an IPv6 address from the user’s computer leaks on to the Internet, that provides a unique identifier that may allow a connection to the users real-world identity.

We use a new data source, WhatIsMyIPAddress.com, to look at what IPv4 and IPv6 addresses users *actually* expose (§4). We show that, in practice between 5 and 57% of all IPv4-only VPN users expose a local IPv6 address. While prior work has examined VPNs for IPv6 leaks based on testbed experiments [9–12], our use of real-world data allows us to see leaks that occur potentially due to user misconfiguration or OS bugs.

Our second contribution is to show that **most dual-stack VPNs users actually select IPv4 instead of IPv6**, even when both are available from the user and the destination (§5). We observe this problem in our WhatIsMyIPAddress.com data. We show that the root cause is that the user’s computer follows the standard address-selection rules when examining local network interfaces, while the VPN uses a ULA IPv6 address. The standard address-preferencing rules [15] will always select the private IPv4 address over this IPv6 ULA address, for reasons we explain in §5.2.

Our final contribution is to show **solutions to these rough edges**. We suggest that all VPNs need to support IPv6 so they can protect IPv6 addresses appropriately (§4.4). We suggest a new IPv6 address class can solve IPv6 de-preferencing (§5.5). We demonstrate this change through an implementation in Linux.

Ethical Considerations and Data Availability: Our study was reviewed by our university’s IRB (number anonymized), which classified it as Not Human Subjects Research (NHSR). The client data we analyze consist of IP logs collected during ordinary website visits, in line with the site’s privacy policy. We discuss ethics in Appendix A. As part of our data use agreement, our WhatIsMyIPAddress.com data is not available, but we provide data from our testbed experiments [8].

2 Background and Related Work

2.1 Background: Dual-Stack Connection

Dual-stack operation, when both the source and destinations support both IPv4 and IPv6, is the primary transition strategy for IPv6 today.

With years of transition experience, connections today follow three steps to select a protocol with minimal user latency: (1) DNS resolution, (2) address selection, and (3) connection racing. These steps are implemented jointly by the application (typically a web browser) and the operating system (networking libraries and the kernel); we describe the behavior of popular browsers in §B.1.

Suppose, a dual-stacked user wants to visit dual-stack website. First, the browser issues DNS queries to obtain a set of destination addresses (both A and AAAA records). Second, for each destination address (A and AAAA), the host chooses an appropriate source address and interface and *prioritizes* the resulting (source, destination) pairs into a ranked list according to its address-selection policy [15]. We describe this policy in detail in §B.2. Finally, if there multiple protocol options remain, the application may use the Happy Eyeballs algorithm [14] to select a quickly responding protocol. Although browsers differ in their implementations of these steps, the results we report are due to the RFC specifications and repeat across all implementations.

2.2 VPN Privacy and IPv6 Leakage

Several prior studies have evaluated IPv6 traffic leakage from VPN providers in 2015 [11], 2016 [9], 2018 [10], and 2022 [12] showing VPN evolution over time. These studies show VPNs are improving and leaking less IPv6 traffic. The first studies showed nearly all VPNs leaked IPv6 (14 of 14 in 2015, and 56 of 67 in 2016), with far fewer in the last two studies (12 of 43 in 2018 and 5 of 80 in 2022). However, all prior work evaluated VPNs as leak or not-leak; we instead consider the percentage of sessions we see that leak and show that several VPNs prior studies identified as “safe” do leak sometimes.

Each of these prior studies uses a similar methodology: they install and test selected VPNs by monitoring device-level traffic for leaks. They differ by adding more VPNs, or more operating systems.

This methodology suffers from combinatorial explosion with five major OS platforms and dozens of VPNs. Our work instead uses data from a popular website to study actual VPN users, providing insight into many platforms (more than can be tested by hand in prior work). Like their work, we test specific software to confirm our results and test our proposed changes. We provide a complete list of comparisons of VPN providers in §C.1.

2.3 Evaluation of Happy Eyeballs and IPv6 Prefencing

Prior work has examined failures in Step 1 and Step 3 of §2.1. Some studies focus on *DNS behavior* (Step 1)—for example, resolver bias, delayed AAAA responses, or inconsistent dual-stack DNS handling [2]. Other studies focus on *Happy Eyeballs behavior* (Step 3), showing that faulty browser implementation can cause IPv6 de-preference even when IPv6 is otherwise functional [13].

In contrast, our work focuses on the *address-selection stage* (Step 2)—the stage that precedes Happy Eyeballs. We show that local addressing configurations in commercial VPNs can shift RFC 6724 outcomes and cause systematic IPv6 de-preference, even when: (i) DNS resolution returns valid AAAA records, and (ii) Happy Eyeballs would correctly prefer IPv6 if given a properly ordered candidate list.

3 Data Source

To get a sense of how VPNs interact with IPv6 in the real-world, we require a data source that samples millions of users of VPNs with both IPv4 and IPv6 address observed. We use data collected by WhatIsMyIPAddress.com, a website millions of people use to identify their IP addresses. This website responds to users evaluating their IPv4 and IPv6 addresses, if any. Although it reports computers and their IP addresses, it does not collect any information about the users of those computers, and we agree not to attempt to identify individuals in this data (consistent with our IRB).

When a computer accesses WhatIsMyIPAddress.com, it connects to the website using either IPv4 or IPv6 based on some algorithm, often a version of Happy Eyeballs. The website then identifies the other protocol by returning HTML with embedded resources that use only other IP protocol, allowing it to identify both addresses and log which protocol is preferred.

