A Performance Comparison of WireGuard and OpenVPN

Poster · March 2020

DOI: 10.1145/3374664.3379532

Yuan Cheng
California State University, Sacramento
18 PUBLICATIONS **352** CITATIONS

A Performance Comparison of WireGuard and OpenVPN

Steven Mackey
smackey@csus.edu
California State University,
Sacramento
Sacramento, CA

Ivan Mihov
ivanmihov@csus.edu
California State University,
Sacramento
Sacramento, CA

Alex Nosenko
anosenko@csus.edu
California State University,
Sacramento
Sacramento, CA

Francisco Vega
fvega@csus.edu
California State University,
Sacramento
Sacramento, CA

Yuan Cheng
yuan.cheng@csus.edu
California State University,
Sacramento
Sacramento, CA

ABSTRACT

A fundamental problem that confronts virtual private network (VPN) applications is the overhead on throughput, ease of deployment and use, and overall utilization. WireGuard is a recently introduced light and secure cross-platform VPN application. It aims to simplify the process of setting up a secure connection, while utilizing the multi-threading capability and minimizing the use of bandwidth. There have been several follow-up studies on WireGuard since its birth, most of which focus on the security analysis of the protocol. Despite the author's claim that WireGuard has impressive wins over OpenVPN and IPsec, there is no rigorous analysis on its performance to date. This paper presents a performance comparison of WireGuard and its main rival OpenVPN on various metrics. We construct an automated test framework and deploy it on a total of eight nodes, including remote AWS instances and local virtual machines. Our test results clearly show two main edges that WireGuard has over OpenVPN, its performance on multi-core machines and its light codebase.

CCS CONCEPTS

• **Security and privacy → Cryptography**; • **Networks → Network performance analysis**; *Network layer protocols.*

KEYWORDS

Virtual private networks, Performance analysis, OpenVPN

ACM Reference Format:
Steven Mackey, Ivan Mihov, Alex Nosenko, Francisco Vega, and Yuan Cheng. 2020. A Performance Comparison of WireGuard and OpenVPN. In *CODASPY '20: ACM Conference on Data and Application Security and Privacy, March 16–18, 2020, New Orleans, LA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

WireGuard [2] is a relatively new product on the mature market of Virtual Private Networks (VPNs). Its main goal is to simplify the process of establishing a secure connection by minimizing the overcomplications that other options have, such as OpenVPN [8], and provide better resource utilization. To establish a VPN connection, users must exchange public keys, and the rest is handled by WireGuard. Users do not have to manage connections, states, or even daemons. Recent studies have been focused on the cryptographic analysis of the WireGuard protocol, including its correctness and several other security properties [3, 5, 7].

The authors of WireGuard claimed that it, in theory, should achieve very high performance and an unoptimized WireGuard already has impressive wins over OpenVPN and IPsec. However, we are not aware of supporting evidences from an objective third party. In this paper, we aim to provide a performance analysis on WireGuard and its main rival on the market, OpenVPN. To the best of our knowledge, this is the first performance study on WireGuard from an unbiased view.

2 DEPLOYMENT

For the performance evaluation, we need a modular and easy way of deploying WireGuard and OpenVPN over multiple hosts, with minimal manual configuration. Since the evaluated VPN solutions have vastly different design choices, we decide to use Ansible [1] as our software provisioning tool. Ansible provides us with flexibility and allows a single command to install, configure, run, and connect both VPN solutions. We develop separate Ansible roles for each VPN and tie them together in a playbook.

WireGuard installs from source in a matter of seconds. It comes with a utility for generating the needed keys, making the deployment quick and effortless. In WireGuard, the notion of server and client is not present, as every host is considered a peer, and we require one configuration file shared among peers.

OpenVPN, on the other hand, has a much larger footprint and makes the installation slower and more convoluted. Developing the automation role for OpenVPN is complex, because it requires the generation of separate configuration files for servers and clients.

Overall, we are able to completely automate the process of installation, configuration, and establishing a connection using both VPN packages; however, the WireGuard automation role, with its

CODASPY '20, March 16–18, 2020, New Orleans, LA
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/1122445.1122456

100 lines of code, takes considerably less effort to develop than the three times longer OpenVPN role.

Our initial deployment is done on two AWS t2.micro instances in two different regions - California (CA) and Ohio (OH). We eventually decide to expand our testing setup and include six additional nodes for a total of eight hosts. Table 1 shows all nodes used and their specifications.

Table 1: Description of The Nodes Used

# of nodes	Type	Location	CPU cores	RAM (GB)	VM
2	AWS	OR	1	1	Y
1	AWS	CA	1	1	Y
1	AWS	OH	1	1	Y
1	AWS	SG	1	1	Y
2	Ubuntu	CA	1	2	Y
1	Centos	CA	4	8	N

3 METHODOLOGY

To perform a comprehensive evaluation, we opt to test in two environments: AWS and local virtual machines. These environments will be described in detail in Section 4. Each environment consists of two nodes with one node acting as a server (i.e., listening on a TCP/UDP socket) and the other acting as a client. For each environment, we deploy WireGuard and OpenVPN using our custom Ansible playbooks. With the VPNs deployed, we then run our test framework. After establishing these baseline metrics, we turn to more in-depth performance analysis.

