Poster: Evaluating Cascading-VPN Performance

Sebastian Pahl*, Florian Adamsky*, Daniel Kaiser†, and Thomas Engel†
*Institute of Information Systems (iisys), Hof University of Applied Sciences, Germany
{name.surname}@hof-university.de
†University of Luxembourg, Luxembourg
{name.surname}@uni.lu

***Abstract*—Virtual Private Network (VPN) protocols provide means for establishing secure inter-network links. However, they do not provide anonymity. VPN providers can monitor both ends of the connection. On the other hand, Onion Routing offers very good anonymity properties but offers significantly less throughput than typical VPN setups.**

An interesting compromise is using several VPN servers connected in series (cascading VPN). This paper evaluates the throughput of two VPN protocols, WireGuard and OpenVPN, in a cascading environment.

I. Introduction

Virtual Private Network (VPN) protocols such as OpenVPN [1] and WireGuard [4] provide a secure tunnel to other networks. When not fully controlling a VPN server, the VPN provider and the data centre where the server is located have to be trusted. Commercial VPN providers typically promise not to record any user activity, however, this is often not the case [1], [11]. Onion Routing [6] offers high anonymity, but at the cost of a significant performance loss.

A compromise between performance and anonymity is using several VPN servers in series, shown in Figure 1, which we refer to as cascading VPN. Traffic is then routed through multiple VPN servers. If the VPN servers are located in separate jurisdictions, attacks against anonymity become even more challenging to execute, limiting an attacker to more sophisticated attacks such as timing attacks. A series of VPNs also helps to bypass censorship infrastructure.

The contributions of this paper are (1) a preliminary evaluation of the performance of WireGuard and OpenVPN when used in a cascading setup, and (2) a fair performance comparison of these VPN protocols on modern hardware featuring extensions such as Advanced Encryption Standard New Instructions (AES-NI), confirming results of [9].

II. Experimental Setup

For our experiments, we established the three-hop VPN setup shown in Figure 1 using Linux network namespaces. It comprises five namespaces, two for the endpoints (R1 and R2) and three for the nodes (VPN1–VPN3). Each of the four connections features a VPN tunnel. All virtual Ethernet interfaces are limited to a maximum throughput of 1 Gbit/s.

We introduce realistic latency and packet loss corresponding to intra-European 1 Gbit/s Internet connections based on guarantees from Verizon [10]: 30 ms latency and a packet loss

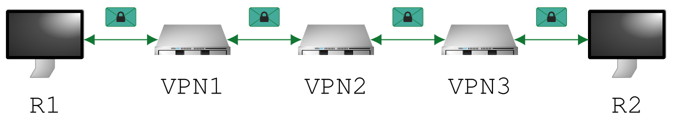

Fig. 1. Architecture of our virtualised experimental cascading VPN setup. All green connections are encrypted network tunnels. All nodes are in separate network namespaces.

of 0.5 %. Further, we introduce 1 ms of jitter. Latency values are chosen from a normal distribution with a 25 % correlation, accounting for the fact that latency is not entirely random. We will refer to these settings as Internet conditions.

We ran our experiments on a desktop computer with an AMD Ryzen 7 3700X 8-Core Processor, 16 GB of RAM, and Kernel 5.10.7-3-MANJARO. Our experiments did neither fully utilize the cores nor the memory, so we are neither processor nor memory limited. We tested WireGuard v1.0.20200827 and OpenVPN 2.5.0, with local and Internet conditions, respectively. We did not evaluate Internet Protocol Security (IPsec) since it has a bad reputation based on its complexity [5].

We use OpenVPN in Transport Layer Security (TLS) mode, with cipher AES-256-GCM and UDP as transport protocol. To measure throughput we use `iperf3` 3.9, which runs in client mode on R1 and server mode on R2. Besides our cascading setup, we also tested a simple single-hop (2 tunnels) setup for comparison. Our scripts are available on GitLab[2] and allow reenacting our experiments.

III. Experimental Results

The results of our experiments with local and Internet conditions, respectively, are shown in Figure 2. The experiments ran for 10 min. We took intermediate values for every second.