We use data from WhatIsMyIPAddress.com covering 30 days (5.2 million sessions, of which 4 million are unique after removal of repeats in each hour), from 2025-02-22 to 2025-03-24.

Supplemental data: We augment WhatIsMyIPAddress.com data with VPN detection and classification from IP-info [5], organizations from CAIDA’s AS2Org [4], and AS classifications from ASdb [17].

4 How Often Do VPNs Leak IPv6 Traffic?

VPNs are intended to protect user privacy and bypass censorship by encrypting traffic and tunneling through a VPN server. However, some VPNs fail to tunnel IPv6 traffic, causing to exit through the user’s native network interface and expose the real IPv6 address.

4.1 Methodology: Detecting Leaks

We define an *IPv6 leak* on a dual-stack device as when a user’s IPv4 traffic uses the VPN, but their IPv6 does not. Without the VPN for IPv6, the user does not get the privacy they expected: their non-VPN’ed IPv6 address is externally visible, and their traffic risks exposure because it is not encrypted by the VPN.

We observe IPv6 leaks from paired IP logs our WhatIsMyIPAddress.com data (§3): We detect a VPN when the IPv4 address is associated with a VPN service, and we identify a *VPN leak* when the IPv6 address (i) does not belong to a VPN service, and (ii) originates from a different organization, and (iii) is classified as ISP AS. We determine using (i) IPinfo VPN classification [5], (ii) from CAIDA’s AS2Org [4], and (iii) from ASdb [17].

4.2 Basic Traffic and IPv6 Leaks Across All Users

We first look at overall traffic and the rate of leaked IPv6 addresses. Our WhatIsMyIPAddress.com dataset spans 30 days and 4M users after preprocessing (§3).

We identify VPN users by IPv4 addresses that come from a known VPN address, as determined by IPinfo [5]. Figure 1a shows the mean number of sessions (log-scale) for each day, broken out by VPN. We detect 123 VPNs in our data, and report only the 35 VPNs that have more than mean of 100 sessions per day after cleaning. Here we sort VPNs by overall IPv6 leak rate, and label dual-stack (names are blue with stars), v4-only (red with triangles), and operator-policy-dependent (gray with an X). We manually determine IPv6 support based on the information provided on each VPN service’s webpage.

We then identify IPv6 leaks as described in §4.1. Figure 1b shows the distribution of connections by category. The top red bar represents the fraction of VPN connections that leak IPv6. The next, sky-blue bar corresponds to VPN connections that are safe because the IPv6 traffic is from Chrome’s prefetch server. Below that, the dark-blue bar shows dual-stack connections where both IPv4 and IPv6 are correctly tunneled through the VPN, and are therefore safe. Finally, the bottom light-green bar represents IPv4-only connections from VPN.

Leak rate across all users of VPNs: In Figure 1c, the dashed red line shows the fraction of IPv6 leaks that occur across all visitors to WhatIsMyIPAddress.com. We see that leaks range from about 0% to 30% based on VPN package. IPv4-only VPNs leak more frequently (mean 6.5%, median 2.8%) than dual-stack VPNs (mean 2.9%, median 1.8%) or unidentified services (mean 0.9%, median 0.4%)

Looking at all sessions shows that most IPv4-only VPN packages protect their users because they *do* usually disable IPv6, but sometimes they don’t.

4.3 IPv6 Leaks from Dual-Stack Visitors

We next focus on the subset of VPN users who are dual-stacked. We find that, although we would normally expect all dual-stacked connections from IPv4-only VPNs to be a leak, the actual leak rate varies.

