SCHOOL OF APPLIED SCIENCES AND TECHNOLOGY

BACHELOR OF APPLIED
INFORMATION SYSTEMS TECHNOLOGY

Assessing Wireguard

BAIS 4992

Submitted: 8/6/2021

Evaluation of Wireguard as an VPN alternative to OpenVPN and IPSec

Abstract

While Wireguard is relatively simple to set up in VyOS, its minimalism in configuration and tooling could lend to increased administrative overhead if scaling as environment gets more complicated and more clients are added. Implementation in the existing firewall market is also limited. Even still, its implementation in products in other open-source products pfSense have been embroiled in controversy and have been pulled from deployment due to poor code. Conversely, it is quite performant when compared to OpenVPN and IPsec. Average transfer speeds via iPerf3 on site-to-site connections were considerably faster OpenVPN and L2TP for remote client throughput. This is especially the case with the client-to-site-to-site testing in which there was a significant performance penalty for IPsec and OpenVPN. Despite these performance gains, Wireguard's lack of maturity and integration with major vendors makes a niche protocol. However, if speed is a major concern, and the topology simple and managed centrally, Wireguard is worthy of consideration.

Table of Contents

Table of Figures

Introduction

Created as an alternative to IPsec and OpenVPN technologies in the market by Jason Donenfeld, Wireguard aims to be an easy to deploy alternative more traditional VPN technologies. By enabling secure connections through exchanging public keys, and limiting the cryptographic choice, Wireguard appears to reduce the administrative burden that more cryptographically flexible VPN technologies like IPsec and OpenVPN could potentially inflict. This research paper will cover key marketed features of Wireguard as well as provide a brief overview of protocol. It will also aim to compare performance, configuration and evaluate on-going commentary regarding Wireguard, IPsec and OpenVPN. In doing so, it may provide insight into whether Wireguard is truly a disruptive technology or just another alternative to IPsec or OpenVPN.

Wireguard Features

The following sections will discuss some of the key features that Donenfeld highlights as a selling point for Wireguard. Those being its simplicity, crypto key routing, and its DDOS mitigation abilities.

Simplicity

Wireguard makes the claims that it is easier to deploy. Unlike OpenVPN, Wireguard does not need a PKI infrastructure. Unlike IPsec with IKE, Wireguard does not require configuring a separate key exchange layer. Per the maintainer of the project Donenfeld, "[a]fter configuring the interface with a private key and the various public keys with whom it will communicate security, it simply just works" (Donenfeld, 2017, p. 3). On that front, Donenfeld is correct. There is demonstrably less configuration and complexity when compared IPsec or OpenVPN as shown in the subsequent sections. Where IPSec requires configuring additional GRE or LT2P interfaces for site-to-site or client configurations, OpenVPN and Wireguard do not require an additional tunneling protocol to be configured. Unlike OpenVPN, Wireguard does not require configuration of certificates in order to work. Security and identity of each party in Wireguard is done entirely via private and public keys.

Another area in which Wireguard is simpler is lack of cryptographic choice. While OpenVPN and IPsec provide a level of customization in terms of its cryptographic algorithms, Wireguard is cryptographically opinionated and is set up to use what its maintainers believe to be the most secure algorithm at the time (Donenfeld, 2017, p. 3). As such, there is no need to decide to encryption or hashing. See the Table 1 for the current primitives used by Wireguard. Provided that Wireguard is up to date and the maintainers remain vigilant in ensuring primitives are secure, it is less likely that administrators will implement something insecure.

Table 1 Cryptographic Primitives

Primitive	**Use case**
ChaCHa20/Poly1305	Symmetric encryption and authentication
Curve25519	ECDH
BLAKE2s	Hashing, keyed Hashing
SipHash24	Hashtable Keys
HKDF	Key Derivation

Crypto Key Routing

Another major feature Donenfeld advertises about Wireguard is crypto key routing. Peers are identified via a public 32-byte Curve25519 point key. Public keys are then associated with a set of allowed IP addresses. Each interface also has a private key. Traffic is routed and encrypted/decrypted based on the IP address and the public key. If a packet is received on the Wireguard interface with a different than expected allowed internal IP address and key, the packet is rejected. Similarly, if a packet is received with a different than expected public key, the packet is also rejected. On sending traffic, packets will be encrypted using the public key associated with destination IP address. For Roaming clients, clients simply need to setup the public socket, and public key. Provided the server has the public key of client, once the server receives a packet with the public key, it will update its table with the clients public IP address(Donenfeld, 2017, p. 3).

DDOS Mitigation and Cookies

Donenfeld concedes that the Curve25519 encryption is computational expensive when compared to AES-256 which most modern desktop x86 processors have built in acceleration for. This characteristic can be manipulated for CPU-exhaustion attacks. To mitigate this, the receiver can opt to do two things. First option, provided the message is deemed authentic, is the receiver can reply with a cookie reply message. This is a less computationally heavy response informing the sender that it is busy. This reduces the affect the effect of amplifying attacks(Donenfeld, 2017, p. 13). If the receiver cannot authenticate the message, it will not respond to any unauthenticated packets. This makes it invisible to illegitimate peers and network scanners.

While the Donenfeld concedes that replay attack could trick Wireguard into regenerating ephemeral keys, a 12-byte encrypted TAI64 timestamp is included in each packet and is tracked on each device running Wireguard against specific peers (Donenfeld, 2017, p. 7). Wireguard will discard any packets that are equal or less than timestamp kept for each peer. This ensures that even if an attacker replays a packet or if the attacker manages to forge an initiation packet, because the TAI64 timestamp would be equal or less than what is existing on the responders, the packet would be discarded and would not interrupt any on-going session.

Protocol Breakdown

There are 4 types of Wireguard message. Those are Handshake Initiator, Handshake Response, Data, Cookie Reply. During testing, the former three were able to be captured and will be used as examples in the following sections. Under normal operations, upon initiation of communication, a two-way handshake is exchanged between the devices via the Handshake Initiator and a Handshake response. This then followed by Data type Messages. The following is an exchange that was captured during the speed testing described in later sections.

```
1     0.000000   192.168.191.1   192.168.191.130 WireGuard  190
      Handshake Initiation, sender=0x7B09510F
2     0.029961   192.168.191.130 192.168.191.1   WireGuard  134
      Handshake Response, sender=0x1B140D05, receiver=0x7B09510F
3     0.030886   192.168.191.1   192.168.191.130 WireGuard  74
      Keepalive, receiver=0x1B140D05, counter=0
4     0.153479   192.168.191.1   192.168.191.130 WireGuard  170
      Transport Data, receiver=0x1B140D05, counter=1, datalen=96
```

The following subsections will describe each message in greater detail.

Packet 1: Handshake Initiator Message (Type 1)

Based on the Wireshark dissector, Initiate messages are 148 Bytes in length and include the type, a 3-byte reserved space, sender index, 32 Byte Ephemeral Key, 32 byte encrypted key, 12 Byte encrypted timestamp and 16 Byte MAC1 and MAC2.

The following is a graphical representation of an initiator Packet that:

Type (1 Bytes)	Reserved (3 bytes)
Sender Index (4 Bytes)	
Initiator Ephemeral Public Key (32 Bytes)	
Encrypted initiator Public Static Key (48 Bytes)	
Encrypted Timestamp (28 Bytes)	
mac1 (16 Bytes)	mac2 (16 Bytes)

The following is a Raw Wireguard Header capture during Wireguard testing.

```
0000    01 00 00 00 0f 51 09 7b d2 14 2e eb 6e 18 f6 e3
0010    be 12 d1 bf d8 71 77 07 33 bc 8d 00 dc 9a 55 e7
0020    f9 c9 50 df 31 34 cf 25 93 df 1f 11 b6 9f aa 9a
0030    ed b9 f0 89 2e e7 b0 a3 a4 8c 26 76 b6 b1 bb 6a
0040    88 11 b3 d7 23 1d c3 cf 4c 55 83 45 63 53 26 07
0050    7e 96 28 7a 15 c6 33 b6 a8 d6 be 8d b0 3b b8 56
0060    33 33 ea 4e 44 2c fe c6 4c 55 d0 fc c3 d9 28 56
0070    0a a8 7d d1 3f 77 9e 53 84 98 3c 97 cd a9 48 ad
0080    61 b2 c2 92 00 00 00 00 00 00 00 00 00 00 00 00
0090    00 00 00 00
```

The first 8 bits highlighted (1 Byte) defines the type of Wireguard packet is. This is followed by 3 bytes that are reserved. Per the white paper, this is done to make the first four bytes readable as a little-endian integer (Donenfeld, 2017, p. 13). Following the first 4 bytes is the sender index. This is a value generated by the packet originator that is used to identify sender locally to the receiving device. Next is the Encrypted Initiator ephemeral public key which is created a via a random Curve25519 private key, a derivative public key which are used to ensure perfect forward secrecy(Wu, 2019, p. 10). Following this the 32-byte static encrypted key which appears to be 48-byte in length. After that is 28 Bytes of Encrypted TAI64N time stamp. Finally, there are two MAC values for integrity and authenticity. MAC1 is 16 byte Blake2s keyed hash. The key is derived via 32-byte Blake2s Hash of a UTF-8 literal "mac1----" and the receiver's Public Key. The data that is hashed with this key is the message starting from the type up until the mac1 field (Donenfeld, 2017, pp. 10–11). Secondary MAC2 is not used under normal network load situations.

Per Wu, Handling of initiator message on receiver side is as follows in a low load scenario(Wu, 2019, pp. 23–24):

1. Verify MAC1 Hash.

Once a receiver receives the initiator message, responder will verify the value of blake2s keyed MAC1 field by recalculating the hashing with its own stored public key. If the hash generated from its own key does not match what is received in the MAC1 field, then no response is returned per Wireguard operating principles.

2. Check Time Stamp Length, Verify Offered Peer Static Public Key

Else, decryption of the offered message is processed, and receiver will examine the length of the timestamp. Message will not be processed further if the timestamp is not exactly 12 bytes. It will then also look up the offered public key once it has been decrypted. If no public key of the sender found in the receivers database, the packet is then dropped.

3. Check Timestamp Value

Finally, the receiver will check if there were previous connections from the client. If the timestamp received in the initiator message is less than or equal to the decrypted timestamp, the packet is also dropped.

4. Prepare Handshake Response (Type 2)

If the public key corresponds to configured peer and offered timestamp is less than or equal to the timestamp stored on the receiver, then the receiver will prepare and send Handshake Response.

Packet 2: Initiator Response (Type 2)

The next type of packet is a Type 2 Packet.

Message Layout:

Type (1 Bytes)	Reserved (3 bytes)
Sender Index (4 Bytes)	
Receiver Index (4 Byte)	
Responder Ephemeral Public Key (32 Bytes)	
Encrypted Empty (16 Byte)	
mac1 (16 Bytes)	mac2 (16 Bytes)

Raw Packet:

```
0000   02 00 00 00 05 0d 14 1b 0f 51 09 7b c5 b3 f4 56
0010   62 12 98 61 34 40 0e f1 36 c9 d8 91 c6 f9 81 f3
0020   3e 7a 5c 1c 2d ca b0 ba c5 91 97 14 26 2c 82 cd
0030   fb c1 4f 33 6b fb 93 a7 32 73 e5 58 55 e3 00 6c
0040   92 7f e8 84 3f 6c a5 21 b5 35 58 91 00 00 00 00
0050   00 00 00 00 00 00 00 00 00 00 00 00
```

The first 8 Bytes of the message share the same layout as the imitator except for the value in type. Initiator value would be 2 rather than 1. However, following the sender index, the receiver places on its own 4-byte identifier index which is its own identifier. This is followed by the responder's ephemeral key, an empty 16 byte encrypted section, and the two MAC values. This time the MAC address will be the hash of the public key of the initiator.

The process for validating the handshake response is as follows(Wu, 2019, p. 25):

1. **Verify the MAC1 Hash**

   Much like the validation on the responder side, the initiator will check the MAC1 Hash against its own static key. If the hash is different, the message is dropped. In this case the hash matches.

2. **Checks state of the handshake matching the receiver index.**

   Connection is aborted if there is no state or if a handshake has been previously completed.

3. **Decrypt the empty section**

   This section is left empty on purpose. If the value is not empty, then the packet is dropped.

4. **Derive Transport keys and complete cryptographic handshake**

Packet 3: Data Message (Type 4)

The final packet part of normal operation is the data message. This is the message that typically contains data that is being transported. However, keep alive packets also are fall into the type 4 category.

Type (1 Bytes)	Reserved (3 bytes)
Reciever Index (4 Byte)	
Counter (8 Bytes)	
Data(≥16)	

Raw Message of a keep alive:

```
0000    04 00 00 00 05 0d 14 1b 00 00 00 00 00 00 00 00
0010    ba 38 92 fb 66 ff 9a 9f 4a 27 02 fa 38 33 db a2
```

The following is a raw packet of a non-keep-alive:

```
0000    04 00 00 00 05 0d 14 1b 01 00 00 00 00 00 00 00
0010    5d 6d e9 50 b4 01 36 8c ab 92 83 b1 42 92 e8 bb
0020    05 92 1d 7a f5 63 b6 90 f1 4b fd 09 a9 68 69 fb
0030    a7 58 6d 66 96 c5 1c 97 34 85 78 00 df 67 7e 6b
0040    66 33 ee 59 1c d9 30 2e 66 fc 88 bf 6f 70 c3 03
0050    ec 63 99 9d dd ca 1c b0 4c 89 bd a2 ac 6a 87 41
0060    70 63 2c 21 ec 5b 30 38 76 73 51 94 33 7b a4 c0
0070    e4 e0 10 ad df 91 6c 1e b1 fe cb 33 11 fe 06 67
```

In the case that the UDP packet does not hit the maximum MTU, the message is padded in order to be a multiple of 16 bytes(Donenfeld, n.d.). If the message is a keep alive, no padding is done and encrypted packet is 16 Bytes in Length, which is the length of Poly1305 authentication tag

that is used for authenticated encryption of encapsulated packets in the Wireguard tunnel. The data in the packet is null(Donenfeld, 2017, p. 15). At this point, there is no sender index, only the receiver index in following packets depending on where the packet originated from. In front of the data section of the message, an 8 Byte Counter field is included for the purposes of packet order since UDP is reliable like TCP. Note that the data packet in this exchange has a counter iterated by 1 versus the 0 in the keep alive. Counter is maintained based on the direction of the packet. Given the maximum MTU of normal sized frame is 1518 bytes, header overhead for IPv4 (20-bytes for IPv4 header, 8 Byte UDP header, 4-byte type, 4-byte receiver index, 8-Byte counter, and 16-byte authentication tag), maximum actual data would be 1458.

Reply Cookie Message (Type 3)

During testing, no cookie messages were observed. However, the following is the basic topology of the reply to cookie message.