With reproducability in mind, we construct an automated test framework, which leverages `iperf3` [4] as well as Python's `psutil` library [6]. The framework is deployed to our test nodes and invoked (in either server or client mode) via an ssh connection from a third node (a personal laptop). The results are then parsed and converted into JSON format, which can be eventually summarized into an easily interpreted breakdown. This allows us to rapidly test on disparate environments in a reproducible manner.

4 RESULTS AND ANALYSIS

For all test cases, we run for a duration of 60 seconds, unless otherwise noted.

4.1 AWS

The AWS short-haul testing is conducted on two micro instances both deployed in Oregon (OR). At the beginning, we notice that the throughput is very low for all nodes. Upon further inspection, we discover that the sustained TCP connections are being throttled. In order to account for this, we have to drastically reduce our testing duration from 60 seconds to 10 seconds. Figures 1, 2, and 3 give a breakdown of the short-haul results. Interestingly, UDP performance is much lower than TCP in all cases. Our best guess is that this is more AWS interference. Despite AWS meddling in our tests, we begin to see that WireGuard has clear advantages over OpenVPN. While the results lean in favor of WireGuard, we choose to rely more on our local VM testing, due to the fear that the reduced test duration would be too 'noisy', and due to the strange behaviour of AWS interfering in the testing.

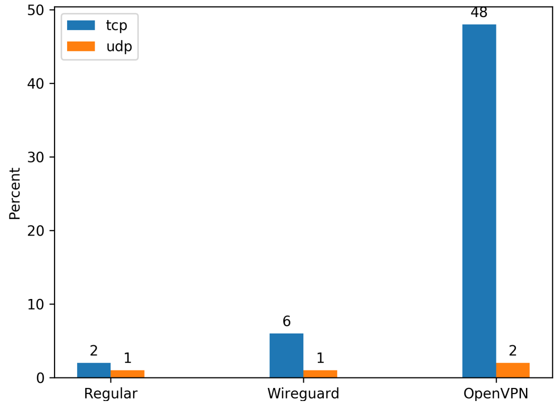

Figure 1: AWS Short-haul Server CPU Usage.

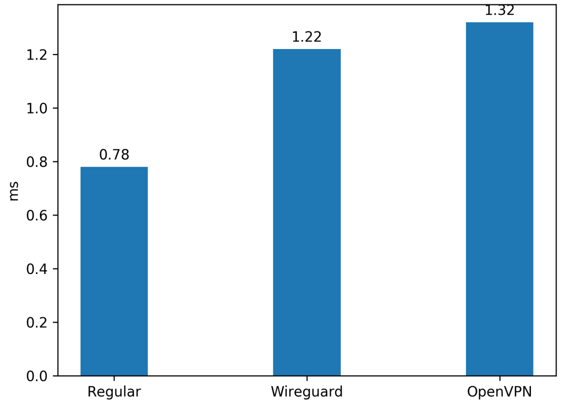

Figure 2: AWS Short-haul Server RTT.

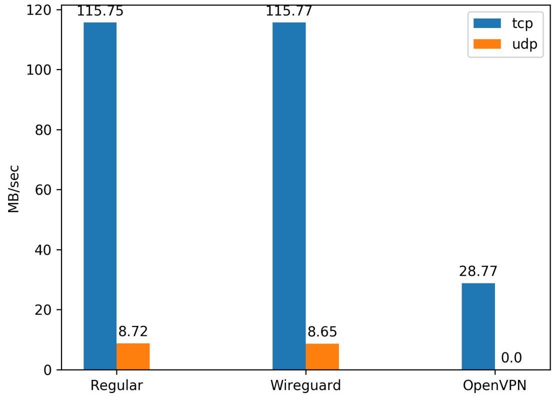

Figure 3: AWS Short-haul Server Throughput.

4.2 Local VM

Our second test environment consists of two Ubuntu virtual machines co-located on the same host. Each virtual machine is allocated 1GB of RAM and 1 virtual CPU. Our rationale behind this test is that we can give a fair test between the VPNs, which would be limited by CPU rather than the network interface controller capacity. The results are largely in line with the AWS environment results, as shown in Figures 4, 5, and 6. At this point, it is clear that WireGuard outperforms OpenVPN in every regard. It is also obvious that WireGuard performs worse than regular sockets, though this is expected. With our baseline results established, we move on to find out why WireGuard performs so much better than OpenVPN.