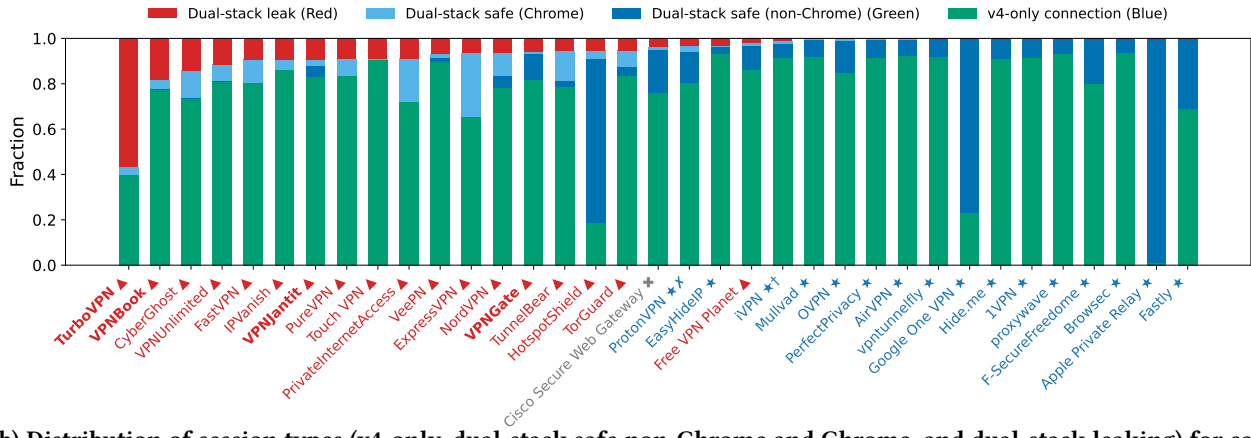
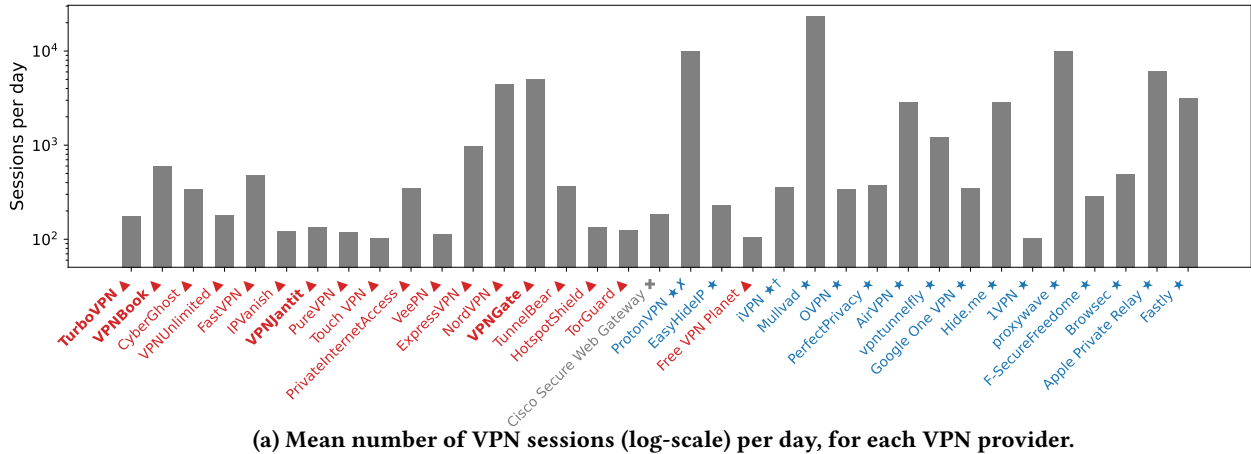
The top solid blue line in Figure 1c in gives the fraction of VPN users, who are dual-stacked, (those we observe both v4 and v6 addresses) who leaks native IPv6 addresses, grouped by VPN operator. We first consider the percentage of leaks relative to all dual stack users in the solid red line. We observe that VPNs with IPv6 support (blue stars) show lower leak rates than those that are IPv4-only (red triangles). We are surprised that some IPv4-only providers sometimes do protect IPv6 traffic, showing leak rates below 100% (ranging from about 95% down to 5%). We find that some self-identified IPv4-only providers still protect IPv6 traffic, with leak rates as low as 5%. Hotspot Shield is a striking example: although it identifies as an IPv4-only VPN and its customer support claims it “does not currently support IPv6,” our measurements show that roughly 95% of its dual-stack traffic is nevertheless protected. We verified that all ‘safe’ traffic originates from the same AS for both IPv4 and IPv6.

Two special cases are relevant to our classification: First, we sometimes observe partial IPv6 deployment in some VPNs clients. These cases do not satisfy our leak criteria (i to iii), because the IPv6 address does belong to the VPN’s AS, so they appear as unexpected protection, not a leak. Second, some traffic classified as safe, dual-stack sessions is due by Chrome prefetching. The Chrome browser speculatively loads resources in advance to improve page load performance. When it does so, the initial fetch of main page comes from Chrome’s AS, but subsequent requests follow through the user’s VPN. Thus the prefetched IPv6 connection appears outside the VPN and would be misclassified as a leak. To correct this case we treat traffic from prefixes assigned to Chrome Prefetch [1] as safe.

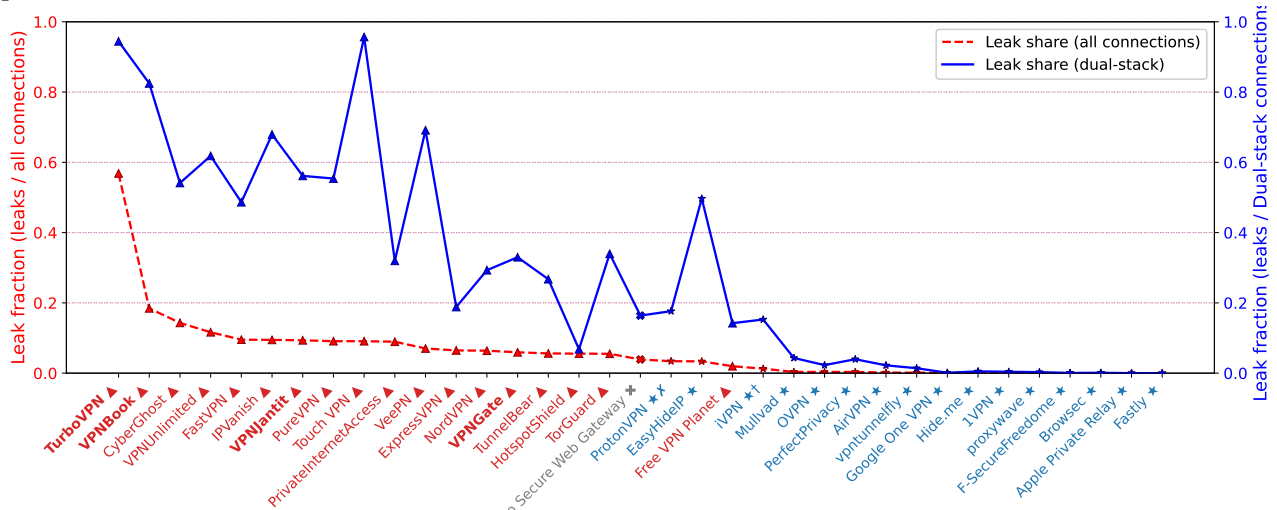
4.4 Testbed Validation and Recommendations

To confirm our WhatIsMyIPAddress.com findings, we tested few free VPNs in a controlled testbed. We installed TurboVPN, VPNBook, VPNJantit, VPNGate, and TunnelBear (listed in decreasing order of IPv6 leakage from our analysis) and found that all but TunnelBear leaked IPv6 traffic. This validates that the IPv6 leakage behaviors observed in our dataset also occur in real-world VPN clients. We list the experiment environment for each VPN in the §C.2.