Type (1 Bytes)	Reserved (3 bytes)
Reciever Index (4 Byte)	
Nounce (24 byte)	
Encrypted Cookie (32 Byte)	

The purpose of the Cookie Reply message is for handling handshakes during situations of high load where the encrypted cookie is local 256-bit secret value that is rotated every 2 minutes. Message size is 64 Bytes versus 98 bytes of a normal handshake response. By reducing the size of the handshake, Wireguard can avoid assisting in application attacks. It is derived from the original source socket of the original handshake. A separate Nounce is used from the regular counter for the purposes of preventing replay attacks. Once normal service is restored, the cookie's hash is used filled into the MAC2 field in further communications (Wu, 2019, pp. 26–27).

VyOS and Wireguard Configuring

Basic Setup network setup

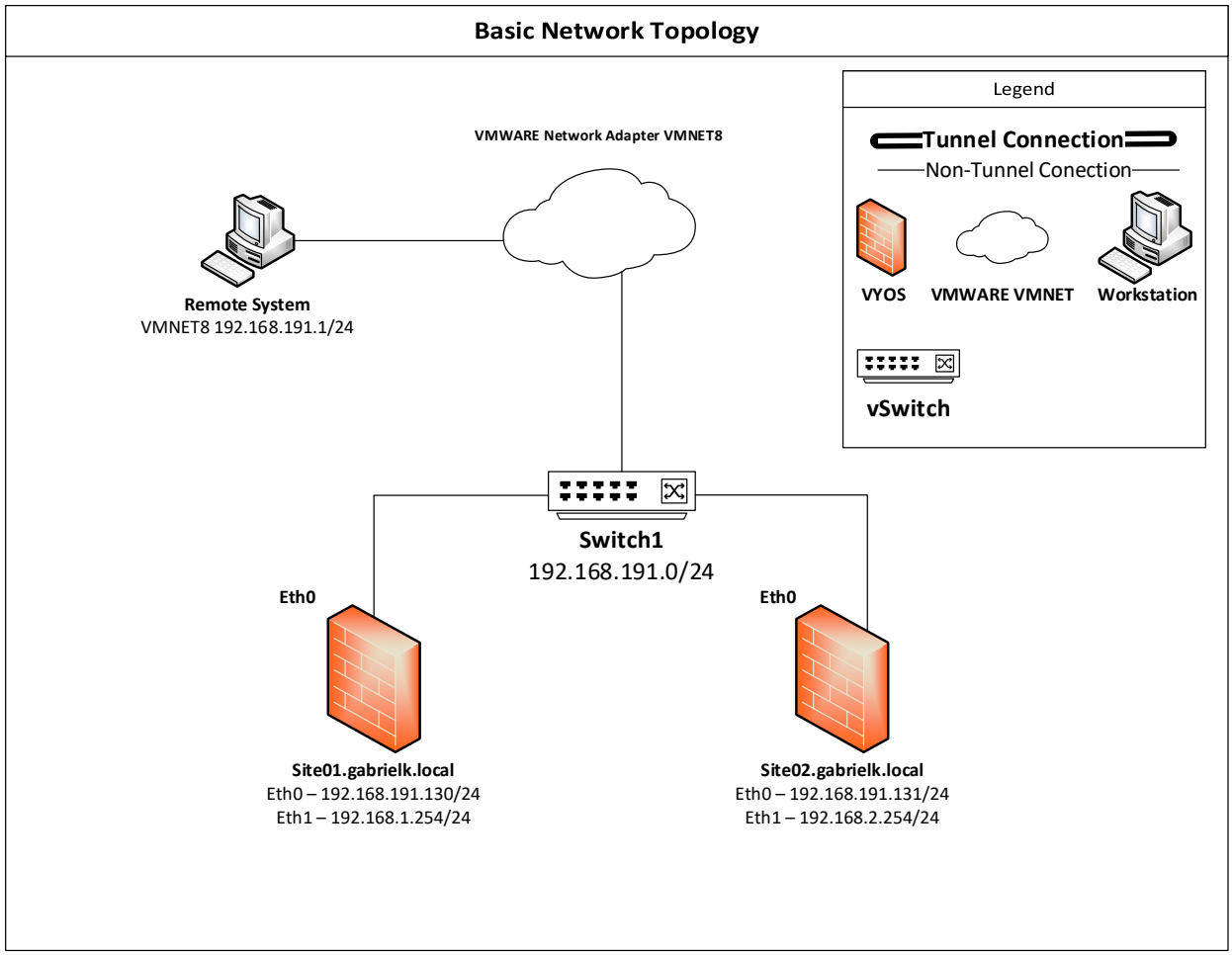


Figure 1 Logical Network Topology

All testing is done on GNS3. Each VyOS 1.3 router is given 512MB of 1 Virtual Processor and are running on a nested GNS3 VM running in VMware workstation 15x. Test remote client is a Windows 10 BareMetal device that is also running the VMware workstation. All traffic is on a NAT VMNET handled by VMware.

Site 1 Subnet is 192.168.1.0/24, and 192.168.2.0/24 is the Site 2 internal subnet. WAN subnet is 192.168.191.0/24.

Interface setup on site1 are as follows:

```
set interfaces ethernet eth0 address '192.168.191.130/24'
set interfaces ethernet eth0 description 'SITE1_WAN'
set interfaces ethernet eth0 firewall in name 'OUTSIDE-IN'
set interfaces ethernet eth0 firewall local name 'OUTSIDE-LOCAL'
set interfaces ethernet eth1 address '192.168.1.254/24'
set interfaces ethernet eth1 description 'SITE1_LOCAL'
```

Future firewall rules required by Wireguard, OpenVPN, and IPsec will be applied on the Outside-LOCAL ACL.

NAT on Site1:

```
set nat source rule 100 outbound-interface 'eth0'
set nat source rule 100 source address '192.168.1.0/24'
set nat source rule 100 translation address 'masquerade'
```

Masquerade used to enable PAT based on interface the rule is applied on.
The configuration is similar for Site 2 with its source being the 192.168.2.0/24 subnet.

Enabling SSH for remote access and file transfer:

```
set service ssh ciphers 'aes256-cbc'
set service ssh ciphers 'aes256-ctr'
set service ssh listen-address '192.168.191.130'
set service ssh port '22'
```

Basic Firewalls include setting default actino to drop, allowing established connections, SSH and ICMP:

1. Allow establish connections going out from local to outside subnets and setting default action to drop.

```
set firewall name OUTSIDE-IN default-action 'drop'
set firewall name OUTSIDE-IN rule 10 action 'accept'
set firewall name OUTSIDE-IN rule 10 state established 'enable'
set firewall name OUTSIDE-IN rule 10 state related 'enable'
```

2. Allowing established connections from outside to local subnet and setting default action to drop

```
set firewall name OUTSIDE-LOCAL default-action 'drop'
set firewall name OUTSIDE-LOCAL rule 10 action 'accept'
set firewall name OUTSIDE-LOCAL rule 10 state established 'enable'
set firewall name OUTSIDE-LOCAL rule 10 state related 'enable'
3. Allowing ICMP echo requests on WAN interface

set firewall name OUTSIDE-LOCAL rule 20 action 'accept'
set firewall name OUTSIDE-LOCAL rule 20 icmp type-name 'echo-request'
set firewall name OUTSIDE-LOCAL rule 20 protocol 'icmp'
set firewall name OUTSIDE-LOCAL rule 20 state new 'enable'
```

4. Allowing SSH and blocking spam requests

```
set firewall name OUTSIDE-LOCAL rule 22 action 'drop'
set firewall name OUTSIDE-LOCAL rule 22 destination port '22'
set firewall name OUTSIDE-LOCAL rule 22 protocol 'tcp'
set firewall name OUTSIDE-LOCAL rule 22 recent count '4'
set firewall name OUTSIDE-LOCAL rule 22 recent time '60'
set firewall name OUTSIDE-LOCAL rule 22 state new 'enable'
```

```
set firewall name OUTSIDE-LOCAL rule 23 action 'accept'
set firewall name OUTSIDE-LOCAL rule 23 destination port '22'
set firewall name OUTSIDE-LOCAL rule 23 protocol 'tcp'
set firewall name OUTSIDE-LOCAL rule 23 state new 'enable'
```

5. Apply firewall rules to interface

```
set interfaces ethernet eth0 firewall in name 'OUTSIDE-IN'
set interfaces ethernet eth0 firewall local name 'OUTSIDE-LOCAL'
```

Site-to-Site Wireguard

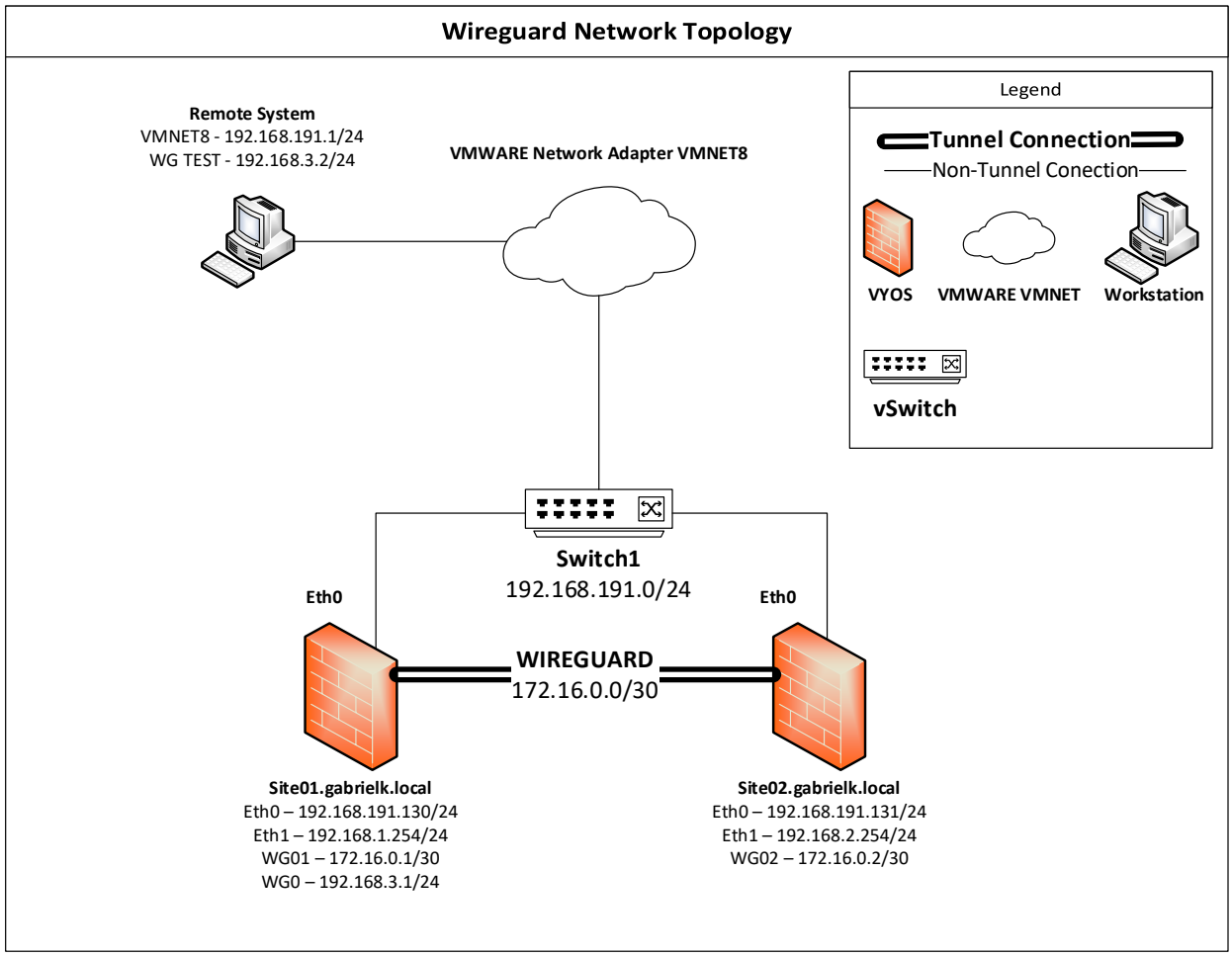


Figure 2 Wireguard Topology Site-to-Site

Configuration of site-to-site Wireguard is shown in the figure above. It consists of two parts for Wireguard on VyOS and was completed through following the official guide from VyOS for version 1.3(*Wireguard*, n.d.). The sections are:

1. Generating keys
2. Wireguard tunnel interface.

Generating Key Pair

To generate a key via the `generate Wireguard default-keypair.`
Once this has been generated, to show the default keypair generated use `show wireguard keypairs pubkey default:`

```
vyos@site01:~$ show wireguard keypairs pubkey default
4e4L8AgoUdANZTrzBr5K906CsUJZBwpZwdz1QQ6QT3c=
```

Wireguard tunnel interface

To create a Wireguard interface, you will need the following parameters

1. **Address of the tunnel interface**
2. **Port to listen on**
3. **The public peer addresses**
4. **allowed subnets on the tunnel**
5. **The public key of the peer**
6. **Firewall Rules**
7. **Routes to remote subnets**

The following configuration is done from the perspective of site2

1. **Set the tunnel address and optional description**

```
set interfaces wireguard wg02 address '172.16.0.2/30'
set interfaces wireguard wg02 description 'VPN-to-wg01'
```

2. **Set allowed IP address on the tunnel. If this is set to `0.0.0.0/0` it will cause the device to allow all traffic through the tunnel.**

```
set interface wireguard wg02 peer to-wg01 allowed-ips 192.168.0.0/16
set interface wireguard wg02 peer to-wg01 allowed-ips 172.16.0.0/30
```

3. **Provide the peers public IP address and wireguard port**

```
set interfaces wireguard wg02 peer to-wg01 address '192.168.191.130'
set interfaces wireguard wg02 peer to-wg01 port '51820'
```

4. **Set the public key of the peer**

```
set interfaces wireguard wg02 peer to-wg01 pubkey ‘4e4L8…’
```

5. **Port the device will listen on**

```
set interfaces wireguard wg02 port '51820'
```

6. **Configuring routing for tunnel. In order to reduce the number of variables that could affect performance testing later, static routes will be use in this example.**

```
set protocols static route 192.168.1.0/24 next-hop 172.16.0.1
```

7. Configure Firewall， Minimal firewall configuration is required. Allowing traffic on UDP 51820 is enough

```
set firewall name OUTSIDE_LOCAL rule 30 action accept
set firewall name OUTSIDE_LOCAL rule 30 description WireGuard_IN
set firewall name OUTSIDE_LOCAL rule 30 destination port 51820
set firewall name OUTSIDE_LOCAL rule 30 log enable
set firewall name OUTSIDE_LOCAL rule 30 protocol udp
set firewall name OUTSIDE_LOCAL rule 30 source
```

