# Poster: The Impact of the Client Environment on Residential IP Proxies Detection

Elisa Chiapponi
EURECOM
France
elisa.chiapponi@eurecom.fr

Marc Dacier
KAUST
Saudi Arabia
marc.dacier@kaust.edu.sa

Olivier Thonnard
Amadeus IT Group
France
olivier.thonnard@amadeus.com

**ACM Reference Format:**
Elisa Chiapponi, Marc Dacier, and Olivier Thonnard. 2023. Poster: The Impact of the Client Environment on Residential IP Proxies Detection. In *Proceedings of the 2023 ACM Internet Measurement Conference (IMC '23), October 24–26, 2023, Montreal, QC, Canada.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3618257.3624993

## 1 INTRODUCTION

Residential IP Proxies (RESIPS) enable proxying out requests through a vast network of residential devices (GATEWAYS) without inserting any information revealing it. While RESIPS can be used for legitimate purposes, previous works also associated them with malicious activities [2, 5, 7].

In [3], we have proposed a new server-side method to identify connections proxied through RESIPS: we compare the Round Trip Times (RTTS) at the TCP and TLS layers. RESIPS break the TCP session between client and server but maintain the TLS one end-to-end. Thus, TCP packets sent by the server travel only to the GATEWAY while the TLS ones, after reaching it, are forwarded inside the RESIP infrastructure and then sent to the client. The $\text{RTT}_{TCP}$ and the $\text{RTT}_{TLS}$ mirror this path difference. Hence, in a RESIP connection, the $\delta_{RTT}$ (=$\text{RTT}_{TLS}$- $\text{RTT}_{TCP}$) is significantly higher than in a direct one. Our 4-months measurement campaign shows that a $\delta_{RTT}$ higher than 50ms corresponds to a RESIP request.

Our technique measures the RTTs at the server side. Hence, beyond reflecting the distance traveled by packets, the $\text{RTT}_{TLS}$ includes the client processing time. In [3], we performed connections among well-connected data center machines using Python scripts. This is not a common way for end users to connect to websites. Generally, they use personal devices and navigate the Internet with web browsers. The preliminary results obtained with this new work show that these factors only impact marginally the measurement.

Moreover, our technique does not recognize only RESIP connections. It identifies all proxies that break the TCP session but not the TLS one. Mobile TCP Terminating Proxies (MTTPS) are an example of this kind of proxy [8]. MTTPS belong to specific mobile ISPS and break the TCP session between the device and a server at the antenna. This enhances performance since the probability of packet loss is higher in the electromagnetic wave transmission between the device and the antenna than in the wired part of the connection. Breaking the TCP, the device needs to resend lost packets only for the short path between the antenna and the device itself.

While the exact percentage of ISPS using MTTPS is unknown, mobile network connections are widely used nowadays. It is thus reasonable to assume that MTTPS play a role in generating false positives (FPS) to our technique. However, we believe the $\delta_{RTT}$ observed in MTTP connections is smaller than in RESIPS ones. With MTTPS, the $\delta_{RTT}$ is due to short device-antenna distances. In contrast, RESIPS involve significant packet detours due to their globally distributed infrastructures and do not use GATEWAYS near the destination server [1]. Our initial findings support this idea and indicate a higher threshold could reduce FPS caused by MTTPS.

## 2 CLIENT ENVIRONMENT ANALYSIS

Numerous mobile networks and web browsers exist nowadays. In this experiment, we consider only a subset of them to preliminary assess possible impacts on the RESIP detection.

We use a personal computer (PC) and a mobile phone, both located in France, as client machines. With them, we query a server in the UAE that runs our RESIP RTT detection [3]. We perform 20 connections to it with each of the following configurations. With the personal computer, we connect to the Internet using three setups. The first one uses a 5GHZ Wi-Fi signal from a residential access point. In the other setups, we use the mobile device included in the experiment as a hotspot for the personal computer. First, the mobile phone is connected to the residential Wi-Fi, then to the 4G network thanks to the French provider SFR. In this way, we analyze the impact of the two types of network, as well as the effect of using a hotspot, on the $\delta_{RTT}$.

