Residential Proxy Detection

Sophia Barnes Stephanie Vezich Tamayo

Abstract

Residential Proxy as a Service (RESIP) providers have become a primary tool for automating large-scale, often malicious web activities while masking traffic as coming from legitimate domestic users. Current detection methods leverage the measurable difference between TCP and TLS Round-Trip Times (RTT) to identify proxied connections. This research explores both the characterization of RESIP traffic and the potential for providers to evade these timing-based detections. First, we registered as proxy devices on bandwidth sharing services and analyze traffic routed through our active RESIP nodes, finding distinct patterns including distribution of destination ASNs, gateway vs. destination traffic ratio, time of day use, and client TLS fingerprints. Second, we demonstrate how a RESIP provider can evade the RTT signal by intentionally delaying TCP acknowledgements. By manipulating the acknowledgement timing in a simulated environment, we show that timing-based detection can be effectively bypassed, necessitating more robust RESIP detection methods.

1 Introduction

Residential proxies function much like traditional network proxies (e.g., commercial VPNs) except that they are deployed in residential or cellular IP space instead of data centers [2, 3]. They work as a backconnect proxy; a connection initiated by a proxy client sends traffic to a gateway server, which then forwards the traffic to a RESIP node, which in turn forwards traffic to the targeted server (Figure 1). RESIP providers allow customers to configure several attributes of their desired exit nodes, from geographical location to stickiness to the same

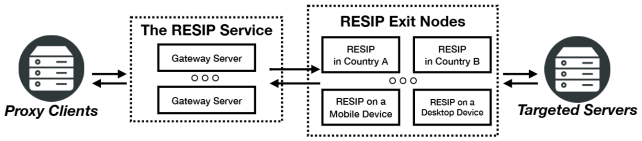

Figure 1: The Backconnect Proxy Mode of RESIPs

node. While RESIP node recruitment has been framed as voluntary or profit-generating bandwidth sharing by RESIP providers, past work has shown that in reality, traffic is often proxied through a residential user's IP without their knowledge via potentially unwanted programs (PUPs) or 3rd party SDKs on mobile apps [2, 3]. They are often used to relay malicious traffic such as accessing sensitive websites, spamming, DDoS, etc. because relaying traffic through residential nodes enables them to evade typical proxy detection techniques which target exit nodes in commercial data centers [2, 3].

Given the growth of RESIPs in recent years, the current research focuses on three distinct inquiries that may be of interest to destination services hoping to prevent inbound RESIP traffic. First, we provide a descriptive analysis of relayed traffic by volunteering an IP address for two RESIP services and examining outbound packets. This analysis allows us to characterize relayed traffic, answering questions such as common destinations for RESIP clients, protocols used by RESIP services, etc. Second, we build a RESIP detection technique from the perspective of destination servers by measuring the difference between TLS and TCP RTTs. Finally, given the adversarial nature of this problem, we implement one method RESIP services could use to evade detection (namely, injecting delay in RTT acknowledgements) and

discuss how destination services might respond.

2 Related Work

Prior research on characterizing relayed traffic has generally focused on client-side observation; that is, researchers sign up as customers of RESIP services and send requests through them, as opposed to observing from the point of view of the exit node. The findings have measured several dimensions of RESIP traffic, including scale, traffic patterns, and abuse potential.

Mi et al. subscribed to five RESIP providers and built a web crawler that sent HTTP requests with special tags through each provider's proxy network to web and DNS servers under their control [2]. These tags enabled their servers to record the exit node's IP address and DNS resolver. They collected 6.18 million unique RESIP IPs across more than 230 countries and 52000 ISPs over a period of 4 months. They found that RESIP services were often used for several malicious activities, including ad click fraud (specifically, automating requests to spoof user engagement with ads), blackhat SEO promotion of target sites, and fast-flux hosting (rapid cycling through many residential IPs for phishing or malware distribution purposes). They later extended this approach to mobile proxies, and similarly found that RESIP exit nodes are often used in ad fraud to simulate legitimate ad interactions and in distributing payloads for large-scale botnets such as Hajime and Mirai [3].

Yang et al. subscribed to five RESIP providers in China specifically and similarly sent tagged requests to controlled servers [5]. They collected over 8 million RESIP IPs over 6 months. In line with prior research, they found that the proxy pool was highly transient, with 91% of RESIP lifetimes lasting less than 10 days. They also found that 80% of RESIP exit nodes served at least 1 malicious traffic flow. Common malicious activities included cryptojacking (in which the attacker uses the exit node's resource to mine cryptocurrency), botnet payload distribution, and accessing corporate networks of sensitive organizations including government agencies.