8. Add static route to route traffic from local subnet to site 1 traffic via tunnel

```
set protocols static route 192.168.1.0/24 next-hop 172.16.0.1
```

Configuration of the peer would be match that of above, swapping the peer ip and different tunnel address. Once the configuration is set. Wireguard should be functional. You can verify if traffic is passing through the tunnel via `show interfaces wireguard wg01` command mode:

```
vyos@site02:~$ show interfaces wireguard wg02
wg02: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1420 qdisc noqueue state UNKNOWN group default qlen 1000
    link/none
    inet 172.16.0.2/30 brd 172.16.0.3 scope global wg02
       valid_lft forever preferred_lft forever
    inet6 fe80::f552:62ff:feb8:8c27/64 scope link
       valid_lft forever preferred_lft forever
    Description: VPN-to-wg01

    RX:  bytes  packets  errors  dropped  overrun       mcast
          8680       69       0        0        0           0
    TX:  bytes  packets  errors  dropped  carrier  collisions
          8536       68       0        0        0           0
```

Client-to-Site Wireguard

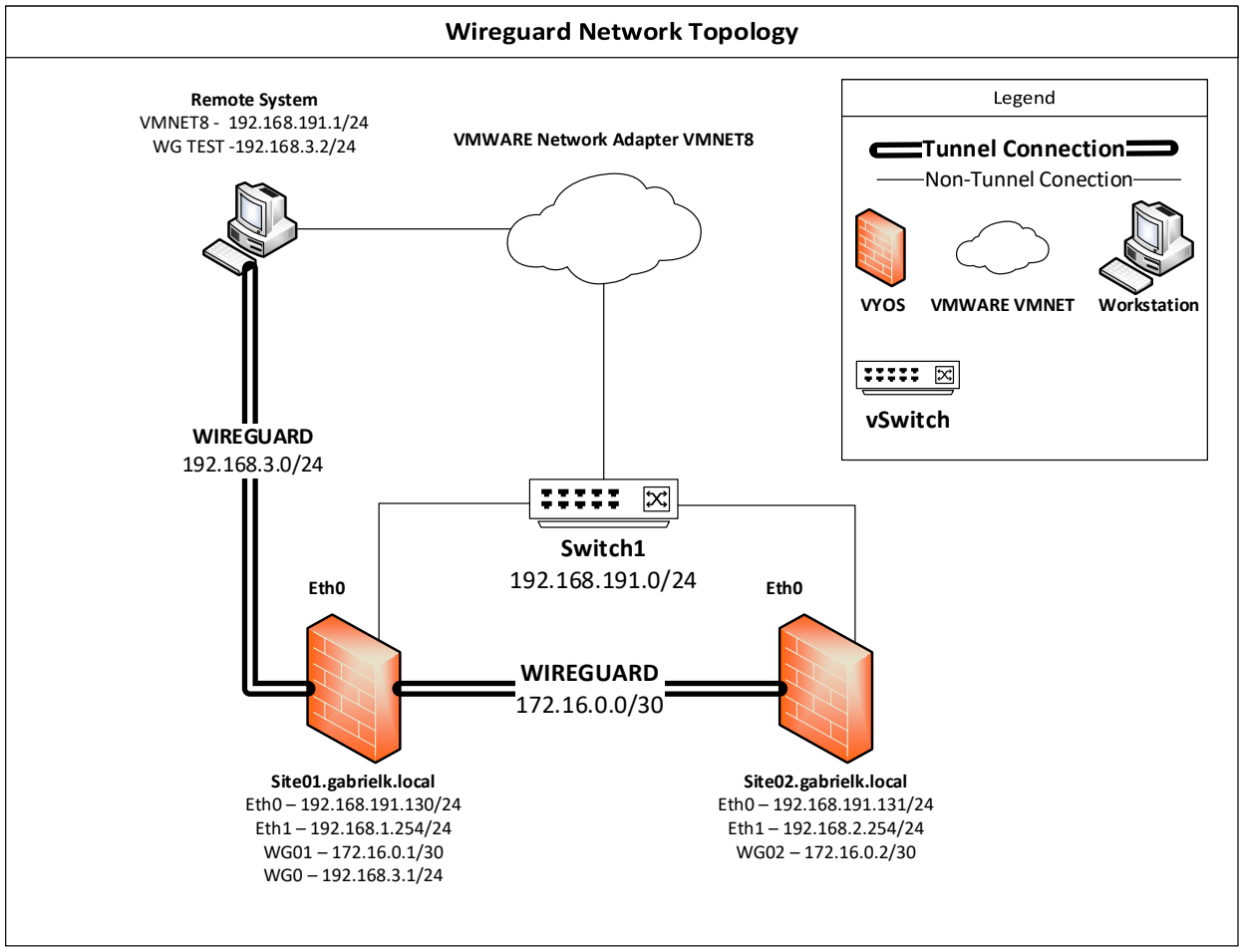


Figure 3 Wireguard Topology with Remote Client

Wireguard configuration is quite like the site-to-site on VyOS. Key difference being no public IP of the remote clients on the server side. This is because the client's IP address is not guaranteed.

Server-Side Configuration

Set up of remote roaming clients requires less configuration when compared to site-to-site configuration. It requires the following:

1. **Interface IP address**
2. **Port**
3. **Allowed-IPS**
4. **Public Key**
5. **Routes**

Public key will need to be generated by the client. Windows client and MacOS client will auto-generate key pair upon install. Linux client will require manual generation. If no keys have been generated by Vyos previously, a key pair will also need to be generated to provide the client with a pubkey. Again, verify the keypair on VYOS and provide this to the client via `show wireguard keypairs pubkey default.` The following is configured on site1:

1. Set Interface IP address

```
set interfaces wireguard wg0 address 192.168.3.1/24
   2. Set port that peer ‘test’ will listen on
set interfaces wireguard wg0 peer test port '51821'
   3. Set the allowed IP addresses from this client
set interfaces wireguard wg0 peer test allowed-ips 192.168.3.2/32
   4. Set the public key of the peer
set interfaces wireguard wg0 peer test pubkey 4e4L…
```

Firewall rules need to be enabled as well as shown below:

```
set firewall name OUTSIDE-LOCAL rule 40 action accept
set firewall name OUTSIDE-LOCAL rule 40 description RoadWarior_IN
set firewall name OUTSIDE-LOCAL rule 40 destination port 51821
set firewall name OUTSIDE-LOCAL rule 40 log enable
set firewall name OUTSIDE-LOCAL rule 40 protocol udp
set firewall name OUTSIDE-LOCAL rule 40 source
```

5. Add necessary routes

Be sure to add routes on site two so traffic on local subnet of site 2 knows how to reach the remote clients:

```
SITE02:
set protocols static route 192.168.3.0/24 next-hop 172.16.0.1
```

No additional routes are need on site01 in our topology.

Road Warrior Client Configuration

The following are instructions for client configuration on Windows. MacOS will have a similar setup. If on Ubuntu 20.04 LTS, Wireguard configuration has not yet been fully integrated with Network Manager. Configuration is done in CLI in that case.

On the client side, open the Wireguard application

1. **In the bottom left, select the Add Tunnel Button > Add Empty Tunnel…**
2. **In the "Edit Tunnel" Window, provide a name for the Name Textbox**
3. **Copy the "Public Key" textbox. This is the value that is required in the peer configuration.**
4. **In the textbox underneath, it should show the private key under [Interface]**
5. **Under said [Interface] add the following:**

```
Address = 192.168.3.2/24
DNS = 8.8.8.8
```

6. **Create a new header [Peer] and add the following underneath**

```
PublicKey = 5G5s…
Endpoint = 192.168.191.130:51821
AllowedIPs = 192.168.1.0/24, 192.168.2.0/24, 192.168.3.0/24
```

See the following screenshot for reference.

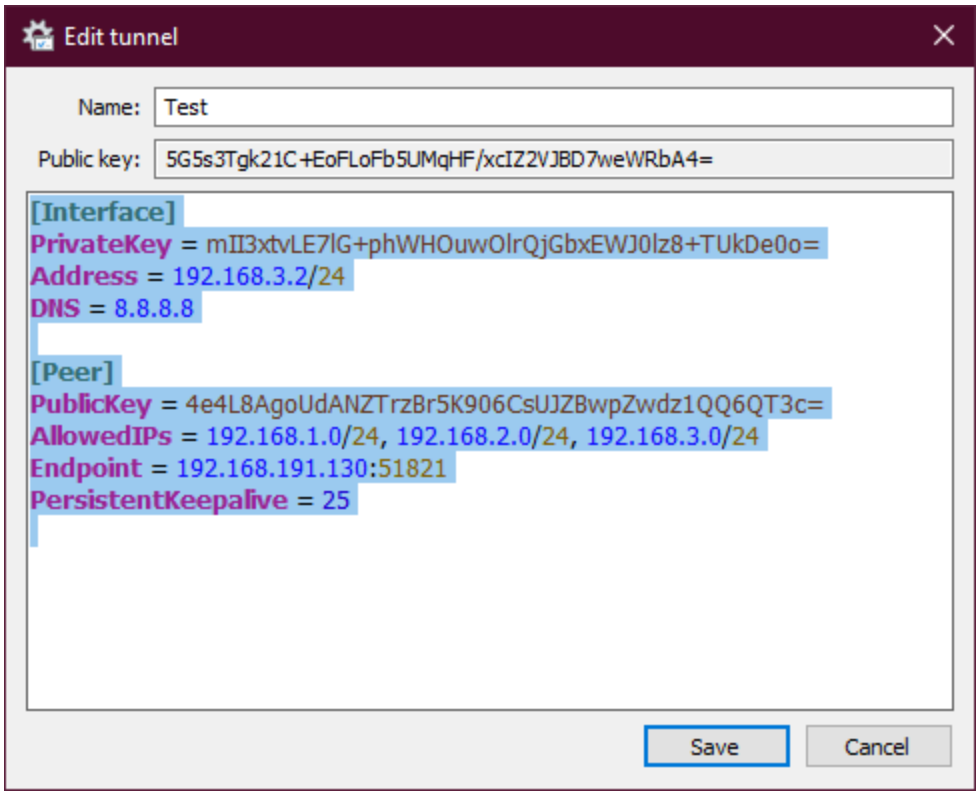


Figure 4 Creating Tunnel on Windows Client

Address IP needs to match the address in allowed in the allowed-ip configuration of the peer on the VyOS configuration. Furthermore, the AllowedIPs under will affect how traffic is routed. If the value is set to 0.0.0.0/0, all traffic will be routed through the tunnel. By specifying subnets, effectively enables split tunneling; only traffic that are destined for those locations will be routed through the tunnel. In example configuration, only traffic destined for 192.168.1.0/24, 192.168.2.0/24 and 192.168.3.0/24 will be allowed through the tunnel.

7. **Click save to save the tunnel configuration**
8. **Select the tunnel in the main Wireguard window and click activate under the interface pane.**
9. **Once activated you should see handshakes and transfer received and sent should increase**

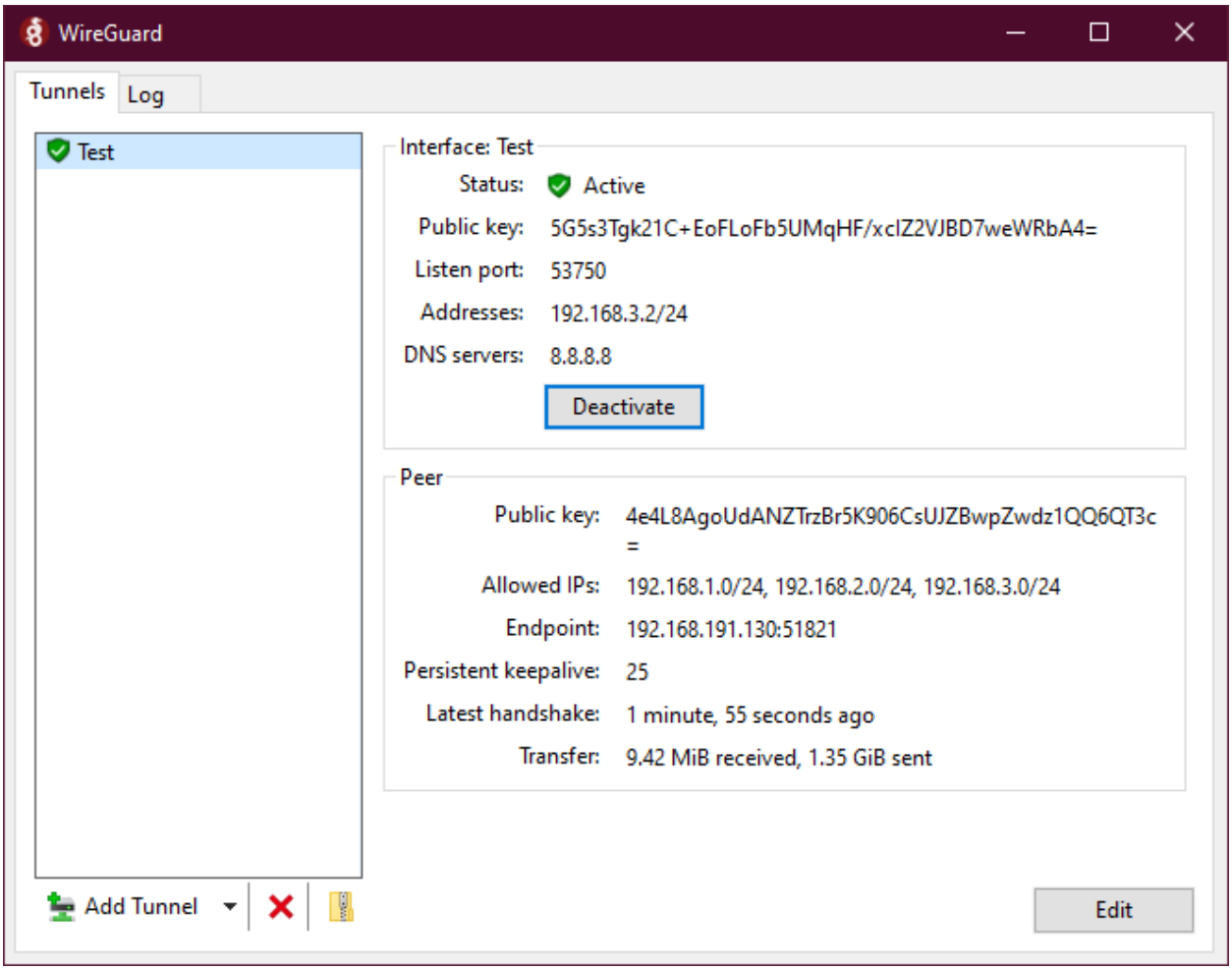


Figure 5 Established Wireguard Connection after some Iperf3 tests

10. You can also verify by checking on the VyOS side via show interfaces wireguard wg0:

```
vyos@site01:~$ show interfaces wireguard wg0
wg0: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 1420 qdisc noqueue state UNKNOWN group default qlen 1000
    link/none
    inet 192.168.3.1/24 brd 192.168.3.255 scope global wg0
       valid_lft forever preferred_lft forever
    inet6 fe80::fc78:44ff:fe33:f5ae/64 scope link
       valid_lft forever preferred_lft forever

    RX:      bytes  packets  errors  dropped  overrun       mcast
         218139572   150375       0        0        0           0
    TX:      bytes  packets  errors  dropped  carrier  collisions
            840252    10077       0        7        0           0
```