We test these settings using no VPN, NordVPN (WireGuard protocol) and Tor. We know that both these tunneling techniques break the TCP session but not the TLS one and are thus detected by our technique. We use them as a reference to compare with the $\delta_{RTT}$ caused by MTTPS. For these connections, we randomly choose a new exit location for each request. We perform connections using the two most used web browsers, Google Chrome and Microsoft Edge [6]. For Tor connections, we use the Tor Browser. We manually perform the queries to the server, to reproduce a real user interaction. We also send queries with Python scripts to have a benchmark for the delays introduced by web browsers.

For the mobile device, we send requests through Wi-Fi and 4G (from SFR) with Google Chrome. In this way, we assess the mobile network delay without any hotspot contribution.

Table 1 displays the median value plus or minus the median absolute deviation of the twenty collected $\delta_{RTT}$ values for each configuration. When no VPN is used and connections are made via Wi-Fi (with/without hotspot), the $\delta_{RTT}$ remains below the threshold


*IMC '23, October 24–26, 2023, Montreal, QC, Canada*

ACM ISBN 979-8-4007-0382-9/23/10.
https://doi.org/10.1145/3618257.3624993

**Table 1: Tested client environment combinations.**

| Device | VPN | Network | Browser | $\delta_{RTT}$ (ms) |
|---|---|---|---|---|
| PC | No VPN | Wi-Fi | Chrome | 28±16.5 |
| | | | Edge | 13±6.5 |
| | | | Python | 3±1 |
| | | Hotspot (Wi-Fi) | Chrome | 36±10 |
| | | | Edge | 32±15.5 |
| | | | Python | 3.5±2.5 |
| | | Hotspot (4G) | Chrome | 117±56 |
| | | | Edge | 77±18 |
| | | | Python | 52.5±6.5 |
| | Nord VPN | Wi-Fi | Chrome | 127±4.5 |
| | | | Edge | 158±14.5 |
| | | | Python | 140±20 |
| | | Hotspot (Wi-Fi) | Chrome | 127.5±1.5 |
| | | | Edge | 129±1.5 |
| | | | Python | 124.5±5 |
| | | Hotspot (4G) | Chrome | 188±34 |
| | | | Edge | 170±8 |
| | | | Python | 219.5±70.5 |
| | Tor | Wi-Fi | Tor | 278±50.5 |
| | | Hotspot (Wi-Fi) | Tor | 982.5±274 |
| | | Hotspot (4G) | Tor | 888±183 |
| Mobile phone | No VPN | Wi-Fi | Chrome | 10.5±6 |
| | | 4G | Chrome | 51.5±14.5 |

of 50ms for both devices. However, in the cases where connections are made with no VPN through 4G, the $\delta_{RTT}$ generally exceeds 50ms. This discrepancy seems to suggest that SFR utilizes MTTPS.

We can see that using a browser introduces a slightly larger delay than when using a Python script. However, the $\delta_{RTT}$ values remain below the threshold when there is no other delaying factor (NordVPN, Tor, 4G). This indicates that the use of web browsers alone does not lead to FPS. Furthermore, when using a hotspot to reach the Wi-Fi signal, the $\delta_{RTT}$ increases compared to a direct Wi-Fi connection. However, this factor alone does not result in a $\delta_{RTT}$ exceeding 50ms.

When considering NordVPN and Tor connections using 4G, the $\delta_{RTT}$ is generally higher than using Wi-Fi. This indicates that NordVPN, Tor, and the SFR mobile network each contribute to the total delay independently. When a connection combines two of these factors, their delays accumulate, resulting in a larger $\delta_{RTT}$.