Our recommendation to address IPv6 leaks is that VPNs should fully support IPv6, as is shown by the very few leaks we see in dual-stack VPNs (blue VPNs marked with a star in Figure 1c).



(b) Distribution of session types (v4-only, dual-stack safe non-Chrome and Chrome, and dual-stack leaking) for each VPN provider.



(c) Fraction of WhatIsMyIPAddress.com sessions with IPv6 leaks, shown relative to all sessions (the lower, red, dashed line) and dual-stack sessions (the upper, dark blue, solid line).

Figure 1: Evaluation of WhatIsMyIPAddress.com data and IPv6 leaks. VPN providers are shown on the x -axis, sorted by their leak rate over all sessions (the dashed red in Figure 1c). VPN type: v6-supported (VPN name blue with stars), v4-only (red with triangles), and operator-policy-dependent (gray with an X).

4.5 Limitations in Studying IPv6 Leaks

Although our data from WhatIsMyIPAddress.com captures a broad view of IPv6 traffic and potential leaks, it is not perfect.

First, WhatIsMyIPAddress.com is often used by users debugging networking problems. We do not claim that it represents the general Internet population, but only that it represents a broad population of millions of users.

Second, some leaks may be because users explicitly modified their network configuration. Thus even though the VPN disables IPv6, the user overrides this safe setting. Although these cases are real IPv6 leaks, user override should not be attributed as a limitation fault of the VPN software. Potentially such overrides may be more common in our data if WhatIsMyIPAddress.com attracts more technically savvy users.

Finally, our work assumes the supplement data we use from external sources is correct. Misclassification in VPN classification data could bias which providers are included in our study.

In fact, in a discussion with one VPN company (anonymized), we learned that the database we used covered only about 1% of their actual VPN exit IP addresses, which means our findings may not linearly reflect the number of VPN leaks in the real world. In addition, the ASdb database was last updated in May 2021, while our work was done in 2025, so it may not fully reflect current ASes.

5 Do VPNs De-prioritize IPv6?

When users have both IPv4 and v6, they have an option which protocol to use. Happy Eyeballs favors IPv6 (§2), but we show that in practice, 5 out of 6 VPNs favor IPv4. We show the cause of VPN de-prioritizing and suggest how this problem can be fixed.

5.1 Observing IPv6 De-Preferencing in the Wild

Happy Eyeballs (§2) is designed to favor IPv6 when given a choice of protocol. We evaluated the protocol choice in WhatIsMyIPAddress.com.

Surprisingly, we find that *54% of dual-stack VPN visitors prefer IPv4 over IPv6*, suggesting that Happy Eyeballs is not working for these users. Figure 2 shows the fraction of IPv6 depreferencing by VPN provider, broken out by VPN. We report fractions only for VPNs that see at least 20 dual-stack, safe sessions per day; 54% is the traffic-weighted mean of these fractions.

By contrast, only 13% of all dual-stack, non-VPNed users who prefer IPv4 because the Happy Eyeballs algorithm intentionally prefers IPv6 [14].

5.2 Protocol Preferences and a Root Cause

While on the surface Happy Eyeballs races IPv4 and IPv6, with a head-start for IPv6, HE only comes in to play when equivalent IPv4 and v6 addresses are available. The Root Cause of VPN depreferencing of IPv6 is that private addresses are not considered equivalent in IPv4 and IPv6.

IP address prioritization is evaluated based on RFC-6724 [15], which classifies and ranks all possible (source, destination) pairs. (We summarize these rules in §B.2.) When accessing the Internet without a VPN, clients usually use *private* IPv4 and public IPv6 addresses. RFC 6724 prioritizes public IPv6 over private IPv4, so IPv6 addresses appear earlier in the sorted address list. Happy Eyeballs then initiates connection attempts in that order, giving the first address a head start. As a result, IPv6 is usually selected, and IPv4 is used only if IPv6 fails or is significantly slower.

When running a VPN, the VPN often assigns *private* IPv4 and IPv6 addresses to VPN’s tunnel, using RFC-1918 space for IPv4 and a ULA address for IPv6. Here, with private IPv4 and ULA IPv6 addresses, RFC-6724 prioritization produces a different outcome. De-prioritization occurs because private IPv4 addresses are treated differently from ULA IPv6 addresses—ULA addresses are considered “link local” and their use is discouraged, so HE only tries these IPv6 address after a delay.

Thus different treatment of RFC-1918 IPv4 space and ULA IPv6 space with standard prioritization rules *systematically de-prioritize v6 use for VPN users*.

5.3 Testbed Validation of De-Preference

To confirm our understanding of VPN de-prioritizing of IPv6 we test 6 commercial VPNs that advertise support of native v6. We install each vendor’s VPN client on an Android device, activate the VPN, then inspect the assigned tunnel addresses using the Android Debug Bridge.

We find that one out of the six VPN services (Proton-VPN) uses a GUA address on the tunnel interface. Proton-VPN mostly used IPv6 for connection (28% de-preference). Although they use GUAs, there is still slightly more depreferencing than the 13% baseline for non-VPN users.