The results are as follows: WireGuard performs 39.6 % better than OpenVPN with TCP, partly because WireGuard runs in the Kernel while OpenVPN has a user space TUN/TAP implementation, and partly because they handle replay windows differently. Generally, UDP performs better than TCP under Internet conditions, which is expected as UDP is not effected by latency. The higher variance of the throughput of TCP can be explained by the randomness in latency and packet loss leading to variance in the occurrence of out-of-order packets and retransmits. The variance is especially high if TCP and the VPN protocols' replay windows

[1]https://openvpn.net/

[2]https://gitlab.com/spahl/eval-serial-vpn

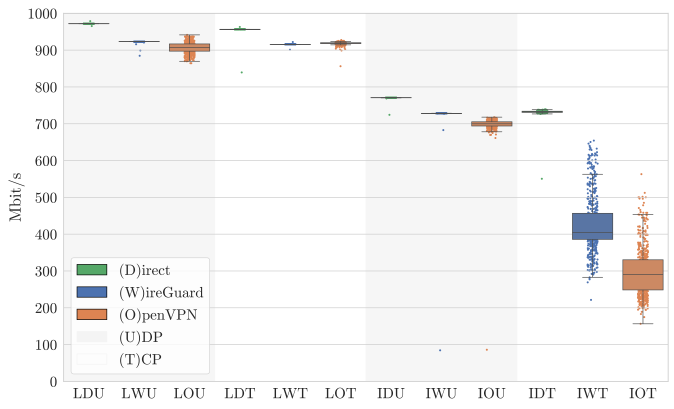

Fig. 2. Three hop: Throughput results where data was sent from R1 to R2. The first letter stands for (I)nternet conditions or (L)ocal conditions. The second letter describes the protocol: (D)irect, (W)ireGuard, or (O)penVPN. The third letter stands for (U)DP or (T)CP.

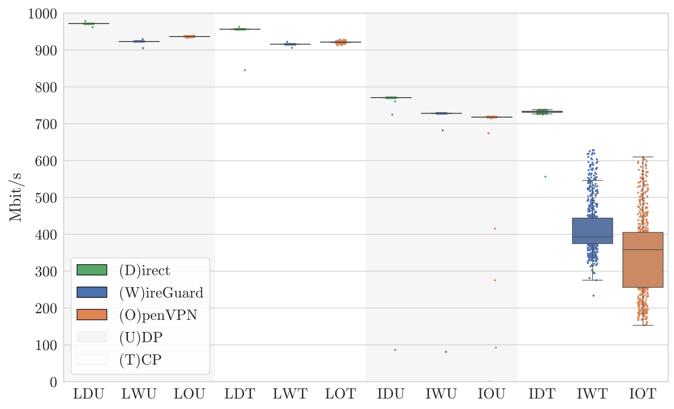

Fig. 3. Single hop: Throughput results where data was sent from R1 to R2. The first letter stands for (I)nternet conditions or (L)ocal conditions. The second letter describes the protocol: (D)irect, (W)ireGuard, or (O)penVPN. The third letter stands for (U)DP or (T)CP.

are used together. Local conditions perform better than Internet conditions, which is expected as packet loss and latency reduce throughput for all TCP connections. Further, under local conditions, OpenVPN slightly outperforms WireGuard, which is likely an artefact of the optimal conditions.

Comparing the median throughput of the three and the single hop experiments under Internet conditions, only the OpenVPN throughput increases significantly. Otherwise, there is no significant change in throughput as shown in Figure 3.

IV. Related Work

We are not aware of related work covering modern VPN performance tests used in a cascading configuration to the best of our knowledge. Our experimental setup described in Section II is close to the two-hop SOCKS5 implementation by Mullvad [7], which means providers offer such a solution off the shelf. Mullvad also offers a two-hop WireGuard setup were packets are end-to-end encrypted [8].