Considering 4G connections from the mobile device, we observe that the $\delta_{RTT}$ exceeds the threshold but remains close to its value. Moreover, this value is much smaller than the ones for Tor and NordVPN. Hence, our intuition about the low $\delta_{RTT}$ delay introduced by MTTPS compared to other proxies breaking the TCP session is confirmed for this provider. To further support this initial finding and generalize it, we analyze mobile FPS connections to real-world web domains. This approach enables us to see whether the same pattern occurs with other mobile network providers utilizing MTTPS.

**Real-World Mobile Connections.** Our detection technique is implemented in front of real-world domains suffering from bot attacks. For a representative week (01/06/23-07/06/23), we collected the IP addresses and the corresponding $\delta_{RTT}$ values of connections reaching the *Page x* of eight different protected domains. Only successful purchases lead to *Page x*. Thus, we assume that only genuine users arrive there. We then consider any connection to *Page x* with an $\delta_{RTT}$ greater than the 50ms threshold as a FP. Among these FPS, we consider the connections originating from mobile networks thanks to the Mobile Carrier Database of Digital Element [4] and we calculate the corresponding median $\delta_{RTT}$. This value is 88.5. With the elements at our disposal, we can not know with certainty that the only factor that increases the $\delta_{RTT}$ in these connections is the usage of the mobile network. There could be other factors, on top of it, that increase the delay. However, based on the results in the previous section, the reported median delay is more compatible with the connections using SFR 4G with a web browser and/or hotspot than the ones leveraging NordVPN or Tor. This seems to suggest that the delaying factor in these connections is MTTPS.

In [3], 96.57% of RESIP connections had a $\delta_{RTT}$ higher than this median value. This additional evidence strengthens our idea that MTTPS create a significantly lower $\delta_{RTT}$ compared to other proxies that break the TCP but not the TLS session.

## 3 CONCLUSIONS

Our analyses indicate that commonly used client-side features (web browsers, hotspot) do impact, yet only marginally, our RTT-based RESIP detection technique. Moreover, they show that the MTTPS $\delta_{RTT}$ is much smaller than the RESIP one. This suggests that we can reduce FPS caused by MTTPS by highering the threshold. A thorough sensitivity analysis is required for proper threshold adjustment.

## REFERENCES


[1] E. Chiapponi, M. Dacier, and O. Thonnard. 2023. Inside Residential IP Proxies: Lessons Learned from Large Measurement Campaigns. In *2023 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*.

[2] E. Chiapponi, M. Dacier, O. Thonnard, M. Fangar, M. Mattsson, and V. Rigal. 2022. An industrial perspective on web scraping characteristics and open issues. In *2022 52nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks - Supplemental Volume (DSN-S)*.

[3] E. Chiapponi, M. Dacier, O. Thonnard, M. Fangar, and V. Rigal. 2022. BADPASS: Bots Taking ADvantage of Proxy as a Service. In *Information Security Practice and Experience (ISPEC)*.

[4] Digital Element. 2023. *Carrier Data Insights.* https://www.digitalelement.com/solutions/user-context/carrier-data/

[5] X. Mi, X. Feng, X. Liao, B. Liu, X. Wang, F. Qian, Z. Li, S. Alrwais, L. Sun, and Y. Liu. 2019. Resident Evil: Understanding Residential IP Proxy as a Dark Service. In *2019 IEEE Symposium on Security and Privacy (S&P)*.

[6] Oberlo. 2023. *Most popular web browser in 2023.* https://www.oberlo.com/statistics/browser-market-share

[7] M. Yang, Y. Yu, X. Mi, S. Tang, Y. Guo, S. Li, X. Zheng, and H. Duan. 2022. An Extensive Study of Residential Proxies in China. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*.

[8] R. Zullo, A. Pescapé, K. Edeline, and B. Donnet. 2019. Hic Sunt Proxies: Unveiling Proxy Phenomena in Mobile Networks. In *2019 Network Traffic Measurement and Analysis Conference (TMA)*.