Leveraging Mi et al.'s dataset, Choi et al. compared geospatial and behavioral patterns between 1.05 million open proxy IPs and 6.4 million RESIP IPs [2, 6]. They found that Turkey and India were more common locations for residential proxy exit nodes while China and the United States were relatively more common for open proxies. Both were commonly used for malicious activities, with 86% of residential proxy IPs and 79% of open proxy IPs appearing on threat denylists. However, RESIP IPs showed a lower rate of use for verified attacks (0.27% relative to 6.97% of open proxies), which the authors took to indicate that residential proxies are more commonly used for low-and-slow activities (e.g., spam, credential stuffing) rather than overt attacks.

While the above works focus primarily on proxy population measurement and abuse characterization, Chiapponi et al. introduced BADPASS, a server-side detection technique based on timing discrepancies induced by RESIP forwarding [4]. Their key observation is that proxied connections typically create two distinct TCP sessions (client-to-proxy and proxy-to-server) while preserving an end-to-end TLS session between client and server, which causes the TCP and TLS round-trip times to diverge at the destination. BADPASS operationalizes this intuition by measuring the difference between TCP-RTT and TLS-RTT on inbound connections and classifying connections as proxied when the difference exceeds an empirically chosen threshold (50 ms). In a multi-month measurement across multiple commercial RESIP providers, they report very low false positive rates and low false negative rates at this threshold. Our work builds on BADPASS by (1) independently validating that commercial RESIP paths exhibit large RTT gaps across multiple regions, and (2) experimentally demonstrating an evasion mechanism that suppresses the BADPASS timing signal by delaying acknowledgements without terminating TLS.

3 Methods

3.1 Traffic analysis

We complement prior work that observed RESIP relayed traffic from the client of the proxy network by instead observing from the perspective of the exit node. We volunteered a host's IP address as an exit point for two popular bandwidth sharing services, Honeygain and Packetstream. Prior work has indicated that bandwidth provided by these nodes is exclusively used to relay RESIP traffic, making our approach a straightforward way to collect data from the exit node itself [1].

We captured the resulting outbound traffic and analyzed the pcap traces to characterize the traffic that proxy

customers relay through residential proxy IPs.[1] This vantage point enables analyses that are difficult from the client side, such as direct observation of the tunnel protocol between RESIP gateway servers and the exit node, TLS Client Hello fingerprint diversity across clients, per-gateway temporal patterns, and connection concurrency. While prior work took a somewhat indirect approach to analyze relayed traffic (e.g., comparing exit node IPs against threat intelligence feeds), our unique contribution is direct measurement of all traffic transiting through a particular RESIP exit node, including measurements of timing, volume, protocols, and destinations.

3.2 Validating BADPASS

Our detection methodology follows the BADPASS observation that a RESIP "backconnect" relay naturally decouples transport-layer and application-layer timing at the destination [4]. In a direct connection, both the TCP three-way handshake and the TLS handshake are between the same two endpoints (client and server). In a RESIP connection, the destination server completes its TCP handshake with the exit relay (the residential node), but the TLS session remains end-to-end between the original client and the destination: the relay forwards TLS records without terminating or re-encrypting them. As a result, the server-observed TCP round-trip time primarily reflects the exit relay–to–server path, while the server-observed TLS timing reflects the longer client–to–server path.

For each inbound HTTPS connection captured at the destination server, we compute two timing features from server-side packet traces: (1) RTT_{TCP}, defined as the time between the server's SYN-ACK and the corresponding ACK that completes the TCP handshake, and (2) RTT_{TLS}, defined as the time between the server's first TLS handshake flight (e.g., ServerHello and subsequent handshake records) and the first client-to-server TLS response. We then compute the RTT gap as $\Delta = RTT_{TLS} - RTT_{TCP}$. To validate BADPASS against a commercial provider, we purchased bandwidth from an IPRoyal residential proxy plan and issued HTTPS connections through the service to destination servers that we controlled in Oregon, South Carolina, the Netherlands, Tokyo, and São Paulo. For each region, we also collected a direct-connection baseline.