Donenfeld [4] evaluated the performance of OpenVPN versus WireGuard in a single VPN setup. His results show a 4-times performance gain of WireGuard over OpenVPN. A possible explanation for OpenVPN's poor performance in Donenfeld's evaluation is not utilizing the AES-NI extensions in conjunction with the worse performing AES-CBC mode of operation. We consider our results more practical relevant, since modern CISC-CPUs support AES-NI. The results of Osswald et al. [9] match ours in this respect; the added contribution of our paper is the investigation of a cascading setup.

Tor [3], the most widely used anonymity network, provides much stronger anonymity properties compared to our setup. However, Tor suffers from poor performance, an issue that has been researched extensively [2].

V. Future Work and Conclusion

In this paper, we compared the performance of WireGuard and OpenVPN, and evaluated both protocols in a cascading environment. We measured the percentage change of the median throughput of WireGuard against OpenVPN with UDP and TCP in a single and three hop configuration under Internet conditions. Our preliminary results show that WireGuard outperforms OpenVPN in cascading environments.

In future work, we would like to test these protocols in real Internet conditions. Additionally, we would like to integrate Onion Routing capabilities into VPN protocols, which we believe to result in the best compromise between anonymity and network performance. This will improve censorship resistance, security, and privacy.

References

[1] Report: No-Log VPNs Reveal Users' Personal Data and Logs. [Online]. Available: https://www.vpnmentor.com/blog/report-free-vpns-leak/

[2] M. AlSabah and I. Goldberg, "Performance and security improvements for tor: A survey," *ACM Computing Surveys (CSUR)*, vol. 49, no. 2, pp. 1–36, 2016.

[3] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation Onion Router:," Defense Technical Information Center, Tech. Rep., 2004. [Online]. Available: http://www.dtic.mil/docs/citations/ADA465464

[4] J. A. Donenfeld, "WireGuard: Next generation kernel network tunnel," in *Proceedings 2017 Network and Distributed System Security Symposium*. Internet Society, 2017. [Online]. Available: https://www.ndss-symposium.org/ndss2017/ndss-2017-programme/wireguard-next-generation-kernel-network-tunnel/

[5] N. Ferguson and B. Schneier, "A Cryptographic Evaluation of IPsec," p. 28, 1999.

[6] D. Goldschlag, M. Reed, and P. Syverson, "Onion Routing for Anonymous and Private Internet Connections," *Communications of the ACM*, vol. 42, pp. 39–41, 1999.

[7] Mullvad. (2021) Different entry/exit node using WireGuard and SOCKS5 proxy. [Online]. Available: https://mullvad.net/en/help/different-entryexit-node-using-wireguard-and-socks5-proxy

[8] ——. (2021) Multihop with WireGuard. [Online]. Available: https://mullvad.net/de/help/multihop-wireguard

[9] L. Osswald, M. Haeberle, and M. Menth, "Performance Comparison of VPN Solutions," in *Proceedings of the 1st ITG Workshop on IT Security (ITSec)*. Universitätsbibliothek Tübingen, 2020.

[10] verizon. (2021) Monthly IP Latency Data — Verizon Enterprise Solutions. [Online]. Available: https://enterprise.verizon.com/terms/latency

[11] J. Youngren. Hidden VPN Owners Unveiled: 101 VPNs Run by 23 Companies. [Online]. Available: https://vpnpro.com/blog/hidden-vpn-owners-unveiled-97-vpns-23-companies/

Evaluating Cascading-VPN Performance

Sebastian Pahl[1], Florian Adamsky[1], Daniel Kaiser[2], and Thomas Engel[2]

[1]Institute of Information Systems (iisys), Hof University of Applied Sciences, Germany
[2]Faculty of Science, Technology and Medicine (FSTM), University of Luxembourg, Luxembourg

iisys Institute of Information Systems at Hof University

UNIVERSITÉ DU LUXEMBOURG

Abstract

Virtual Private Network (VPN) protocols provide means for establishing secure inter-network links. However, they do not provide anonymity. VPN providers can monitor both ends of the connection. On the other hand, Onion Routing offers very good anonymity properties but offers significantly less throughput than typical VPN setups.
An interesting compromise is using several VPN servers connected in series (cascading VPN). This paper evaluates the throughput of two VPN protocols, WireGuard and OpenVPN, in a cascading environment.