Configuration of OpenVPN

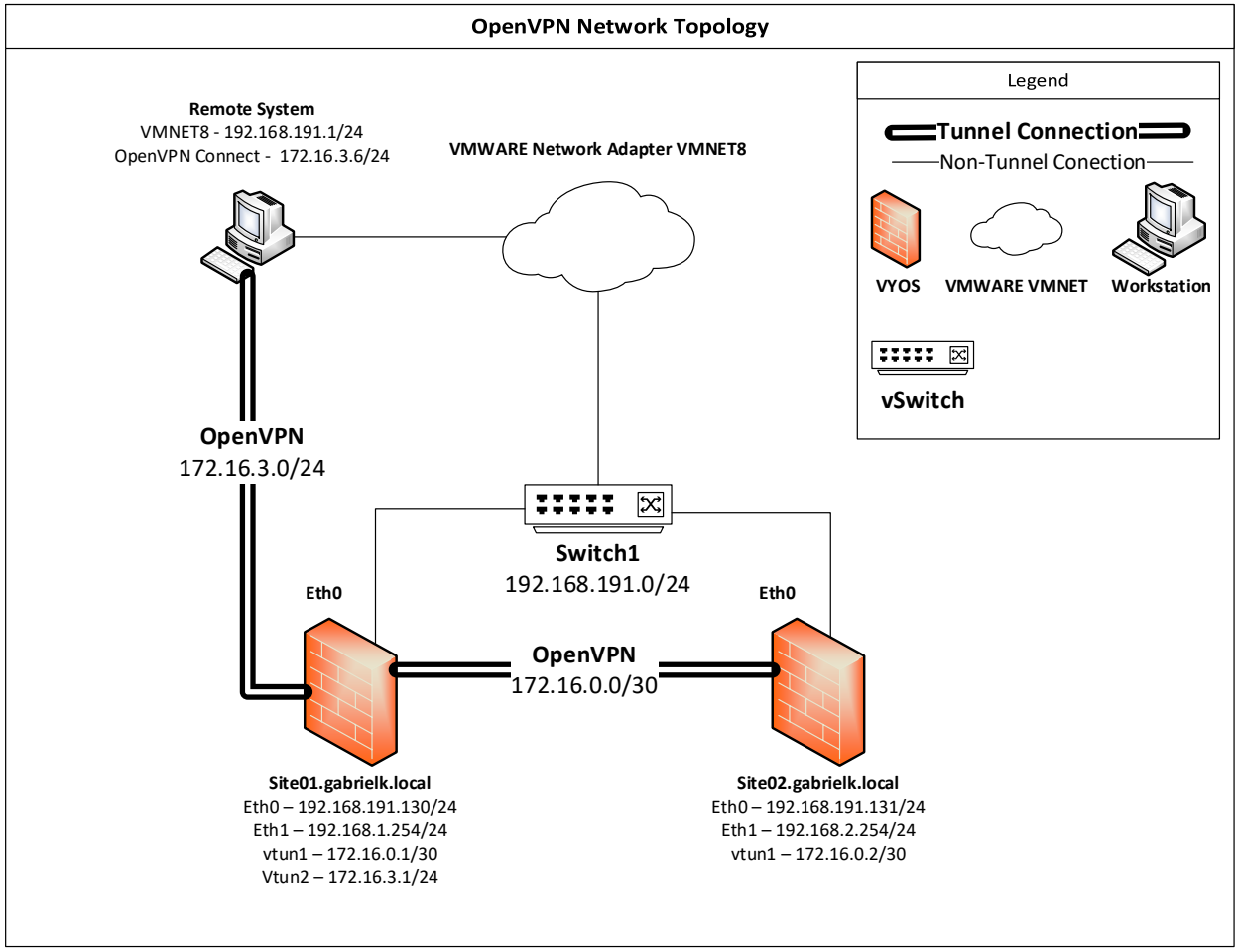


Figure 6 OpenVPN Topology

The following section will briefly describe the setup for OpenVPN for the purposes of this project as shown in Figure 6 OpenVPN Topology. Guidance was taken from VyOS user guide (*OpenVPN — VyOS 1.3.x (Equuleus) Documentation*, n.d.) unless otherwise stated. Sections will consist of the following:

1. Site-to-Site Configuration
2. Client to Site - Server-Side Configuration
3. Client to Site - Client-Side Configuration

Site-to-Site Configuration

Site to site is straightforward. You will need to specify the network settings, security settings, shared key and firewall settings:

For the network settings, a local address, local port, mode, transport protocol, remote-address, remote-host and remote-port.

For this project, the interface configuration would be as follows for site 1

```
set interfaces openvpn vtun1 local-address 172.16.0.2
set interfaces openvpn vtun1 local-port '1195'
set interfaces openvpn vtun1 mode 'site-to-site'
set interfaces openvpn vtun1 persistent-tunnel
set interfaces openvpn vtun1 protocol 'udp'
set interfaces openvpn vtun1 remote-address '172.16.0.1'
set interfaces openvpn vtun1 remote-host '192.168.191.130'
set interfaces openvpn vtun1 remote-port '1195'
```

For the security related settings, we will be using ae256 for encryption, sha256 for hashing as well as a generated secret key:

```
generate openvpn key openvpn_key01
set interfaces openvpn vtun1 encryption cipher 'aes256'
set interfaces openvpn vtun1 hash 'sha256'
set interfaces openvpn vtun1 shared-secret-key-file
'/config/auth/openvpn_key01'
```

For the tunnel to function, port 1995 on UDP must be opened:

```
set firewall name OUTSIDE-LOCAL rule 40 action 'accept'
set firewall name OUTSIDE-LOCAL rule 40 description 'OpenVPN_IN'
set firewall name OUTSIDE-LOCAL rule 40 destination port '1195'
set firewall name OUTSIDE-LOCAL rule 40 log 'enable'
set firewall name OUTSIDE-LOCAL rule 40 protocol 'udp'
set firewall name OUTSIDE-LOCAL rule 40 source
```

Route configuration is same as Wireguard configuration:

```
set protocols static route 192.168.2.0/24 next-hop 172.16.0.2
```

For the other site, the configuration should be the same. With the opposite settings for local-address, remote-address, remote-host, and remote-port. Route should be for 192.168.1.0/24 subnet with destination of 172.16.0.1 instead. The generated OpenVPN key should also be transferred to the other router. You can verify if the tunnel is up via:

vyos@site01:/config/auth/openvpn$ show openvpn site-to-site

```
OpenVPN status on vtun1

Client CN       Remote Host           Local Host           TX bytes    RX bytes   Connected Since
---------       -----------           ----------           --------    --------   ---------------

None (PSK)      192.168.191.131:1195  N/A                  5.3 KB      6.4 KB     N/A
```

Client to Site Configuration – Server Side

Per the guide, Configuration of OpenVPN for Client Server Connections requires certificates and creation of an OpenVPN interface. To create the certificates, per the VyOS user guide, Easy-RSA PKI will be used.

Configuring Easy-RSA PKI and Certificate Creation for Client-Server connections

1. **Copy the easy-rsa files to config and modify the vars configuration with relevant information:**

```
vyos@site01# cp -r /usr/share/easy-rsa/ /config/my-easy-rsa-config
vyos@site01# cd /config/my-easy-rsa-config
vyos@site01# mv vars.example vars
vyos@site01# vi vars
```

The following is a diff of the changes made:

```
vyos@site01:/config/my-easy-rsa-config$ diff -u /usr/share/easy-
rsa/vars.example
vars
--- /usr/share/easy-rsa/vars.example    2019-02-08 14:53:24.000000000 +0000
+++ vars        2021-10-28 00:58:16.120000000 +0000
@@ -80,7 +80,7 @@
 #   cn_only  - use just a CN value
 #   org      - use the "traditional" Country/Province/City/Org/OU/email/CN
form
at

-#set_var EASYRSA_DN    "cn_only"
+set_var EASYRSA_DN     "org"

 # Organizational fields (used with 'org' mode and ignored in 'cn_only' mode.)
 # These are the default values for fields which will be placed in the
```

```
@@ -88,12 +88,12 @@
 # you may omit any specific field by typing the "." symbol (not valid for
 # email.)

-#set_var EASYRSA_REQ_COUNTRY   "US"
-#set_var EASYRSA_REQ_PROVINCE  "California"
-#set_var EASYRSA_REQ_CITY      "San Francisco"
-#set_var EASYRSA_REQ_ORG       "Copyleft Certificate Co"
-#set_var EASYRSA_REQ_EMAIL     "me@example.net"
-#set_var EASYRSA_REQ_OU                "My Organizational Unit"
+set_var EASYRSA_REQ_COUNTRY    "CA"
+set_var EASYRSA_REQ_PROVINCE   "British Columbia"
+set_var EASYRSA_REQ_CITY       "Vancouver"
+set_var EASYRSA_REQ_ORG        "Gabrielk.ca"
+set_var EASYRSA_REQ_EMAIL      "me@gabrielk.ca"
+set_var EASYRSA_REQ_OU         "Vancouver-Office"

 # Choose a size in bits for your keypairs. The recommended value is 2048.
Usin
g
 # 2048-bit keys is considered more than sufficient for many years into the
@@ -101,7 +101,7 @@
 # generation take much longer. Values up to 4096 should be accepted by most
 # software. Only used when the crypto alg is rsa (see below.)

-#set_var EASYRSA_KEY_SIZE      2048
+set_var EASYRSA_KEY_SIZE       2048

 # The default crypto mode is rsa; ec can enable elliptic curve support.
 # Note that not all software supports ECC, so use care when enabling it.
```

2. Initialize the PKI based on vars

```
vyos@site01# ./easyrsa init-pki

Note: using Easy-RSA configuration from: ./vars

init-pki complete; you may now create a CA or requests.
Your newly created PKI dir is: /config/my-easy-rsa-config/pki
```

3. Build Certificate Authority. Keep password safe as it is required when generating and signing certificates later.

```
vyos@site01# ./easyrsa build-ca

Note: using Easy-RSA configuration from: ./vars

Using SSL: openssl OpenSSL 1.1.1d  10 Sep 2019

Enter New CA Key Passphrase:
Re-Enter New CA Key Passphrase:
Generating RSA private key, 2048 bit long modulus (2 primes)
.....................................................................................
........+++++
..+++++
e is 65537 (0x010001)
Can't load /config/my-easy-rsa-config/pki/.rnd into RNG
140156721591424:error:2406F079:random number generator:RAND_load_file:Cannot open
file:../crypto/rand/randfile.c:98:Filename=/config/my-easy-rsa-config/pki/.rnd
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [CA]:CA
```

```
State or Province Name (full name) [British Columbia]:
Locality Name (eg, city) [Vancouver]:
Organization Name (eg, company) [Gabrielk.ca]:
Organizational Unit Name (eg, section) [Vancouver-Office]:
Common Name (eg: your user, host, or server name) [Easy-RSA CA]:site01
Email Address [me@gabrielk.ca]:

CA creation complete and you may now import and sign cert requests.
Your new CA certificate file for publishing is at:
/config/my-easy-rsa-config/pki/ca.crt

vyos@site01:/config/my-easy-rsa-config$ ./easyrsa gen-req van-office nopass

Note: using Easy-RSA configuration from: ./vars

Using SSL: openssl OpenSSL 1.1.1d  10 Sep 2019
Generating a RSA private key
.......+++++
.............................+++++
writing new private key to '/config/my-easy-rsa-config/pki/private/van-
office.key.qVXKouUwBY'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [CA]:
State or Province Name (full name) [British Columbia]:
Locality Name (eg, city) [Vancouver]:
Organization Name (eg, company) [Gabrielk.ca]:
Organizational Unit Name (eg, section) [Vancouver-Office]:
Common Name (eg: your user, host, or server name) [van-office]:
Email Address [me@gabrielk.ca]:
```

```
Keypair and certificate request completed. Your files are:
req: /config/my-easy-rsa-config/pki/reqs/van-office.req
key: /config/my-easy-rsa-config/pki/private/van-office.key
```

4. Sign Site01 Certificate

```
vyos@site01:/config/my-easy-rsa-config$ ./easyrsa sign-req server van-office

Note: using Easy-RSA configuration from: ./vars

Using SSL: openssl OpenSSL 1.1.1d  10 Sep 2019


You are about to sign the following certificate.
Please check over the details shown below for accuracy. Note that this request
has not been cryptographically verified. Please be sure it came from a trusted
source or that you have verified the request checksum with the sender.

Request subject, to be signed as a server certificate for 1080 days:

subject=
    countryName               = CA
    stateOrProvinceName       = British Columbia
    localityName              = Vancouver
    organizationName          = Gabrielk.ca
    organizationalUnitName    = Vancouver-Office
    commonName                = van-office
    emailAddress              = me@gabrielk.ca


Type the word 'yes' to continue, or any other input to abort.
  Confirm request details: yes
Using configuration from /config/my-easy-rsa-config/pki/safessl-easyrsa.cnf
Enter pass phrase for /config/my-easy-rsa-config/pki/private/ca.key:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
```

```
countryName           :PRINTABLE:'CA'
stateOrProvinceName   :ASN.1 12:'British Columbia'
localityName          :ASN.1 12:'Vancouver'
organizationName      :ASN.1 12:'Gabrielk.ca'
organizationalUnitName:ASN.1 12:'Vancouver-Office'
commonName            :ASN.1 12:'van-office'
emailAddress          :IA5STRING:'me@gabrielk.ca'
Certificate is to be certified until Oct 14 03:01:37 2024 GMT (1080 days)

Write out database with 1 new entries
Data Base Updated

Certificate created at: /config/my-easy-rsa-config/pki/issued/van-office.crt
```

5. Generate Certificate for Client.

```
vyos@site01# ./easyrsa build-client-full client1 nopass

Note: using Easy-RSA configuration from: ./vars

Using SSL: openssl OpenSSL 1.1.1d  10 Sep 2019
Generating a RSA private key
.........+++++
.........................+++++
writing       new       private       key       to       '/config/my-easy-rsa-
config/pki/private/client1.key.Gc2NmL36hF'
-----
Using configuration from /config/my-easy-rsa-config/pki/safessl-easyrsa.cnf
Enter pass phrase for /config/my-easy-rsa-config/pki/private/ca.key:
Check that the request matches the signature
Signature ok
The Subject's Distinguished Name is as follows
countryName           :PRINTABLE:'CA'
stateOrProvinceName   :ASN.1 12:'British Columbia'
localityName          :ASN.1 12:'Vancouver'
organizationName      :ASN.1 12:'Gabrielk.ca'
organizationalUnitName:ASN.1 12:'Vancouver-Office'
```

```
commonName            :ASN.1 12:'client1'
emailAddress          :IA5STRING:'me@gabrielk.ca'
Certificate is to be certified until Oct 13 02:35:58 2024 GMT (1080 days)

Write out database with 1 new entries
Data Base Updated
```

6. Generate Diffie-Hellman parameters

```
vyos@site01:/config/my-easy-rsa-config$ ./easyrsa gen-dh

Note: using Easy-RSA configuration from: ./vars

Using SSL: openssl OpenSSL 1.1.1d  10 Sep 2019
Generating DH parameters, 2048 bit long safe prime, generator 2
This is going to take a long time
.........................+.....................................................
.....................................................................................
....+.........+.....................................................................
.....................................................................................
.....+.+.............................................................................
...+.......................+.....................................................
.............................+.....................................................
.....................................................++*++*++*++*
```

```
DH parameters of size 2048 created at /config/my-easy-rsa-config/pki/dh.pem
```

7. Transfer the client certificates and CA certificates to the workstation via sFTP client such as WinSCP or Filezilla

8. Add Routes on Site02

```
set protocols static route 172.16.3.0/24 next-hop 172.16.0.1
```

Client-Server Interface Configuration

Setup for the interface for site-to-site on Vyos comprises network layer settings, Security layer settings, and Firewall rules. Networking layer settings include port, transport protocol, OpenVPN mode, routes and subnet remote devices will belong to. The security settings include specifying the certificates that will be used for authenticating and validating clients. This are the certificates that were generated in the previous section. Finally, firewall rules must be enabled for the traffic to be permitted on the outside interface.