The other five providers we tested (Mullvad, AirVPN, hide-meVPN, Perfect Privacy, and Anonine) all have very high de-preference fractions: 78%, 79%, 69%, 75%, and 100%, respectively. We see that each of these VPNs assigns ULAs, and so they frequently de-prioritize IPv6. (However, because it is a race, sometimes IPv6 still wins.)

These results suggest our analysis of how HE and prioritization interact with IPv6 above is a plausible explanation for the IPv6 de-prioritization we observe in our data.

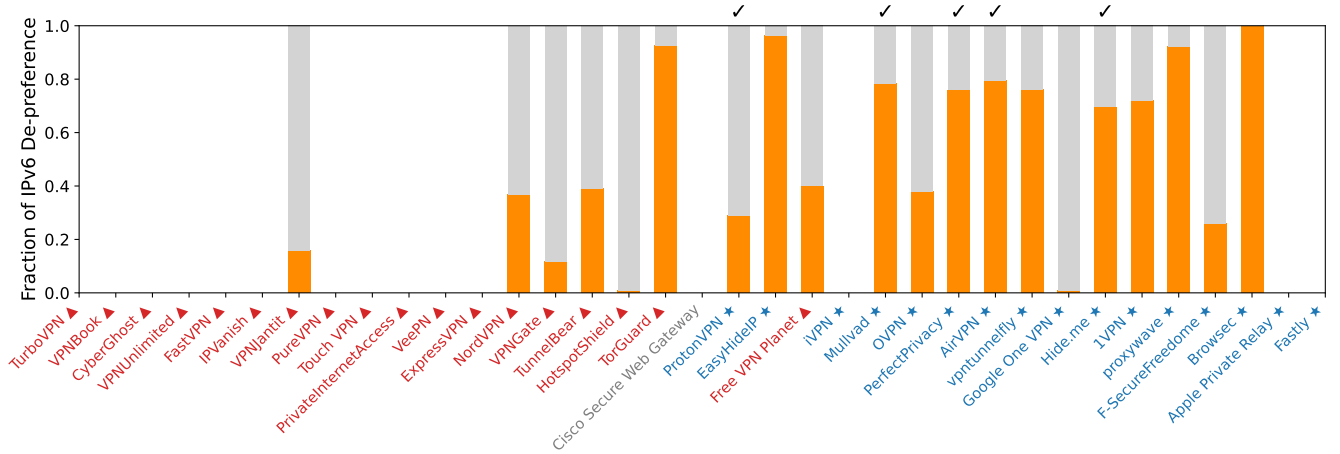


Figure 2: Fraction of dual-stack, safe, VPNs sessions to WhatIsMyIPAddress.com that de-prefere IPv6 for VPNs with at least 20 such sessions per day. VPNs marked with a check were tested in §5.3

5.4 Options to Avoiding De-Preferencing

We see three potential solutions to IPv6 de-preferencing: (1) VPNs can use GUAs for both sides of their tunnel, (2) address prioritization can treat IPv4 and IPv6 private addresses similarly, or (3) we can add a new address class for VPN IPv6 addresses.

While all of these choices are possible, for practical reasons we recommend choice (3) and describe it in §5.5.

During our discussion of option (2) with members of the IETF who are currently revising RFC 6724, they indicated that changing the prioritization of ULAs would be undesirable, as ULAs are defined as non-global addresses.

Although VPNs could use public IPv6 addresses (GUAs), distributing GUA addresses to VPN clients would complicate their implementation and require they treat IPv6 differently than IPv4 (where private RFC-1918 space is the only option), and VPNs like the non-routability of private addresses.

Due to these reasons, it seems unlikely that many VPNs will adopt solution (1).

5.5 Re-preferencing: a New Address Class

We propose to avoid IPv6 de-preferencing in VPNs by defining a new, VPN-specific address class, a *Tunnel Local Address* or TLA. These addresses are intended to be used only as VPN tunnel endpoints, but should be prioritized the same as IPv4 private addresses, thus avoiding de-prioritization while retaining strict separation (and non-routability) of VPN-internal addresses.

Currently ULA space is allocated fc00::/7, but it uses only fd00::/8. We suggest TLA could use currently unallocated

fc00::/8. (Alternatively, it could take part of ULA space, like fc00::/12).

We then suggest updates to RFC-6724 treat TLA space as label 1 (same class as GUA), precedence 35 (below GUA, above IPv4).

Our prototype: We have prototyped TLA in Linux-6.15.8 using prefix fc00::/8. We then modified Linux address selection rules (in /etc/gai.conf) to follow our proposed changes to RFC-6724.

With this configuration, standard browsers will consistently select IPv6 when connecting to v6-enabled Internet destinations, confirming this proposed solution allows VPNs to prioritize IPv6 while maintaining address separation.

6 Conclusion

In this work, we examined two problems of commercial VPN services' IPv6 support —IPv6 traffic leak and IPv6 de-preferencing. Using logs from 129k VPN-using daily visitors to WhatIsMyIPAddress.com and targeted client experiments, we first demonstrated that many IPv4-only VPNs leak tunnel IPv6 traffic to the ISP, exposing default IPv6 address users intend to hide. Second, for VPNs that advertise IPv6 support, we showed that dual-stack connections overwhelmingly prefer IPv4: 54% of Dual-stack VPN sessions use IPv4, around 4 times higher than non-VPN dual-stack users, which shows 13%, and controlled tests on Android reveal that five of six providers assign ULA prefixes to the tunnel, which, under RFC6724, are de-prioritized relative to private IPv4, inducing systematic Happy-Eyeballs failure. To address this structural bias, we proposed the Translatable Local Address (TLA) space in fc00::/8, and implemented a Linux-based prototype that integrates TLA via gai.conf and standard IPv6 translation,

restoring IPv6 preference without kernel changes. Our results indicate that VPNs requires improved handling of IPv6.