[1] github.com/sophiabarness/resip-detection/blob/main/proxy-traffic-analysis/README.md

3.3 Evasion of RTT-Based Detection

To evaluate evasion under a RESIP-like relay architecture, we deployed servers in three separate regions to emulate the gateway, residential relay, and destination roles in a backconnect path: client → gateway → residential relay → destination. Our system consists of: (1) a first-hop gateway that accepts a client proxy connection and forwards it to an upstream relay, and (2) a RESIP relay node with two functions: forwarding encrypted traffic to the destination without terminating TLS, and manipulating server-facing handshake timing.

Our evasion goal is to suppress the BADPASS timing signal $\Delta = RTT_{TLS} - RTT_{TCP}$ without terminating TLS. To do so, the relay measures the true client-to-relay distance using a pre-flight probe. After receiving a SOCKS connection request, the relay sends a fake SOCKS success response back to the client. This triggers the client to immediately send its first application flight (usually a TLS ClientHello), allowing the relay to measure the end-to-end round-trip time (RTT_{probe}) before even initiating a connection to the final server.

Implementing this evasion requires per-flow state. After measuring RTT_{probe} for a given connection, the relay initiates the actual server-side connection and reports the measured value to a local timing-control daemon via localhost UDP datagrams. When the daemon observes the destination's SYN-ACK, it schedules the release of the final outbound ACK so that it reaches the server approximately RTT_{probe} later than it otherwise would. This yields per-connection timing control while preserving the original TLS handshake and application data end-to-end (Figure 2).

In our prototype, the timing-control mechanism is implemented as privileged, in-path packet interception on the relay host, and the relay observes server-facing handshake packets and defers transmission of the final ACK.

4 Results

4.1 Traffic analysis

We collected 69.2 hours of proxy traffic, consisting of 38803 TCP streams to 3419 unique destination IPs across 340 ASNs in 42 countries, and base our results on inspecting packet metadata. The proxy node initiated 42813 TCP connections (SYNs) and performed

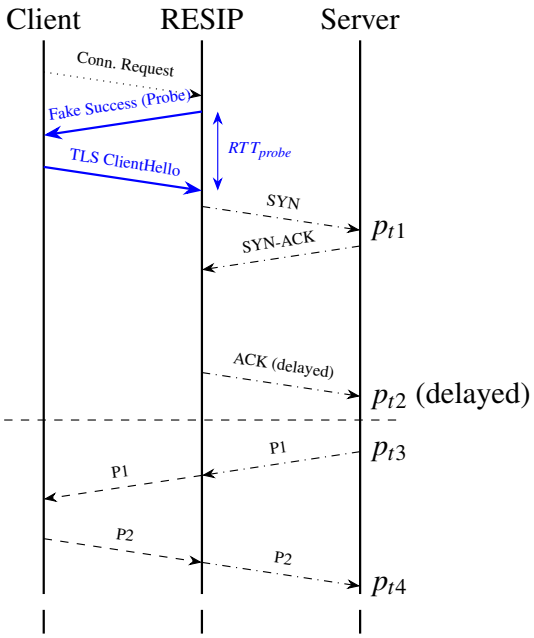

Figure 2: Connection timing with Pre-flight RTT probing and delayed ACK (evasion mode).

649 DNS lookups. We observed significantly higher throughput with Honeygain (112 MB/hr vs. 17 MB/hr) and greater destination IP diversity (2694 unique destinations vs. 725). We also saw relatively high client diversity, with 42 distinct cipher suite fingerprints from Honeygain's 21621 handshakes and 23 fingerprints across Packetstream's 12261 handshakes. The full descriptive statistics and destination-category tables are provided in Appendix A.

In terms of tunneling architecture, Honeygain showed greater attention to security. It routes all proxied traffic through UDP tunnels to 10 unique gateway servers and rotated across them over the capture session. Honeygain also uses DNS-over-HTTPS for name resolution, making DNS lookups invisible in plaintext. Of 24562 TCP streams, 23172 were TLS-encrypted and only 1 included plaintext data. In contrast, with Packstream the proxy node maintained persistent connections to 8 primary relay servers (DNS queries for client.packestream.io resolved to exactly these relay IPs), and uses plaintext DNS instead of DNS-over-HTTPS. Of 14241 TCP streams, 12627 were TLS encrypted and 602 carried plaintext data.