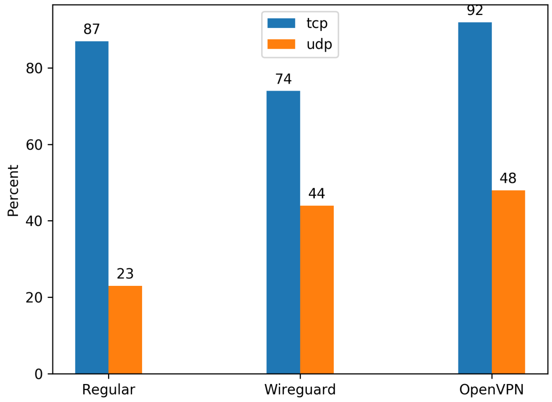

Figure 4: Local VM Server CPU Usage.

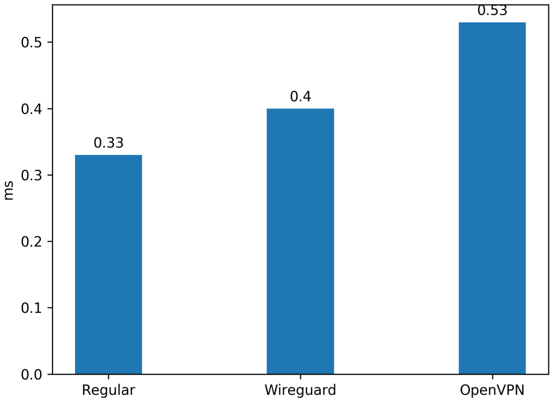

Figure 5: Local VM Server RTT.

4.3 Multi-threaded vs. Single-threaded

We begin our further testing by heuristically confirming the fact that OpenVPN is single-threaded and that WireGuard is multi-threaded. We do this by provisioning our VMs with two cores each and re-running our test framework. The results show that WireGuard can scale effectively with core count, while OpenVPN cannot. This implies that OpenVPN is more dependent on raw clock speed than WireGuard, given that more cores are available.

We now provision our VMs with one core each, and re-run our framework, this time keeping track of the number of system-level context switches, as well as the top five processes sorted by the number of context switches. Surprisingly, WireGuard has far more context switches than OpenVPN, with the WireGuard workers still accounting for the majority of switches. This is contrary to our original assumption that it was context switching that gives WireGuard an edge over OpenVPN.

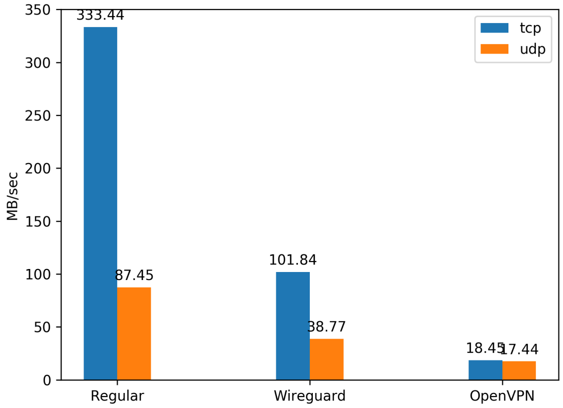

Figure 6: Local VM Server Throughput.

5 FUTURE WORK

There is plenty of more research that can be done on this topic.

- It would be beneficial to test on Windows, iOS, and Android platforms, since the growing amount of Internet users from portable devices.
- It would be interesting to test WireGuard again when it will be mainlined into Linux kernel version 5.6 in 2020.
- We would like to dig deeper into why WireGuard outperforms OpenVPN even on single-core machines.

6 CONCLUSION

Our test results clearly showed the edge that WireGuard has over OpenVPN. It certainly performed better on multi-core machines, where it was able to take full advantage of multi-threading. WireGuard consistently beat OpenVPN in all the testing setups. Another advantage lies within its codebase. Lean and light by design, WireGuard is implemented in just over 4000 lines of code, which will ease auditing and vulnerability finding. It also helps to make the attack surface smaller in comparison to the 60,000 lines of the OpenVPN implementation. Overtime, WireGuard has a chance of becoming a real competitor and occupying a larger share of the market.

REFERENCES

[1] Michael DeHaan. 2012. Ansible. https://www.ansible.com/.
[2] Jason A Donenfeld. 2017. WireGuard: Next Generation Kernel Network Tunnel.. In *NDSS*.
[3] Benjamin Dowling and Kenneth G Paterson. 2018. A cryptographic analysis of the WireGuard protocol. In *International Conference on Applied Cryptography and Network Security*. Springer, 3–21.
[4] Jon Dugan, Seth Elliott, Bruce A Mah, Jeff Poskanzer, and Kaustubh Prabhu. 2014. iPerf3, tool for active measurements of the maximum achievable bandwidth on IP networks. https://iperf.fr/.
[5] Benjamin Lipp, Bruno Blanchet, and Karthikeyan Bhargavan. 2019. A mechanised cryptographic proof of the WireGuard virtual private network protocol. (2019).
[6] Giampaolo Rodola. 2016. Psutil package: a cross-platform library for retrieving information on running processes and system utilization. https://pypi.org/project/psutil/.
[7] Peter Wu. 2019. Analysis of the WireGuard protocol. (2019).
[8] James Yonan. 2001. OpenVPN. https://openvpn.net/.