References

- [1] 2025. Chrome Prefetch Proxy IP list. https://www.gstatic.com/chrome/prefetchproxy/prefetch_proxy_geofeed
- [2] 2023. *IPv6, the DNS and Happy Eyeballs* | blabs. <https://labs.apnic.net/index.php/2023/11/16/ipv6-the-dns-and-happy-eyeballs/>
- [3] 2014. *Mullvad IPv6 support*. <https://mullvad.net/en/blog/ipv6-support>
- [4] visited on 2025-08-01. AS to Organization Mapping Dataset. <https://www.caida.org/catalog/datasets/as-organizations/>
- [5] visited on 2025-08-18. IPInfo Proxy&VPN Detection API. <https://ipinfo.io/products/proxy-vpn-detection-api>
- [6] visited on 2025-11-19. *HideMe IPv6 support*. <https://hide.me/en/features/ipv6-vpn>
- [7] visited on 2025-11-19. ProtonVPN IPv6 support. <https://protonvpn.com/support/prevent-ipv6-vpn-leaks?srsId=AfmBOoqFycFvutorGYCbVPb1tKXXJWOnpL7oEGgrLVkt0mlw0Tic2zC>
- [8] Anonymized for Review. 2025. IPv6 Rough Edges Datasets. <https://example.com/anonymized>.
- [9] Muhammad Ikram, Narseo Vallina-Rodriguez, Suranga Seneviratne, Mohamed Ali Kaafar, and Vern Paxson. 2016. An Analysis of the Privacy and Security Risks of Android VPN Permission-enabled Apps. In *Proceedings of the 2016 Internet Measurement Conference* (Santa Monica California USA, 2016-11-14). ACM, 349–364. doi:10.1145/2987443.2987471
- [10] Mohammad Taha Khan, Joe DeBlasio, Geoffrey M. Voelker, Alex C. Snoeren, Chris Kanich, and Narseo Vallina-Rodriguez. 2018. An Empirical Analysis of the Commercial VPN Ecosystem. In *Proceedings of the Internet Measurement Conference 2018* (Boston MA USA, 2018-10-31). ACM, 443–456. doi:10.1145/3278532.3278570
- [11] Vasile C. Perta, Marco V. Barbera, Gareth Tyson, Hamed Haddadi, and Alessandro Mei. 2015. A Glance through the VPN Looking Glass: IPv6 Leakage and DNS Hijacking in Commercial VPN clients. 2015, 1 (2015), 77–91. doi:10.1515/popets-2015-0006
- [12] Reethika Ramesh, Leonid Evdokimov, Diwen Xue, and Roya Ensafi. 2022. VPNalyzer: Systematic Investigation of the VPN Ecosystem. In *Proceedings 2022 Network and Distributed System Security Symposium* (San Diego, CA, USA, 2022). Internet Society. doi:10.14722/ndss.2022.24285
- [13] Patrick Sattler, Matthias Kirstein, Lars Wüstrich, Johannes Zirngibl, and Georg Carle. 2025. Lazy Eye Inspection: Capturing the State of Happy Eyeballs Implementations. In *Proceedings of the 2025 ACM Internet Measurement Conference* (2025-10-28). 213–221. arXiv:2412.00263 [cs] doi:10.1145/3730567.3732925
- [14] David Schinazi and Tommy Pauly. 2017. Happy Eyeballs Version 2: Better Connectivity Using Concurrency. doi:10.17487/RFC8305 Num Pages: 15.
- [15] Dave Thaler, Richard P. Draves, Arifumi Matsumoto, and Tim Chown. 2012. Default Address Selection for Internet Protocol Version 6 (IPv6). doi:10.17487/RFC6724 Num Pages: 32.
- [16] WhatIsMyIPAddress.com. 2023. *Privacy Policy*. <https://whatismyipaddress.com/privacy-policy>
- [17] Maya Ziv, Liz Izhikevich, Kimberly Ruth, Katherine Izhikevich, and Zakir Durumeric. 2021. ASdb: a system for classifying owners of autonomous systems. In *Proceedings of the 21st ACM Internet Measurement Conference* (Virtual Event, 2021-11-02). ACM, 703–719. doi:10.1145/3487552.3487853

A Ethics

Our analysis uses data from WhatIsMyIPAddress.com. This data includes IP addresses associated with specific computers, triggered by user requests to the website. This data is collected as part of regular site operation and is consistent with the site’s published privacy policy [16].

Our study was reviewed by USC IRB (#UP-25-00838). IRB identified our work as non-human-subjects research because although we know requests are associated with individuals, we have no way of identifying specific individuals. As part of our IRB protocol, we agreed not to deanonymize any IP addresses and not to redistribute any data containing user IP addresses.

B Details about Address Prioritization

As described in §5.2, IPv6 de-prioritization occurs because of an interaction between VPN address assignment and address prioritization rules. Those rules are surprisingly complicated and subtle.

This appendix looks at the browser selection process (§B.1) and the standard rules for how addresses are prioritized (§B.2)

B.1 Browser Implementations to Select Protocols

Browsers and operating system libraries combine to look up addresses and select which to use. Table 1 lists where these rules are given and their implementations for three browsers.

B.2 RFC-6724 Address Prioritization

As described in §5.2, IP address prioritization is the root cause of IPv6 deprioritization in VPNs.