Motivation

- Onion-Routing offers very good anonymity at the cost of performance
- Cascading VPN servers provide a good compromise between anonymity and performance

Cascading VPN

- Interesting compromise between anonymity and performance
- If VPN servers are located in separate jurisdictions, attacks against anonymity become even more difficult to execute

Architecture

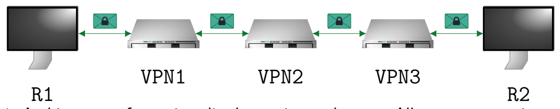

Figure 1: Architecture of our virtualised experimental setup. All green connections are encrypted network tunnels. All nodes are in separate network namespaces.

Experimental Setup

- Figure 1 contains five Linux network namespaces including two endpoints (R1 and R2) and three VPN nodes (VPN1–VPN3)
- All virtual Ethernet interfaces are limited to a maximum throughput of 1 Gbit/s
- Evaluated WireGuard v1.0.20200827 and OpenVPN 2.5.0 in AES-256-GCM mode with local and Internet conditions, respectively
- Simulate Internet conditions with 30 ms latency with a jitter of 1 ms and a packet loss of 0.5 %

Preliminary Results for Single-Hop

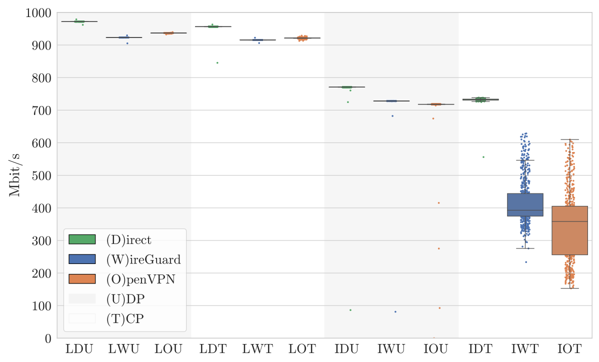

Figure 2: Single hop: Throughput results where data was sent from R1 to R2. The first letter stands for (I)nternet conditions or (L)ocal conditions. The second letter describes the protocol: (D)irect, (W)ireGuard, or (O)penVPN. The third letter stands for (U)DP or (T)CP.

Preliminary Results for Three-Hop

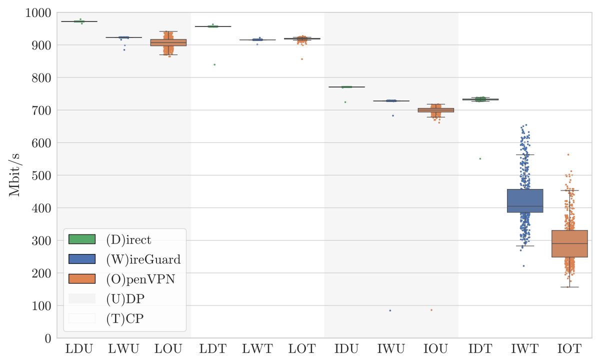

Figure 3: Three hop: Throughput results where data was sent from R1 to R2. The first letter stands for (I)nternet conditions or (L)ocal conditions. The second letter describes the protocol: (D)irect, (W)ireGuard, or (O)penVPN. The third letter stands for (U)DP or (T)CP.

Experimental Results

- Experiments run for 10 min and we took intermediate values for every second
- UDP is more performant than TCP under Internet conditions
- Wireguard performs 39.6 % better than OpenVPN, with TCP under Internet conditions
  - WireGuard runs in the Linux Kernel
  - OpenVPN runs as a TUN/TAP device in user-space
  - OpenVPN and WireGuard handle replay windows differently
- Local conditions perform better than Internet conditions

Conclusion

- Performance evaluation of WireGuard and OpenVPN in a cascading environment
- Wireguard performs 39.6 % better than OpenVPN in our preliminary results

Future Work

- Integrate Onion Routing in WireGuard
- Evaluate VPN protocols in real Internet conditions and compare it with the virtualised experiments