For testing, port will be listening on UDP 1190, persistent-tunnel and in server mode. Routes pushed would be for 192.168.0.0/22 which would include the internal subnets for both site 1 and site 2. This can be done with the following commands:

```
set interfaces openvpn vtun2 local-port '1190'
set interfaces openvpn vtun2 mode 'server'
set interfaces openvpn vtun2 persistent-tunnel
set interfaces openvpn vtun2 protocol 'udp'
set interfaces openvpn vtun2 server push-route 192.168.0.0/22
set interfaces openvpn vtun2 server subnet '172.16.3.0/24'
```

For this setup, a CA cert, site01's own cert & key, a certificate revocation list, and Diffie-Helman certificate must be specified:

```
set interfaces openvpn vtun2 tls ca-cert-file '/config/auth/openvpn/ca.crt'
set interfaces openvpn vtun2 tls cert-file '/config/auth/openvpn/van-office.crt'
set interfaces openvpn vtun2 tls crl-file '/config/auth/openvpn/crl.pem'
set interfaces openvpn vtun2 tls dh-file '/config/auth/openvpn/dh.pem'
set interfaces openvpn vtun2 tls key-file '/config/auth/openvpn/van-office.key'
```

For the firewall rules, UDP 1190 must be allowed to connect to the Site01. See the following commands to do so.

```
set firewall name OUTSIDE-LOCAL rule 50 action 'accept'
set firewall name OUTSIDE-LOCAL rule 50 description 'OpenVPN_CLIENT_IN'
set firewall name OUTSIDE-LOCAL rule 50 destination port '1190'
set firewall name OUTSIDE-LOCAL rule 50 log 'enable'
set firewall name OUTSIDE-LOCAL rule 50 protocol 'udp'
set firewall name OUTSIDE-LOCAL rule 50 source
```

Client to Site Configuration – Client Side

For this project, the official Open VPN Connect client will be used. For the OpenVPN client to connect to the server, it needs to have the certificates that were previously exported as well as create a new OPVPN configuration file. This can be created using a text editor with guidance from the OpenVPN Vyos guide and from Blog post from Brezular(*OpenVPN — VyOS 1.3.x (Equuleus) Documentation*, n.d.; *OpenVPN Remote Access VPNs Using TLS on VyOS | Brezular's Blog*, n.d.). The following are the steps used to implement this.

1. First Create the OPVPN file using notepad or any text editor. It will need to declare the following variables in said file:

    a. Specify the path CRT file generated from Easy-RSA on Site01
    b. Specify the path of client key file that was generated from Easy-RSA on Site01.
    c. Specify the path of the ca.crt previously generated
    d. Declare the public ip address and the port that it will be listening on for OpenVPN connections.

    In this example:

```
client
dev tun2
cert "C:/\Users/\gk/\Nextcloud/\School/\NAIT PROJECT 2/\client1.crt"
key "C:/\Users/\gk/\Nextcloud\School/\NAIT PROJECT 2/\client1.key"
ca "C:/\Users/\gk/\Nextcloud/\School/\NAIT PROJECT 2/\ca.crt"
remote 192.168.191.130 1190
```

2. Open the OpenVPN Connect client and select the hamburger menu on the right

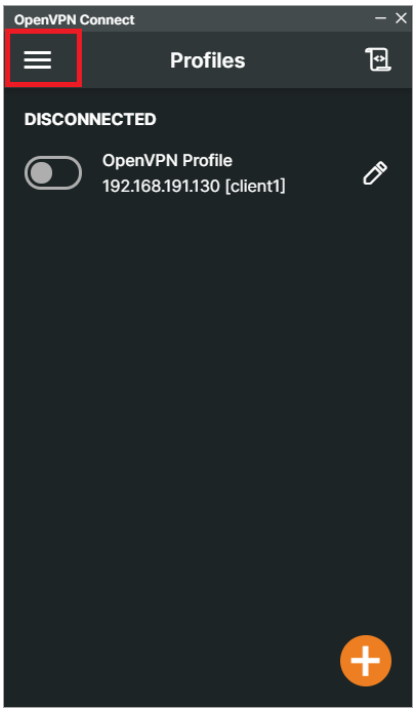


Figure 7 OpenVPN Connect - Hamburger Menu

3. Select import profile

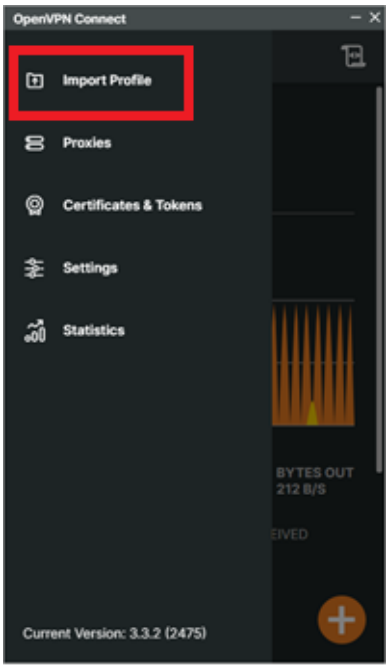


Figure 8 OpenVPN Connect - Select Import Profile

4. Select file tab in the top of the import pane and select browse

Figure 9 OpenVPN Connect - Select File Then Browser

5. Then specify the OPNVPN file that was previous created when prompted

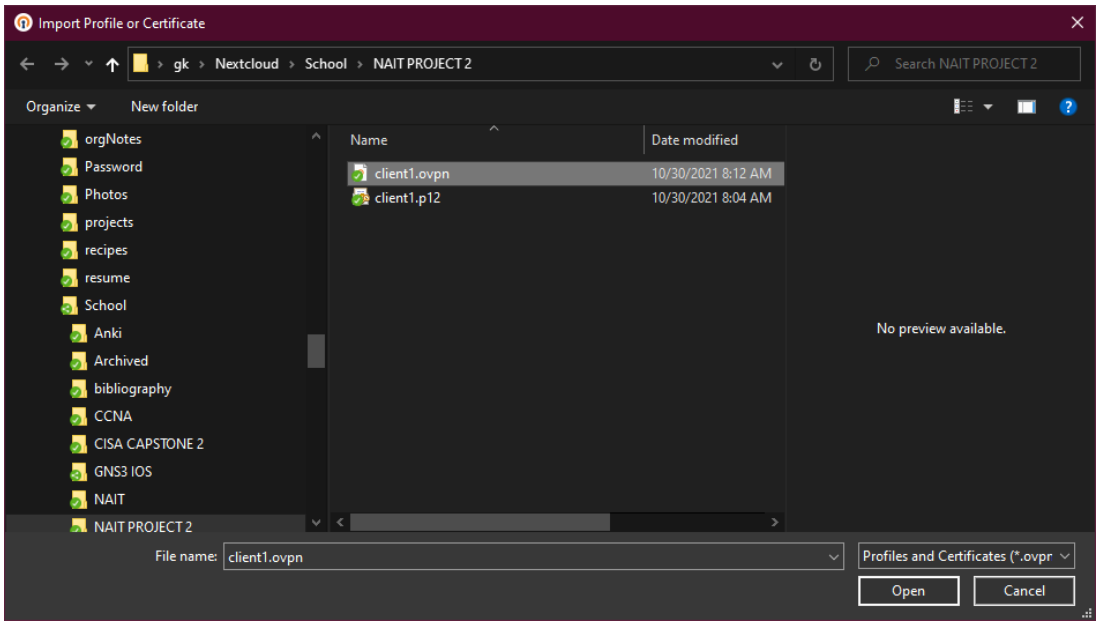


Figure 10 OpenVPN Connect - Selecting. OPVN File

6. The new profile should now show in the main pane as disconnected as seen below. To active, activate the toggle next the profile.

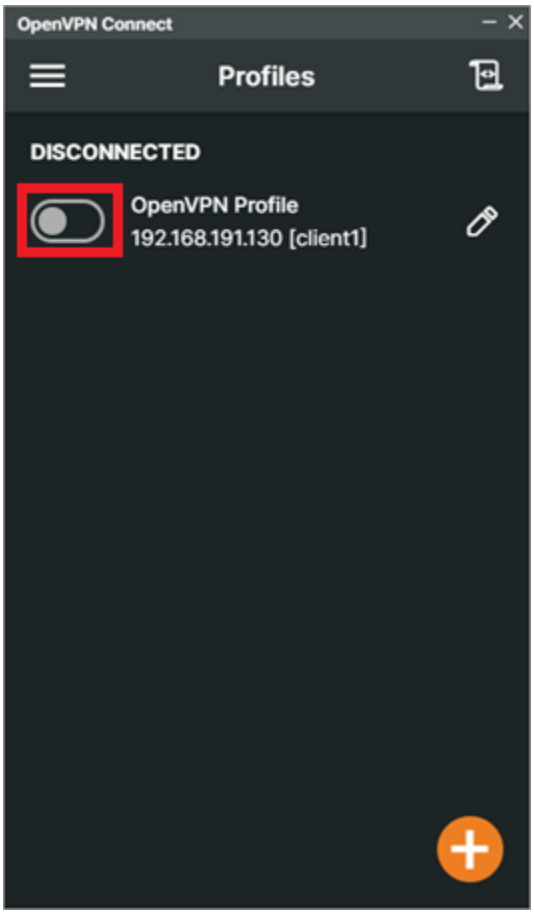


Figure 11 OpenVPN Connect - Select the OpenVPN Profile Created

7. If connection is successful, you should be presented with the following.

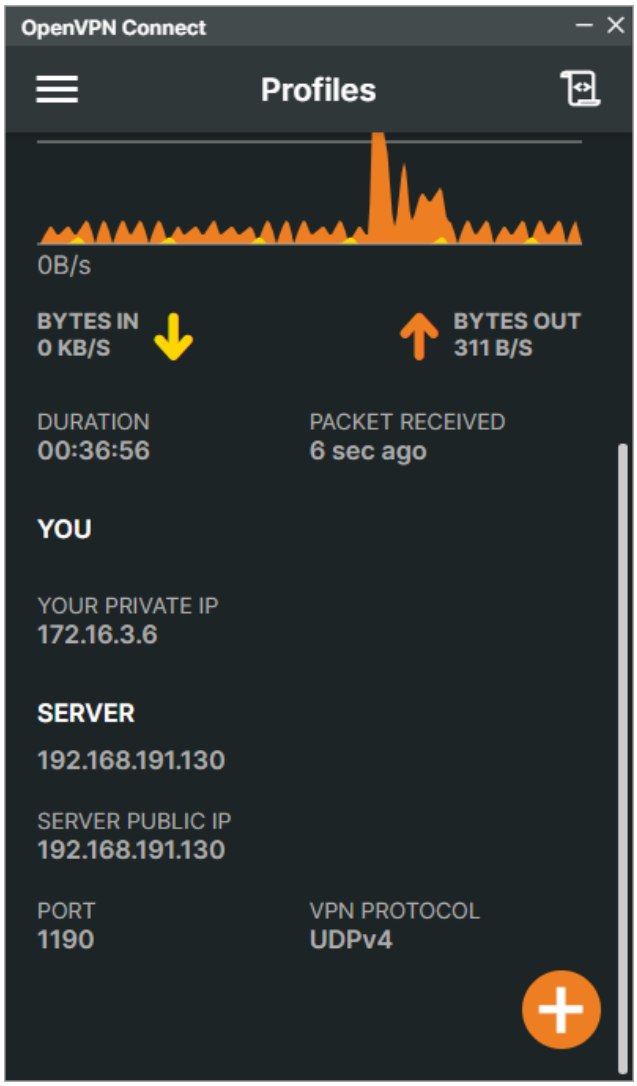


Figure 12 OpenVPN Successful Connection

8. Based on our configuration you should also see that the route being pushed out via route-print. You can also verify if the client is connected using `show openvpn server` in the operational mode. See the following for an example output:

```
vyos@site01:~$ show openvpn server

OpenVPN status on vtun2

Client CN       Remote Host           Local Host           TX bytes    RX bytes   Connected Since
---------       -----------           ----------           --------    --------   ---------------

client1         192.168.191.1:55421   N/A                  4.1 KB      61.9 KB    2021-11-02 00:51:59
```

Furthermore, the route specified in the in the configuration in the routing table on windows. The route is highlighted below:

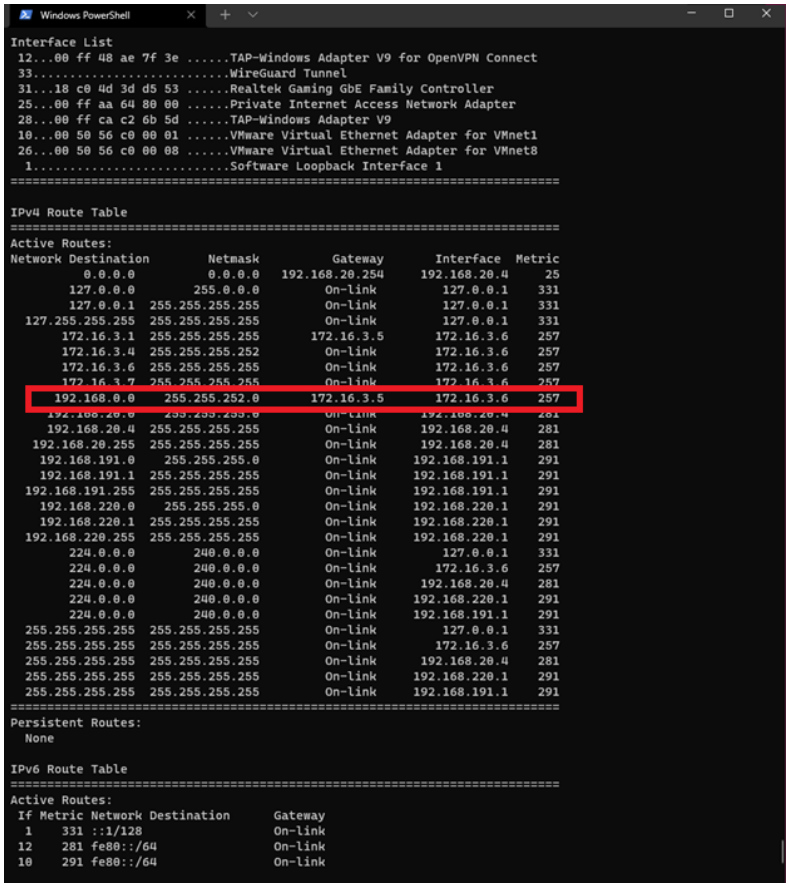


Figure 13 OpenVPN Connect - Inserted Route

Configuration of IPsec

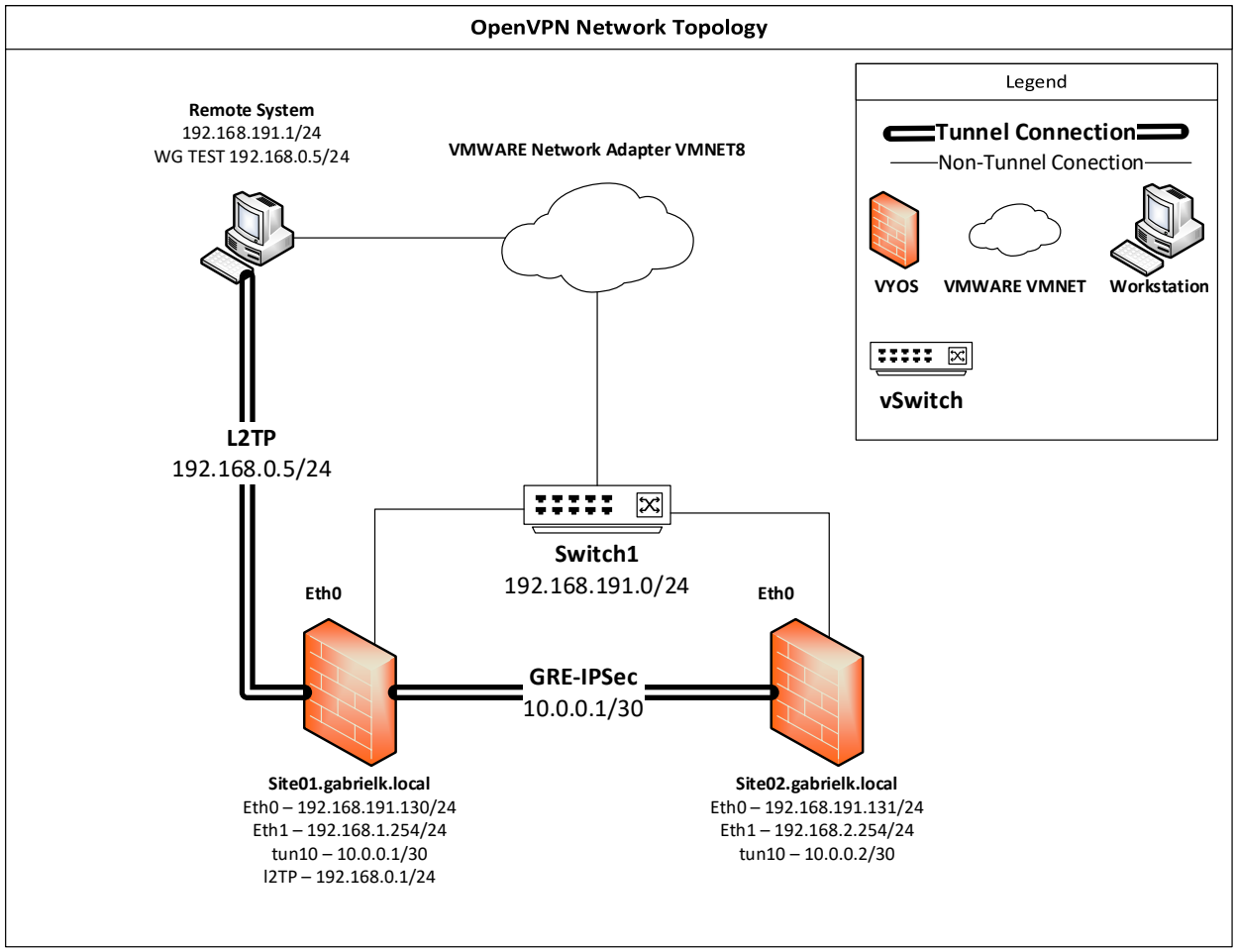


Figure 14 IPsec Topology

IPsec configuration was configured using the guidance from VyOS official guide for 1.3 for both LT2P and Site-to-Site as shown in Figure 14 IPsec Topology (*IPsec — VyOS 1.3.x (Equuleus) Documentation*, n.d.; *L2TP — VyOS 1.3.x (Equuleus) Documentation*, n.d.). The subsections are as follows:

1. Site-to-site
2. Client-Site

Site-to-Site Configuration

There are three major parts to configuring site-to-site configurations. Those are the GRE tunnel, ESP Group and IKE Group. Routes will also need to be configured.

1. To configure, a tunnel address, encapsulation type, remote address, source address, GRE Configuration:

```
set interfaces tunnel tun10 address '10.0.0.1/30'
set interfaces tunnel tun10 encapsulation 'gre'
set interfaces tunnel tun10 remote '192.168.191.131'
set interfaces tunnel tun10 source-address '192.168.191.130'
```

2. ESP Configuration using aes256 for encryption and sha256 for hashing

```
set vpn ipsec esp-group vanESPGroup compression 'disable'
set vpn ipsec esp-group vanESPGroup lifetime '3600'
set vpn ipsec esp-group vanESPGroup mode 'tunnel'
set vpn ipsec esp-group vanESPGroup pfs 'enable'
set vpn ipsec esp-group vanESPGroup proposal 1 encryption 'aes256'
set vpn ipsec esp-group vanESPGroup proposal 1 hash 'sha256'
```

3. IPSEC Configuration with aes256, sha256 for hashing, with nat-tranversal enabled, and on eth0

```
set vpn ipsec ike-group vanIKEGroup close-action 'none'
set vpn ipsec ike-group vanIKEGroup ikev2-reauth 'no'
set vpn ipsec ike-group vanIKEGroup key-exchange 'ikev1'
set vpn ipsec ike-group vanIKEGroup lifetime '28800'
set vpn ipsec ike-group vanIKEGroup proposal 1 dh-group '14'
set vpn ipsec ike-group vanIKEGroup proposal 1 encryption 'aes256'
```

```
set vpn ipsec ike-group vanIKEGroup proposal 1 hash 'sha256'
set vpn ipsec ipsec-interfaces interface 'eth0'
set vpn ipsec nat-networks allowed-network 0.0.0.0/0
set vpn ipsec nat-traversal 'enable'
```

Create the IPsec tunnel and specify the ESP group, IKE group and GRE tunnel:

1. Set a pre-shared key

```
set vpn ipsec site-to-site peer 192.168.191.131 authentication mode 'pre-shared-secret'
set vpn ipsec site-to-site peer 192.168.191.131 authentication pre-shared-secret 'vyos'
```

2. Apply the IKE and GRE group. Ikev2-rauth is set to inherit the behavior of ikegroup

```
set vpn ipsec site-to-site peer 192.168.191.131 default-esp-group 'vanESPGroup'
set vpn ipsec site-to-site peer 192.168.191.131 ike-group 'vanIKEGroup'
set vpn ipsec site-to-site peer 192.168.191.131 ikev2-reauth 'inherit'
```

3. Setting Connection Type. Initiate will start the connection after configuration is applied or after boot. Respond can also be set to not initiate the connection and listen for peer to initiate.

```
set vpn ipsec site-to-site peer 192.168.191.131 connection-type 'initiate'
```

4. Set the address to listen on and disallow NAT networks

```
set vpn ipsec site-to-site peer 192.168.191.131 local-address '192.168.191.130'
set vpn ipsec site-to-site peer 192.168.191.131 tunnel 10 allow-nat-networks 'disable'
set vpn ipsec site-to-site peer 192.168.191.131 tunnel 10 allow-public-networks 'disable'
```

5. Apply encryption on all GRE traffic.

```
set vpn ipsec site-to-site peer 192.168.191.131 tunnel 10 protocol 'gre'
```

6. The firewall rule used to allow all traffic from the peer to be accepted on Site1's side:

```
set firewall name OUTSIDE-LOCAL rule 100 source address '192.168.191.131'
set firewall name OUTSIDE-LOCAL rule 100 action 'accept'
```

Site2 was configured with a Site1's address instead

Once the changes have been committed you can verify if the tunnels are formed by checking the security associations:

vyos@site01:~$ sh vpn ipsec sa

Connection	State	Uptime	Bytes In/Out	Packets In/Out	Remote address	Remote ID	Proposal
peer-192.168.191.131-tunnel-10	up	8m16s	5M/869M	84K/641K	192.168.191.131	N/A	AES_CBC_256/HMAC_SHA2_256_128/MODP_2048
remote-access	up	8m13s	100M/246K	110K/3K	192.168.191.1	N/A	3DES_CBC/HMAC_SHA1_96

vyos@site01:~$ sh vpn ipsec state

src 192.168.191.130 dst 192.168.191.1

proto esp spi 0x9e5e7978 reqid 4 mode transport

replay-window 0

auth-trunc hmac(sha1) 0x4ced2601c7c4640aa895397a21102520e7b06677 96

enc cbc(des3_ede) 0x2afde7df651563cd8921b12d9785c1a55fffd14bf7ae2fb3

anti-replay context: seq 0x0, oseq 0xf81, bitmap 0x00000000
sel src 192.168.191.130/32 dst 192.168.191.1/32
src 192.168.191.1 dst 192.168.191.130
proto esp spi 0xc6ce3174 reqid 4 mode transport
replay-window 32
auth-trunc hmac(sha1) 0x208986a0a31a3013951bb1cc8d3415b9056acfd1 96
enc cbc(des3_ede) 0xae956ba5f53d55714ff01da566fc0b56cee0a305d213166a
anti-replay context: seq 0x1afdf, oseq 0x0, bitmap 0xffffffff
sel src 192.168.191.1/32 dst 192.168.191.130/32
src 192.168.191.130 dst 192.168.191.131
proto esp spi 0xc56b45d9 reqid 1 mode tunnel
replay-window 0 flag af-unspec
auth-trunc hmac(sha256) 0xd724379af241f32549c6e42eed7a6397f197f99ba22fb3552154579dd51328a8 128
enc cbc(aes) 0x23146f7995f4a2218969f1eea205f363549f32203403f2a9be794222ff829f52
anti-replay context: seq 0x0, oseq 0x9caf8, bitmap 0x00000000
src 192.168.191.131 dst 192.168.191.130
proto esp spi 0xc9a76241 reqid 1 mode tunnel
replay-window 32 flag af-unspec
efe48ed5b8944759d8 128
ded6115
anti-replay context: seq 0x14a09, oseq 0x0, bitmap 0xffffffff

Routes for each side as follows:

SITE02:

```
set protocols static route 192.168.1.0/24 next-hop 10.0.0.1
```

SITE01:

```
set protocols static route 192.168.2.0/24 next-hop 10.0.0.2
```

Client-to-Site – Server-Side Configuration

To Configure Site-to-Site, you will need to create users, assign address pool, authentication settings, name servers, interface/ip address to listen on and firewall rules

1. To keep things simple, a new local user is created and use local authentication:

```
set vpn l2tp remote-access authentication local-users username test-user password
'test'
set vpn l2tp remote-access authentication mode 'local'
```

2. Assign IP address range and addressing

```
set vpn l2tp remote-access client-ip-pool start '192.168.0.1'
set vpn l2tp remote-access client-ip-pool stop '192.168.0.250'
set vpn l2tp remote-access name-server '8.8.8.8'
```

3. Setup pre-shared secret for additional authentication

```
set vpn l2tp remote-access ipsec-settings authentication mode 'pre-shared-secret'
set vpn l2tp remote-access ipsec-settings authentication pre-shared-secret 'vyos'
```

4. Set the IP address you listen on:

```
set vpn l2tp remote-access outside-address '192.168.191.130'
```

5. Apply Firewall Rules

Unlike the site-to-site, the IP address of the client is generally not static. As such the firewall rules need to target the protocols that are used in establishing and maintaining connections. Those are ESP Protocol 50, IKE port UDP 500, UDP 4500 for Nat transversal, and L2TP UDP 1701.