Consistent with traffic generated by automated scripts (vs. organic user requests), we observed bursty connection timing for both providers, with coefficients of variation (CV) of 2.09 for Honeygain and 1.57 for Packetstream, and high connection reuse (94% of Honeygain connections and 98% of Packetstream connections were to IPs used more than once). However, given Packetstream's less secure architecture, we observed several patterns consistent with relatively greater abuse. First, there were 346 TCP connections to SMTP servers targeting email exchanges and 10 queries to the disposable email API api.mail.tm. There were also 222 DNS queries for login.microsoftonline.com, suggesting credential harvesting or attempted account logins. The Honeygain traffic did not show these patterns, since the encrypted UDP tunnel architecture effectively prevents protocol-level abuse like SMTP relay. These findings align with prior work showing RESIP use in malicious activity (particularly low-and-slow activity); however, our exit node vantage point contributes a new finding that the prevalence and nature of abuse is determined not only by RESIP customers but also the architectural constraints of the RESIP service itself [2, 5, 6].

In terms of the distribution of destinations (extracted from SNI hostnames for Honeygain and DNS queries for Packetstream), dominant categories are e-commerce/retail and ad tech across both providers, consistent with web scraping and ad fraud use cases (Appendix Table A1). PacketStream shows broader diversity including email/credential infrastructure (SMTP servers, Microsoft/Google login endpoints, disposable email APIs), cryptocurrency (etherscan.io, basescan.org), and gaming (Steam, Square Enix). Honeygain is more focused on US retail and ads. Notably, both providers show a small amount of traffic to government and institutional resources (NIH/PubMed, an Indonesian government site, a Nigerian telecom authority portal). Geographically, these IPs map to the United States (63% of Honeygain and 67% of Packetstream destination IPs) and Canada (26% of Honeygain and 22% of Packstream destination IPs); however, remaining destinations fell across 40 countries. The predominance of US and Canada destinations is likely due to the location of the exit node being in the US.

4.2 RESIP detection

Table 1 summarizes the median RTT gap for direct connections versus traffic routed via a commercial IPRoyal RESIP relay across five global regions. These measurements serve as a direct validation of the BADPASS re-

sult: across all five regions, direct traffic remains tightly clustered around 4 ms while commercial residential proxy traffic consistently exhibits a much larger RTT gap, with medians between 245.89 ms and 264.91 ms. In other words, our independent measurements reproduce the central BADPASS claim that authentic RESIP paths induce a large and easily separable timing discrepancy relative to direct connections.

Table 1: Regional median RTT gaps for direct traffic versus commercial IPRoyal RESIP relay.

Region	n_{direct}	Median direct	n_{relay}	Median relay
Oregon (US-West)	864	3.76 ms	968	249.21 ms
South Carolina (US-East)	831	4.03 ms	975	262.38 ms
Netherlands (EU)	850	3.84 ms	977	245.89 ms
Tokyo (Asia)	841	4.08 ms	932	264.91 ms
Sao Paulo (South America)	850	4.48 ms	623	263.41 ms

4.3 Detection evasion

We validated this deployment using a cross-country setup with the client at Stanford, a first-hop superproxy in Oregon, an exit relay in Tokyo, and the target server in South Carolina. We compared three modes: a direct baseline, the multi-region relay path with evasion disabled, and the same deployment with IPC-based evasion enabled.

Table 2: RTT-gap summary for direct, relay, and evasion modes.

Mode	n	Median RTT Gap	Evasion Rate ($\leq$50 ms)
Direct	953	5.59 ms	87.6%
Relay path (no evasion)	1000	147.55 ms	0.0%
Relay path (evasion)	1001	**6.33 ms**	70.5%

As shown in Table 2, this relay deployment produces a RESIP-like baseline timing signal without terminating TLS. Without timing manipulation, the relay path yields a median RTT gap of 147.55 ms and an evasion rate of 0%, placing every connection above the 50 ms BADPASS threshold. With IPC-based timing manipulation enabled, the median collapses to 6.33 ms and 70.5% of flows fall below the threshold, bringing the evasion-enabled relay much closer to the direct baseline. Figure 3 shows the full per-connection distributions for all three modes.