IP address prioritization in modern operating systems and browsers follows the guidance of RFC-6724 [15]. The RFC defines labels and precedences for IP addresses of different times, but we summarize those rules to the 7 ranks listed in Table 2. Since VPNs route only to global addresses, only ranks 1, 3, 5, and 6 (marked with *) are relevant to our analysis.

Deprioritization occurs because VPNs using private addresses place IPv4 in ranks 5 and IPv6 in rank 6. With different ranks, only IPv4 is considered for Happy Eyeballs and IPv6 is ignored.

C Leak Evaluation

C.1 Leak Evaluation of All VPN and Prior Studies

In §2.2 we discuss prior studies of VPN leaks. Here we compare VPNs covered in those studies in detail, showing the growth of coverage in subsequent studies.

		step				
		0	1	2	3	4
		User req.	DNS query	Find source addr	Sort	Race
specification			RFC 8305: HE [14]	RFC 6724 [15]		RFC 8305: HE
browser	Safari	OS	...	C library (getaddrinfo)	...	browser
	Firefox	(socket	...	C library (getaddrinfo)	...	browser
	Chrome	syscall)		...	browser	...

Table 1: Definition of algorithms and location of their implementations for web browsers.

Rank	Source	Destination	Reason
1*	IPv6 GUA	IPv6 GUA	Same label (1) and high precedence (40); most preferred pair.
2	IPv6 ULA	IPv6 ULA	Same label (13) and precedence (3); for internal communication.
3*	IPv4 public	IPv4 public	Same label (4) and precedence 35; valid global IPv4 path.
4	IPv4 private	IPv4 private	Same label (4) with reachable private addressing; for LANs.
5*	IPv4 private	IPv4 public	Same label (4) and precedence 35, but requires NAT traversal.
6*	IPv6 ULA	IPv6 GUA	Label mismatch (13 vs. 1) causes strong penalty.

Table 2: Address prioritization rules from RFC-6724 [15]. Label groups addresses into the same class (only equality matters). Precedence is a numeric preference score (higher is preferred).

In Table 4 compare coverage by VPN from each study. Our study sees 123 different VPNs, more than prior studies (although we do not see 48 they considered).

More importantly, Table 4 highlights that our study shifts the focus from binary evaluation (leak or non-leaks) in prior work to the fraction of website visitors that show leaks (in our work). In 35 VPNs (marked with % in the “us” column) we are able to evaluate the fraction of visitors that leak, and another 100 VPNs (“U” under “us”) we see but do not have a large enough sample to report a fraction.

C.2 Test Specifications

On MacOS, we tested VPNBook and VPNGate, and in both cases observed IPv6 leakage. For VPNBook, the provider distributes multiple OpenVPN and PPTP configuration files, but our tests were limited to one configuration file, CA149 OpenVPN bundle. For VPNGate, which offers a wide range of servers and protocols, we tested only the top-listed server public-vpn-40.opengw.net using OpenVPN on macOS. We also tested TurboVPN on both Android and iOS clients, where we again observed IPv6 leakage.