To Allow ESP:

```
set firewall name OUTSIDE-LOCAL rule 101 action 'accept'
set firewall name OUTSIDE-LOCAL rule 101 protocol 'esp'
```

```
To Allow IKE:
```

```
set firewall name OUTSIDE-LOCAL rule 102 action 'accept'
```

```
set firewall name OUTSIDE-LOCAL rule 102 destination port '500'
set firewall name OUTSIDE-LOCAL rule 102 protocol 'udp'
```

To Allow NAT Transversal

```
set firewall name OUTSIDE-LOCAL rule 103 action 'accept'
set firewall name OUTSIDE-LOCAL rule 103 destination port '4500'
set firewall name OUTSIDE-LOCAL rule 103 protocol 'udp'
```

To Allow L2TP:

```
set firewall name OUTSIDE-LOCAL rule 104 action 'accept'
set firewall name OUTSIDE-LOCAL rule 104 destination port '1701'
set firewall name OUTSIDE-LOCAL rule 104 ipsec match-ipsec
set firewall name OUTSIDE-LOCAL rule 104 protocol 'udp’
```

6. Set route on site02 for 192.168.2.0/24 to reach 192.168.0.0/24

```
set protocols static route 192.168.0.0/24 next-hop 10.0.0.1
```

Client-to-Site – Client Side Configuration

Client side was configured by creating a new VPN configuration via windows networking settings using the values set on the server side. See the following for an example:

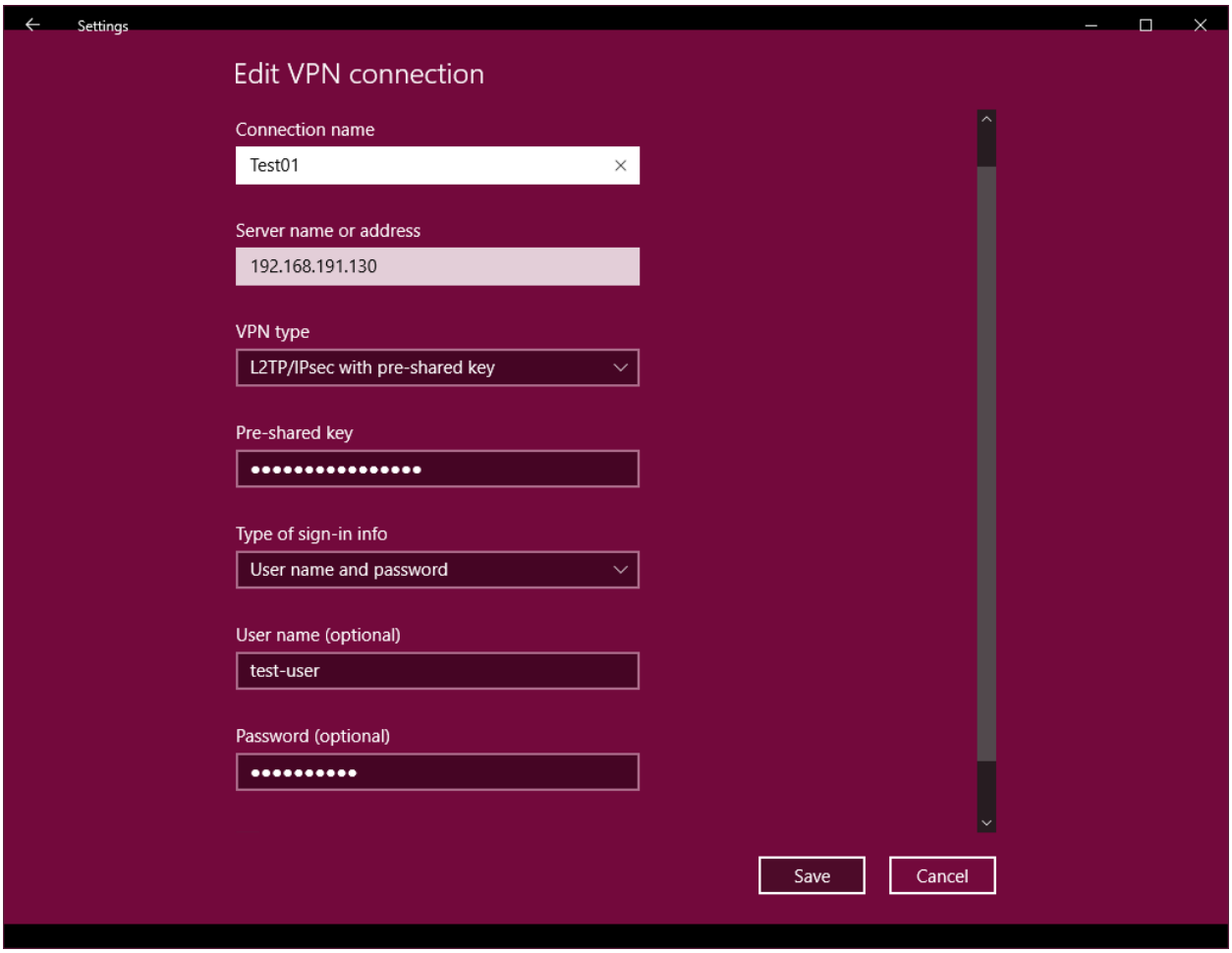


Figure 15 Settings for L2TP on Windows

Once the VPN connection has been created, under the interface properties in Windows Networking Adapters, under security, Microsoft Chap version 2 needs to be enabled. See the following screenshot for details.

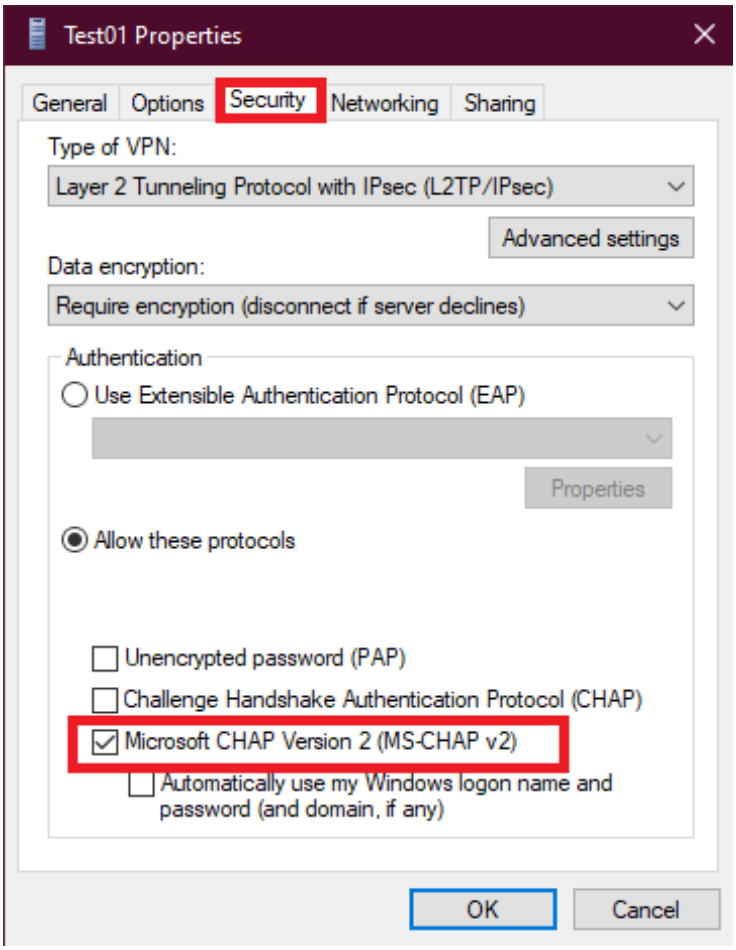


Figure 16 L2TP Windows - Microsoft CHAP Version2

Once this is done, you can activate the configuration. Once Connection is established it should show as below:

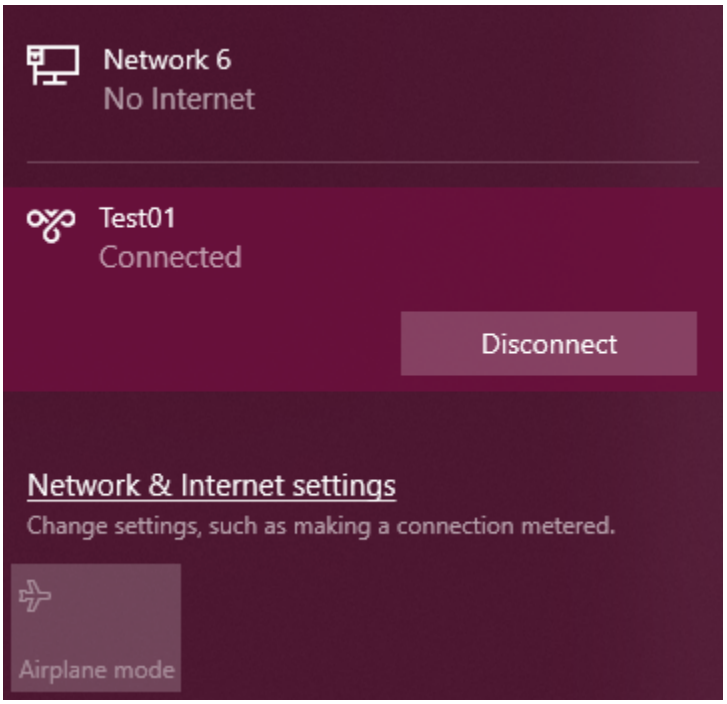


Figure 17 L2TP Windows - Activated l2TP Connection

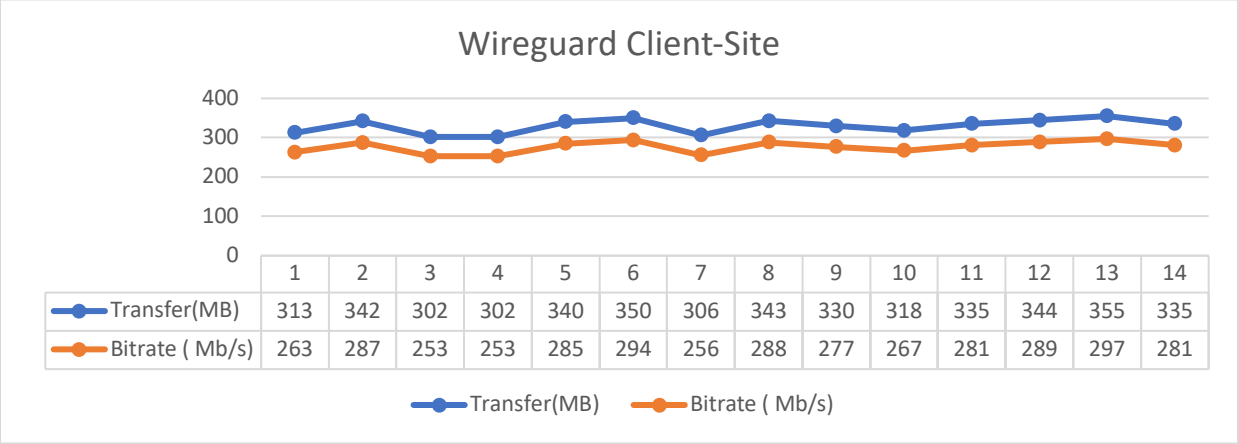
Verify on the Vyos via `show l2tp-server sessions`:

```
vyos@site01:~$ show l2tp-server sessions
ifname | username  |     ip      | ip6 | ip6-dp |  calling-sid  | rate-limit | state  |  uptime  | rx-bytes  | tx-bytes
--------+----------+-------------+-----+--------+---------------+------------+--------+----------+-----------+----------
 l2tp0  | test-user | 192.168.0.5 |     |        | 192.168.191.1 |            | active | 00:08:55 | 976.5 MiB | 4.1 MiB
```

Throughput Testing

Tests were done via iperf3 from remote client to routes, the basic VPCS in GNS3 lack iperf3. Each datapoint is an average of a test that runs for 10 seconds for a total of 15 tests per scenario The scenarios were site-to-site, client-to-site, client-to-site-to-site. Site to site connections were tested using the inside facing interfaces (192.168.1.254 and 192.168.2.254 for site1 and site 1 respectively) going through their respective VPN tunnels to each other. Client-to-site tests were done from the remote connection to site 1 to the iperf3 server running on site 2.

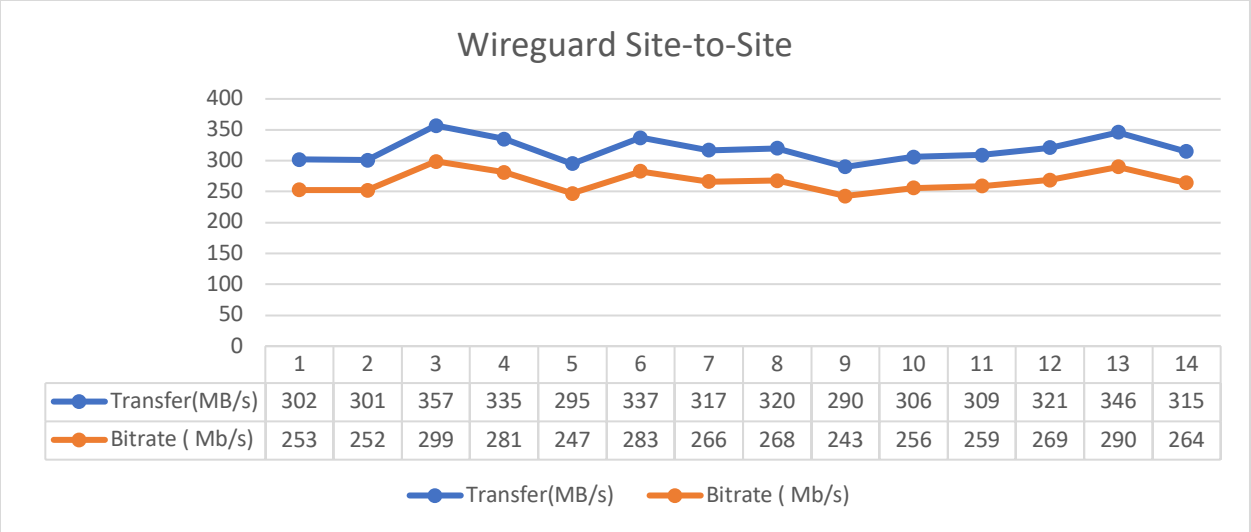
Wireguard



	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Transfer(MB)	313	342	302	302	340	350	306	343	330	318	335	344	355	335
Bitrate (Mb/s)	263	287	253	253	285	294	256	288	277	267	281	289	297	281

Figure 18 Throughput Tests - Wireguard Client-Site

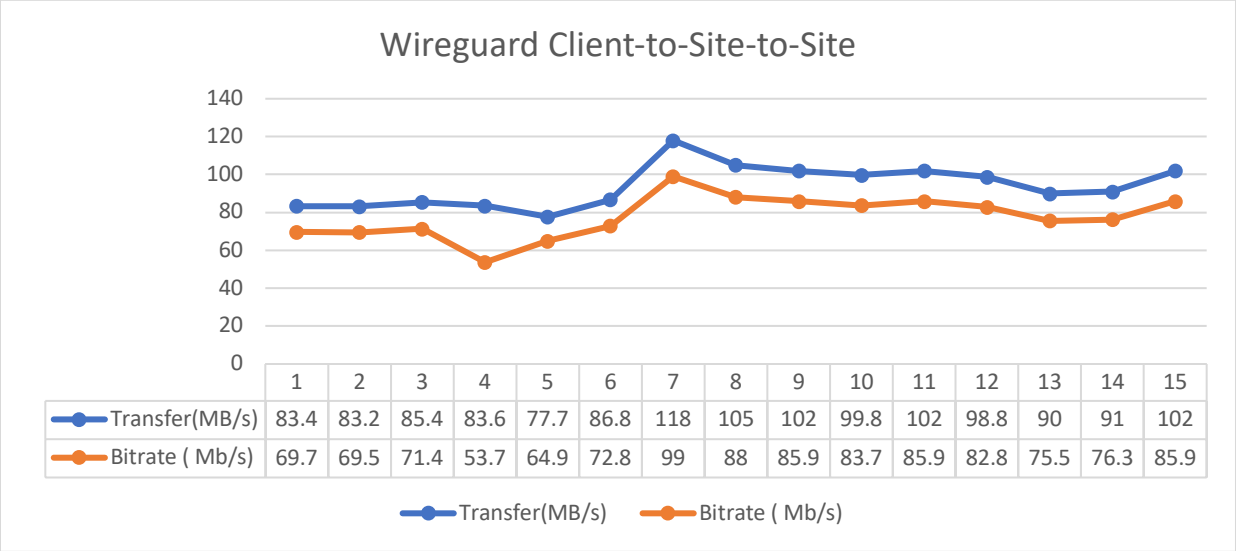
Wireguard Client-Site performance is good with an average transfer of 319.4MB and average 267.66Mbit/s transfer speeds. Maximum transfers were 357MB and minimums of 290MB. Bitrate max was 299mb/s with 243mb/s for lows. That is a difference of 67 for transferred MB and 56 for MB/s.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Transfer(MB/s)	302	301	357	335	295	337	317	320	290	306	309	321	346	315
Bitrate (Mb/s)	253	252	299	281	247	283	266	268	243	256	259	269	290	264

Figure 19 Throughput Tests - Wireguard Site-to-Site

Site-to-site performance is similar. Average of 329.64 MB per test and 276.5mb/s transfer speeds. Maximum transfers of 355 and minimums of 302. Maximum rate of transfer at 297mb/s and minimum of 253mb/s. The difference for transferred data is 53MB and 44Mb/s for transfer rate.