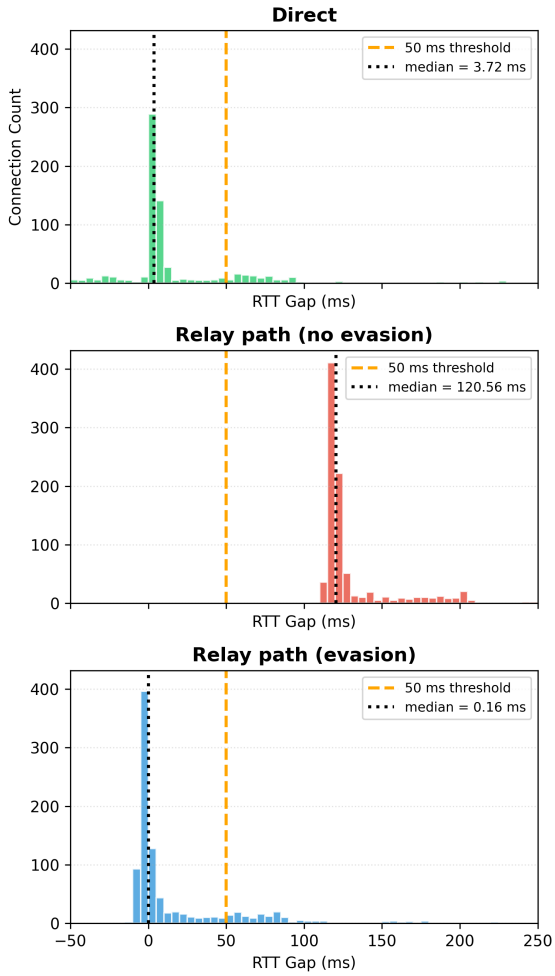

Figure 3: Per-connection RTT-gap distributions for the direct baseline, the relay path without evasion, and the relay path with evasion.

5 Known limitations

Our traffic analysis dataset was collected from a single residential proxy IP address that we enrolled in two RESIP services over a roughly two-day period. Compared to prior work that collected client-side measurements over multi-month periods, our findings are necessarily limited in scope, both in IP space and time. It would be informative to repeat the same process across a more diverse set of RESIP exit nodes and providers, and to run longer captures, to better generalize traffic and geospatial patterns. For example, our geographic results are likely biased by the location of the single exit node, and our setup prevents us from measuring how frequently providers rotate or cycle client connections across dif-

ferent residential IPs.

Our evasion evaluation also makes an important deployment assumption. The timing manipulation requires the ability to intercept and delay specific TCP handshake packets on the relay's server-facing path. In practice, this requires privileged access on the relay host itself and could be unfeasible for RESIP providers acting in userspace to implement without requiring OS support, elevated permissions, or explicit user consent. Additionally, artificially delaying the final ACK can also introduce operational side effects such as retransmissions under tight timeout settings, and timing artifacts that a detector could treat as anomalous.

6 Conclusion

This research has characterized the landscape of residential proxies from both a detection and an evasion perspective. Through global traffic analysis, we identified distinct architectural and behavioral fingerprints of RESIP traffic. We then validated current timing-based detection methods, confirming they are robust against existing commercial providers. Finally, we demonstrated that a multi-region relay deployment with stateful timing manipulation can substantially suppress the BADPASS timing signal without breaking TLS tunneling. Our work underscores the ongoing arms race between proxy providers and security researchers, necessitating more granular and adaptive detection strategies that go beyond simple RTT thresholding.

7 AI Use Disclosure

AI coding agents were utilized in the process of this research.

References

[1] Huang, R., Zhao, D., Mi, X., & Wang, X. (2024). Shining Light into the Tunnel: Understanding and Classifying Network Traffic of Residential Proxies. *arXiv preprint arXiv:2404.10610*.

[2] Mi, X., Feng, X., Liao, X., Liu, B., Wang, X., Qian, F., Li, Z., Alrwais, S., Sun, L., & Liu, Y. (2019). Resident evil: Understanding residential ip proxy as a dark service. In *2019 IEEE symposium on security and privacy (SP)* (pp. 1185–1201). IEEE.

[3] Mi, X., Tang, S., Li, Z., Liao, X., Qian, F., & Wang, X. (2021). Your phone is my proxy: Detecting and understanding mobile proxy networks. In *Proceeding of ISOC Network and Distributed System Security Symposium (NDSS), 2021*.

[4] Chiapponi, E., Dacier, M., Thonnard, O., Fangar, M., & Rigal, V. (2022). Badpass: Bots taking advantage of proxy as a service. In *International Conference on Information Security Practice and Experience* (pp. 327–344). Springer.

