A study of the additional bandwidth requirements when using VPN Protocols

JP Jones for Top10VPN.com
October 2020

Overview

We set out to test a number of commonly used Virtual Private Network (VPN) Protocols in order to understand the impact in additional bandwidth required when said protocols are used.

Test Setup

We set up a lab environment consisting of 2 identical Virtual Machines (VMs), running as guests under Oracle VM VirtualBox 6.1.4.r136177 software.

These VMs were configured as follows:

- 1 vCPU
- 4GB RAM
- Dynamically allocated storage hosted on an otherwise unused NVMW drive
- 1 x Intel Pro/1000 MT Desktop Network Adaptor, Virtualized, in Bridge mode
- 1 x Intel Pro/1000 MT Desktop Network Adaptor, Virtualized, Internal Network
- Ubuntu 18.04 Server

VM A is the test originator.
VM B is the test file recipient.

Within the Ubuntu Server instances, we ensured that only IPv4 was configured. IPv6 is enabled by default in Ubuntu 18.04, but we disabled it by:

- Setting the following in /etc/default/grub:
  ```
  GRUB_CMDLINE_LINUX_DEFAULT="maybe-ubiquity ipv6.disable=1"
  GRUB_CMDLINE_LINUX="ipv6.disable=1"
  ```

- Adding the following the /etc/sysctl.conf:
  ```
  net.ipv6.conf.all.disable_ipv6 = 1
  net.ipv6.conf.default.disable_ipv6 = 1
  net.ipv6.conf.lo.disable_ipv6 = 1
  ```

- Setting `link-local: []` on `/etc/netplan/00-installer-config.yaml`

The primary network interface was set to a static IP address in the bridged network environment.
The secondary interface was set to a static IP address on a subnet shared by both VMs.

Both network cards were set to 100Mb/s speed, full Duplex.

All system services on the VMs were disabled with the exception of SSHD (providing secure CLI access to the Ubuntu boxes via the network).

The tcpdump program was used to listen to the secondary interface to confirm no packets (e.g. ARP discovery) were being sent across the secondary interface.

The FTP server vsftpd was installed onto machine B, and made available on all interfaces.

The primary network interface is used for all SSH connections.

> A 200Mb test file was created on machine A using the command:
> `dd if=/dev/urandom of=/root/200Mb.dat bs=1M count=200`

This file is of a known size, and filled with random data, so as to be difficult to compress.

Methodology

We test total bandwidth transferred when copying a test file from VM A to VM B. We do this over FTP via the secondary interface, and start measuring bandwidth once we have connected and authenticated via FTP.

We measure bandwidth transferred by looking at the counters on the secondary interface directly before and after file transfer, with the command:
`ip -s link show dev enp0s8`

Which outputs in the format:

```
3: enp0s8: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
fq_codel state UP mode DEFAULT group default qlen 1000
    link/ether 08:00:27:10:6c:ed brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    1683096446 1116132  0       0       0       0
    TX: bytes  packets  errors  dropped carrier collsns
```

```
    58107238    420052    0        0        0        0
```

We record the entries for (RX|TX) bytes and packets into a spreadsheet for both before and after transfer values. We then take the difference in bytes to understand how much data transfer occurred (both for the FTP protocol and the actual data transferred) in the operation. We repeat the test 3 times for every configuration.

For No-VPN tests, we use the FTP server directly on the secondary interface.
For VPN tests, we connect using the VPN Server setup on the secondary interface, and use the FTP server available on the VPN-provided subnet on server B (e.g. 10.8.0.1/24)

For every test, the following steps are taken (with 2 SSH windows open)

Machine	**Action**
B	Run the command `rm /var/ftp/pub/200Mb.dat`

A	Run the command
	`ftp <serverB_VPN_Interface_IP>`
	Login with user:
	`anonymous`
	`cd pub`

B	Run the command `ip -s link show dev enp0s8` To get the pre-test counters
A	Transfer the file with the command `put 200Mb.dat`
B	Watch for the transfer to finish in the CLI window for machine A Immediately run the command `ip -s link show dev enp0s8` To get the post-test counters
B	Log the counters into the spreadsheet (copy and paste)

We first performed a series of tests with no vpn enabled, transferring the 200Mb file using FTP directly from VM A to VM B, to understand what the overhead is from using FTP to transfer the 200Mb test file, resulting in a total size of transfer of 209.32Mb, as calculated from the counters.

Following direct test of transferring the test file, we proceeded to test the following VPN protocols in this environment:

- IKEv2/IPSec
- OpenVPN TCP - With and Without compression enabled
- OpenVPN UDP - With and Without compression enabled
- PPTP
- WireGuard

Specific notes for each VPN Protocol:

IKEv2/IPSec

VM B creates a subnet on 10.8.3.0/24.
We added the IP 10.8.3.253 to the secondary network card, so that the FTP server was available on this IP over the VPN connection from VM A.

The server was installed on VM B, following setup instructions as per:
https://www.digitalocean.com/community/tutorials/how-to-set-up-an-ikev2-vpn-server-with-strongswan-on-ubuntu-18-04-2

OpenVPN

VM B creates a subnet on 10.8.0.1/24
We use the latest OpenVPN client in Ubuntu 18.0.5 LTS, which is OpenVPN 2.4.4 as of time of writing.

We tested a range of ciphers in OpenVPN to gauge what different key sizes meant in terms of bandwidth usage. These were forced by setting the following lines in the OpenVPN config files.

```
ncp-ciphers "AES-128-GCM"
      Cipher AES-128-GCM
```

PPTP

The server was installed on VM B with the help of the following instructions:
https://bobcares.com/blog/install-pptp-server-ubuntu/

WireGuard

The server was installed on VM B with the help of the following instructions:
https://linuxize.com/post/how-to-set-up-wireguard-vpn-on-ubuntu-18-04/

Results

The results from each of the testing procedures have been entered into the following spreadsheet:
https://docs.google.com/spreadsheets/d/1JSbeE0Dz8WXadoXHZw3mSK7TQhj3AFLKWsG4pmDmJaY/edit?usp=sharing

The results are summarised as follows:

VPN Type	Rx Avg (MB)	TX Avg (MB)	Total Avg (MB)	VPN Overhead
No VPN	209.16	0.16	209.32	0.00%
WireGuard	218.43	0.37	218.80	4.53%
IKEv2/IPSec	218.59	7.22	225.82	7.88%
PPTP	218.98	7.58	226.56	8.24%
OpenVPN UDP - Compression	230.23	14.97	245.20	17.14%
OpenVPN UDP - No Compression	230.41	14.99	245.40	17.23%
OpenVPN TCP - No Compression	235.03	16.08	251.11	19.96%
OpenVPN TCP - Compression	234.84	16.69	251.53	20.16%

Note:

Results shown are an average of the 3 tests performed per VPN Type.

OpenVPN results are taken from the AES-256-CBC results rows for consistency, given there is very little resulting difference in size between the differing OpenVPN ciphers.

VPN Overhead is a figure derived from the VPN Type test Total Bandwidth compared to the No VPN Total Bandwidth.

From these results, WireGuard shows as being the most efficient from a bandwidth perspective, adding less than 5% overhead on our FTP transfer tests.

OpenVPN in TCP mode is the most bandwidth-heavy of all tested protocols.