Smoothing Rough Edges of IPv6 in VPNs

Provider	2014	2018	2022	Us (2025)	Provider	2014	2018	2022	Us (2025)	Provider	2014	2018	2022	Us (2025)
1ClickVPN	-	-	-	U	holaproxy	-	-	-	U	StreamVia	-	-	-	U
1VPN	-	-	-	0.03%	Home & Away VPN	-	-	-	U	Streisand	-	-	N	-
4ebur.net	-	-	-	U	Hotspot Shield	Y	-	N	5.54%	StrongVPN	Y	-	N	U
AdGuard	-	-	-	U	IBVPN	-	-	-	U	super_unlimited	-	-	-	U
AirVPN	Y	-	N	0.17%	IPBurger VPN	-	-	-	U	SurfEasy	-	-	Y	-
Algo	-	-	N	-	IPVanish	Y	-	N	9.47%	SurfShark	-	-	N	U
AngelVPN	-	-	-	U	ishaanvpn	-	-	-	U	TeklanVPN	-	-	-	U
Anonine	-	-	N	U	Ivacy VPN	-	-	N	U	Telleport	-	-	-	U
AnonymousVPN	-	-	-	U	IVPN	-	-	N	1.28%	titantunnelvpn	-	-	-	U
Apple Private Relay	-	-	-	0.00%	Jego	-	-	-	U	TorGuard	N	-	N	5.51%
Astrill VPN	Y	-	Y	U	K2VPN	-	-	N	-	Touch VPN	-	-	N	9.09%
Atlas VPN	-	-	N	U	Kaspersky	-	-	N	-	Troywell VPN	-	-	-	U
Avast Secureline	-	-	N	U	KeepSolid	-	-	-	U	Trust VPN	-	-	-	U
Avira Phantom	-	-	N	-	KeepSolid VPN	-	-	N	11.64%	Trust.Zone	-	-	N	U
Azure VPN	-	-	N	U	Le VPN	-	Y	N	U	TunnelBear	Y	-	N	5.61%
BelkaVPN	-	-	-	U	LimeVPN	-	-	-	U	Turbo VPN	-	-	Y	56.85%
Best Proxy Switcher	-	-	-	U	LiquidVPN	-	Y	-	-	UltraSurf VPN	-	-	-	U
BestVPN	-	-	N	-	MaxiVPN	-	-	-	U	Unlocator	-	-	-	U
Betternet	-	-	N	-	Mozilla VPN	-	-	N	-	Unspysable	-	-	N	-
BlancVPN	-	-	-	U	Mullvad	N	-	N	0.35%	Urban VPN Desktop	-	-	N	-
Blaze VPN	-	-	-	U	MyExpatriotNetwork	-	-	-	U	uVPN	-	-	-	U
BolchVPN	-	-	N	U	myiphider	-	-	-	U	VanishedVPN	-	-	-	U
BoxPN	-	-	-	U	Namecheap	-	-	N	-	VeePN	-	-	N	7.01%
Browsesec	-	-	-	0.01%	Netshade	-	-	-	U	VPN Bridge	-	-	-	U
BTGuard	-	-	-	U	Njalla	-	-	-	U	VPN Hotspot	-	-	N	-
Buffered VPN	-	Y	-	U	NordLayer	-	-	-	U	VPN Owl	-	-	N	-
BulletVPN	-	Y	-	U	NordVPN	-	-	N	6.39%	VPN Plus	-	-	N	-
Bullguard	-	-	N	-	Norton Secure VPN	-	-	Y	-	VPN Pro	-	-	N	-
BullVPN	-	-	-	U	OpenVPN Access Server	-	-	N	-	VPN Proxy Master	-	-	N	-
Cactus VPN	-	-	N	U	Outline	-	-	N	-	VPN Super	-	-	N	-
CCryptoVPN	-	-	-	U	OVPN	-	-	N	0.34%	VPN.ac	-	-	N	U
Celo	-	-	-	U	Panda VPN	-	-	N	-	VPN.Asia	-	-	-	U
Cisco Secure Web Gateway	-	-	-	3.89%	Perfect Privacy	-	-	N	0.33%	VPN.ht	-	Y	-	U
Cryptostorm	-	-	N	U	Perimeter81	-	-	-	U	VPN99	-	-	-	U
CyberGhost	-	-	N	14.31%	personalVPN	-	-	-	U	VPNAccount	-	-	-	U
EarthVPN	-	-	-	U	PhantomPeer VPN	-	-	-	U	VPNArea	-	-	-	U
EasyHidIP	-	-	-	3.35%	PIA	N	-	N	8.95%	VPNBook	-	-	N	18.44%
elr1c	-	-	-	U	Private Tunnel	-	Y	N	-	VPNGate	-	-	-	5.93%
embracevpn	-	-	-	U	Private VPN	-	Y	N	-	VPNhack	-	-	-	U
Encrypt.me	-	-	N	-	PrivateVPN	-	-	-	U	VPNJantit	-	-	-	9.34%
ExpressVPN	Y	-	N	6.47%	Proton VPN	-	-	N	3.40%	VPNLite	-	-	N	-
F-Secure Freedome	-	-	N	0.01%	proxywave	-	-	-	0.02%	vpnproxymaster	-	-	-	U
FastestVPN	-	-	N	U	Psiphon	-	-	N	U	VpnShop	-	-	-	U
Fastly	-	-	-	0.00%	PureVPN	Y	-	N	9.10%	VPNTunnel	-	-	-	U
FastVPN	-	-	-	9.53%	RedshieldVPN	-	-	-	U	vpntunnelfly	-	-	-	0.11%
FlowVPN	-	-	-	U	Riseup	-	-	N	-	VPNUK	-	-	N	U
Free VPN	-	-	N	-	SaferVPN	-	-	-	U	VyprVPN	N	-	N	U
Free VPN Planet	-	-	-	1.97%	Safum	-	-	-	U	Windscribe	-	-	N	U
FreeOpenVPN	-	-	-	U	SandVPN	-	-	-	U	WiTopia	-	-	-	U
freeprovpn	-	-	-	U	Seed4.me	-	Y	-	-	Working VPN	-	-	-	U
FreeVPN	-	-	-	U	ShellFire	-	-	-	U	WorldVPN	-	Y	-	U
FrootVPN	-	-	-	U	SmartDNSProxy	-	-	-	U	X-VPN	-	-	-	U
GhostPath	-	-	-	U	Snap	-	-	-	U	ZenMate	-	-	N	-
Google Fi VPN	-	-	-	U	Specdify	-	-	N	-	ZoogVPN	-	Y	N	U
Google One VPN	-	-	-	0.08%	Spotflux	-	-	-	U					
Goose VPN	-	-	N	U	Star VPN	-	-	N	-					
Hide My Ass	Y	-	N	U	Steganos	-	-	N	-					
Hide.me	-	-	N	0.04%										
HideIPVPN	-	Y	N	U										
HideMyIP	-	-	-	U										

Table 3: Comparing leaks across all VPNs across our study (last column) and prior studies in 2015 [11], 2018 [10], and 2022 [12], and our study (the rightmost column). 2015 study tested 5 OSes(PC, mobile), 2018 covered only macOS, 2022 study tested in macOS and Windows. U stands for Undersampled (less than 100 observations per day).

2014	2018	2022	Us (2025)	Count	Subtotals
–	–	N	–	29	36
–	Y	–	–	3	
–	–	Y	–	2	
–	Y	N	–	2	
–	–	–	U	74	100
–	Y	–	U	3	
–	–	N	U	16	
–	Y	N	U	3	
N	–	N	U	1	
Y	–	N	U	2	
Y	–	Y	U	1	
–	–	–	%	13	35
–	–	N	%	12	
N	–	N	%	3	
Y	–	N	%	6	
–	–	Y	%	1	
13	11	78	135	171	171

Table 4: Combination of outcomes of VPN leak reports studies in 2015 [11], 2018 [10], and 2022 [12], and our study (the rightmost column).