[5] Yang, M., Yu, Y., Mi, X., Tang, S., Guo, S., Li, Y., Zheng, X., & Duan, H. (2022). An extensive study of residential proxies in China. In *Proceedings of the 2022 ACM SIGSAC conference on computer and communications security* (pp. 3049–3062).

[6] Choi, J., Abuhamad, M., Abusnaina, A., Anwar, A., Alshamrani, S., Park, J., Nyang, D., & Mohaisen, D. (2020). Understanding the proxy ecosystem: A comparative analysis of residential and open proxies on the internet. *IEEE Access*, 8, 111368–111380.

A Traffic Analysis Results

Table A1: Destination categories observed via SNI hostnames (Honeygain, $n = 225$) and DNS queries (PacketStream, $n = 507$), with representative examples.

Category	Honeygain	PacketStream
E-commerce / Retail	walmart.com, target.com, ebay.com, petsmart.com, saksfifthavenue.com, instacart.com, macys.com, chewy.com, rei.com, temu.com, anthropologie.com, victoriassecret.com, coles.com.au, falabella.com	ebay.com, homedepot.com, poshmark.com, farfetch.com, zillow.com, danmurphys.com.au, agoda.com, ticketmaster.com, vividseats.com, shopify.com
Ad-tech / Tracking	googlesyndication, doubleclick, criteo, pubmatic, appnexus, casalemedia, 33across, rubiconproject, openx, adthrive, pinterest, bidswitch	doubleclick, adsrvr.org, criteo, clarity.ms, adroll, stackadapt, plista, bidswitch, exelator, crwdcntrl.net
Search / Portals	google.com, bing.com, google.ro	google.com, yahoo.com, google.co.jp
Social Media	tiktok.com, youtube.com, instagram.com	facebook.com, instagram.com, threads.net, x.com, discord.com, truthsocial.com, deviantart.com, behance.net, twitch.tv
Video / Streaming	youtube.com, i.ytimg.com	youtube.com, googlevideo.com, kick.com, movetv.com
Travel / Hospitality	expedia.com, priceline.com, tripadvisor.com, agoda.com, hotpads.com, movoto.com	expedia.com, expedia.de, agoda.com, trip.com, flightradar24.com, marinetraffic.com
Finance / Crypto	bloomberg.com, wsj.net, barrons.com	wsj.com, seekingalpha.com, reuters.com, investing.com, marketbeat.com, stocktwits.com, etherscan.io, basescan.org
News / Media	web.archive.org, medicalxpress.com, espn.com, abclocal.go.com	axios.com, reuters.com, economist.com, alternet.org, sltrib.com, inquirer.com, boxofficepro.com
Government / Civic	jogjakota.go.id, ncbi.nlm.nih.gov, apa.org	ncbi.nlm.nih.gov, mtnnigeria.net, arztsuche.116117.de, openstreetmap.org
Gaming	—	steampowered.com, square-enix.com, playgame.zone, animaljam.com
Email / Credential	mail.google.com	smtp.gmail.com, gmail-smtp-in, login.microsoftonline.com, login.live.com, login.okta.com, api.mail.tm, uschangeofmail.online
Suspicious / DGA-like	soju88e.cyou, ip.oxylabs.io	8573412.xyz, 7622083.xyz, 293821237.xyz, chesselo2000.cfd, polaamenang.net, salamoonder.com, vegasinocasino.si

Table A2. Descriptive statistics of captured proxy traffic, overall and by provider.

Metric	Overall	Honeygain	PacketStream
Capture duration (hours)	69.1	26.9	42.2
Total traffic (MB)	3,730	3,017	713
Throughput (MB/h)	54.0	112.1	16.9
Unique destination IPs	3,419	2,694	725
Distinct ASNs	340	276	64
Countries reached	42	40	20
Gateway/relay servers	18	10	8
TCP streams	38,803	24,562	14,241
TCP SYNs (new connections)	42,813	24,633	18,180
Connection rate (SYNs/s)	—	0.25	0.12
Mean SYN inter-arrival (s)	—	3.93	8.37
CV of inter-arrival time	—	2.09	1.57
Connection reuse (%)	—	94	98
TLS Client Hellos	33,882	21,621	12,261
Distinct TLS fingerprints	65	42	23
TLS-encrypted streams (%)	92.2	94.3	88.7
Plaintext streams (%)	1.6	0.004	4.2
No-data/aborted streams (%)	6.2	5.7	7.1
DNS domains observed	649	139	510
DNS resolution method	—	DoH (Cloudflare)	Plaintext