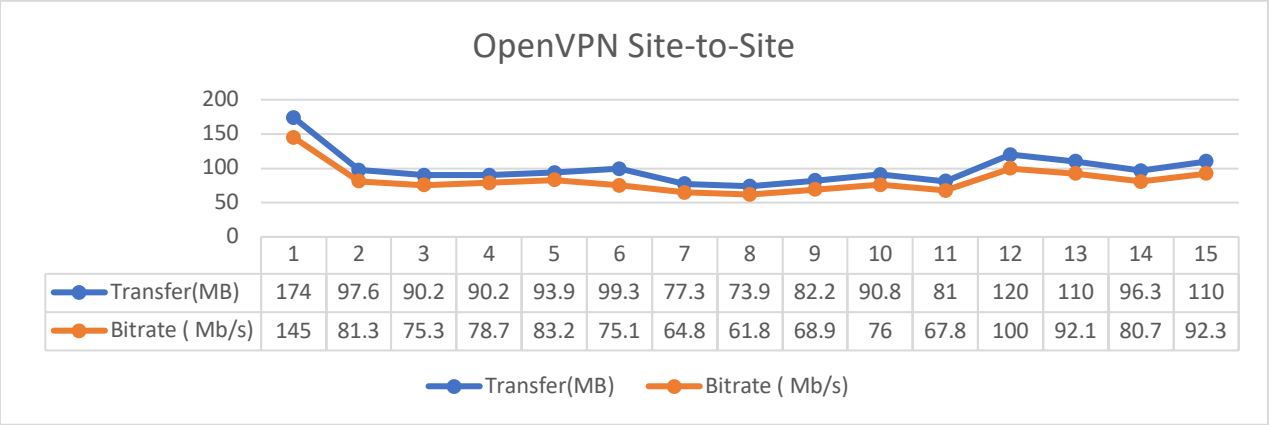
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Transfer(MB/s)	83.4	83.2	85.4	83.6	77.7	86.8	118	105	102	99.8	102	98.8	90	91	102
Bitrate (Mb/s)	69.7	69.5	71.4	53.7	64.9	72.8	99	88	85.9	83.7	85.9	82.8	75.5	76.3	85.9

Figure 20 Throughput Tests - Wireguard Client-to-Site-Site

There is a drastic drop in performance when remote client needs to cross through the tunnel between the two sites. Average transfers of 93.91 and average bitrate of 77.66. The maximum data

transferred was 118MB and minimum was 99MB. Max bitrate of 99 and minimum of 53.7. The difference in maximum and minimum for both transferred data and transfer rate of 40.3 and 45.3 respectively. Compared to the client-site test, we see a significant drop of average transferred data and transfer rates of 235.73MB and 198.83Mb/s respectively.

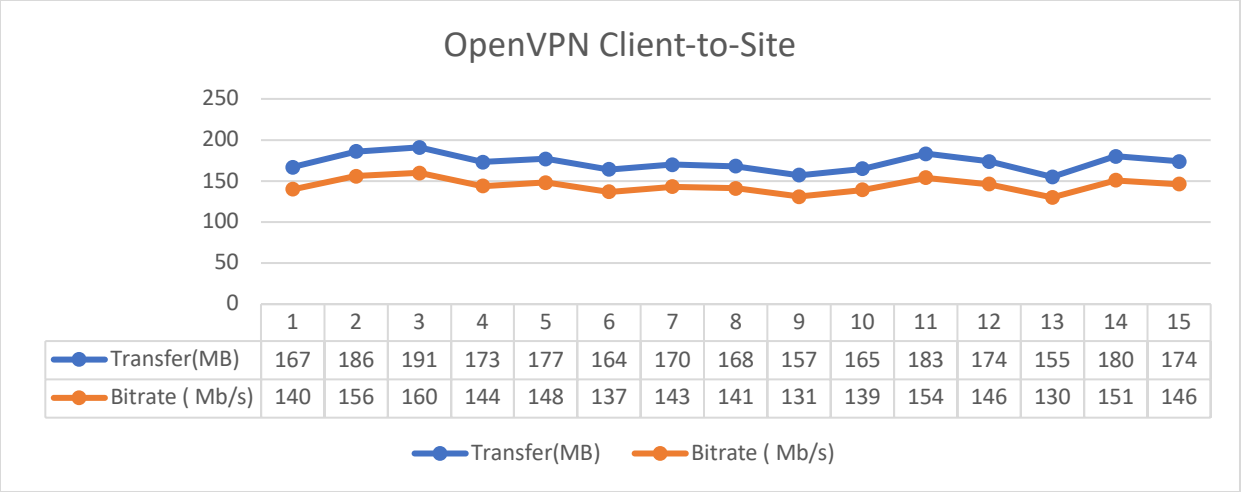
OpenVPN



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Transfer(MB)	174	97.6	90.2	90.2	93.9	99.3	77.3	73.9	82.2	90.8	81	120	110	96.3	110
Bitrate (Mb/s)	145	81.3	75.3	78.7	83.2	75.1	64.8	61.8	68.9	76	67.8	100	92.1	80.7	92.3

Figure 21 Throughput Tests - OpenVPN Site-to-Site

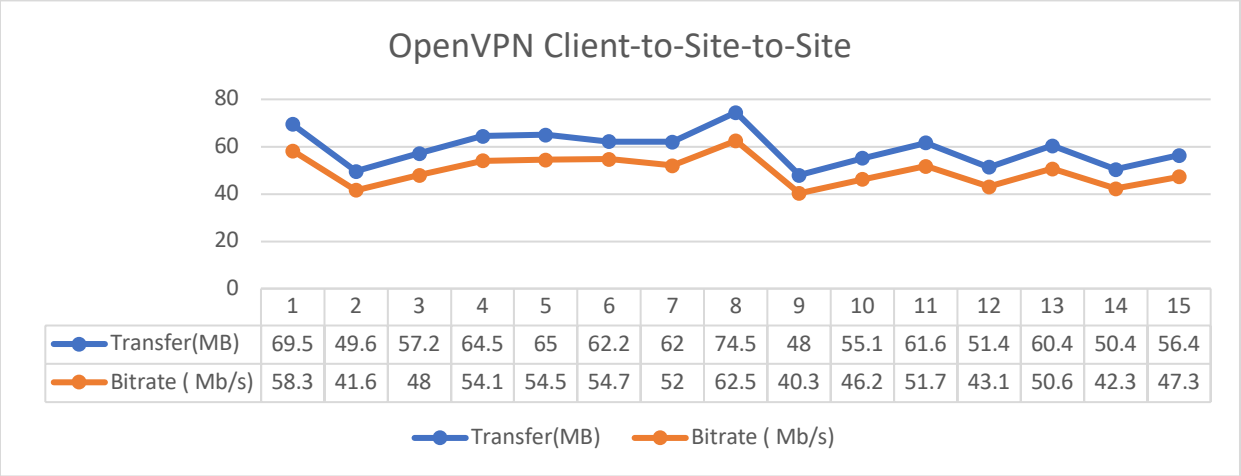
OpenVPN shows some strange behavior with the site-to-site behavior with an initial test showing respectable speeds at 174MB transferred at 145Mb/s but over time settles to an average of 99.11 MB transferred and 82.86Mb/s. This gives it a large range between its peak performance and lowest performance. Max transferred MBs was 174 as mentioned, with a minimum of 73.9. Difference is 100.1 MB. Bit rate range is equally large at 83.2 between its peak at 145Mb/s and 61.8M/b. If we exclude anomaly, difference between the next highest of data transferred and rate of transfer is 46.1 and 58.2 respectively. This range between max and minimum values of transfer rate and transferred data seen in line with the site-to-site testing on Wireguard. Regardless, OpenVPN significantly slower when compared to the 319.4 average data transfer and 267.67 data transfer rate in the site-to-site connection.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Transfer(MB)	167	186	191	173	177	164	170	168	157	165	183	174	155	180	174
Bitrate (Mb/s)	140	156	160	144	148	137	143	141	131	139	154	146	130	151	146

Figure 22 Throughput Tests - OpenVPN Client-to-Site

Client to site performance is quite good compared to the site-to-site performance. Average transferred data and transfer rate were 172.27MB and 144.4 Mb/s respectively. Moreover, range between maximum for transferred data and transfer rate was at 28MB and 24Mb/s respectively.



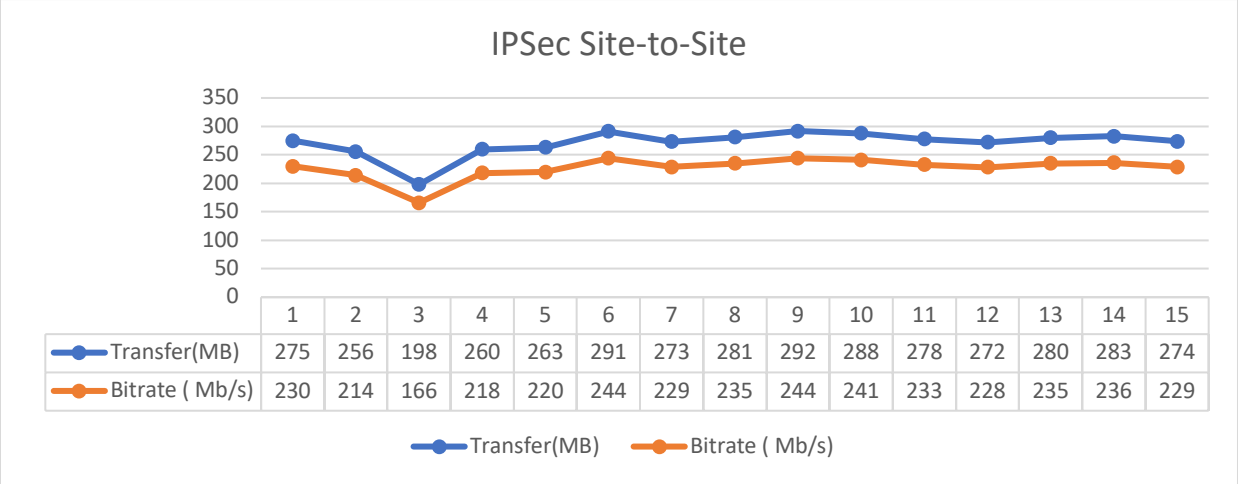
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Transfer(MB)	69.5	49.6	57.2	64.5	65	62.2	62	74.5	48	55.1	61.6	51.4	60.4	50.4	56.4
Bitrate (Mb/s)	58.3	41.6	48	54.1	54.5	54.7	52	62.5	40.3	46.2	51.7	43.1	50.6	42.3	47.3

Figure 23 Throughput Tests - OpenVPN Site-to-Site

Client to site2 shows some erratic behavior like OpenVPN site-to-site as seen above. However, average transferred data was 59.19 MB and average transfer rate was 49.81Mb/s. This is a drop in 39.93 in average data transferred and 33.05 in rate of transfer. This quite respectable given the major drop in throughput compared to Wireguard in a similar scenario.

IPsec\L2TP

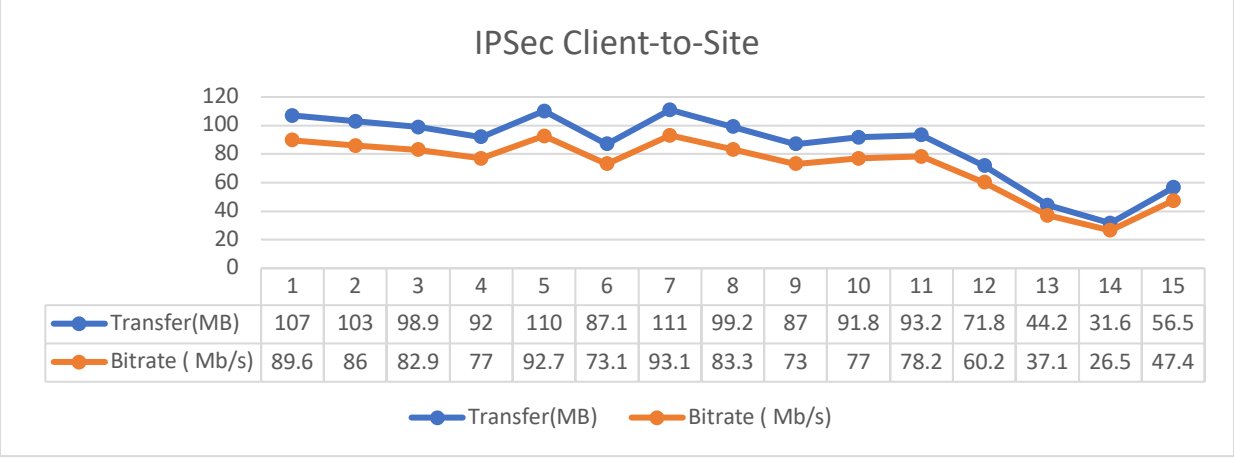
Overall, IPsec performance has been lacking luster based on this test done outside of site-to-site. In between each scenario, the VPN connections had to be restarted as connection with break.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Transfer(MB)	275	256	198	260	263	291	273	281	292	288	278	272	280	283	274
Bitrate (Mb/s)	230	214	166	218	220	244	229	235	244	241	233	228	235	236	229

Figure 24 Throughput Test - IPsec Site-to-Site

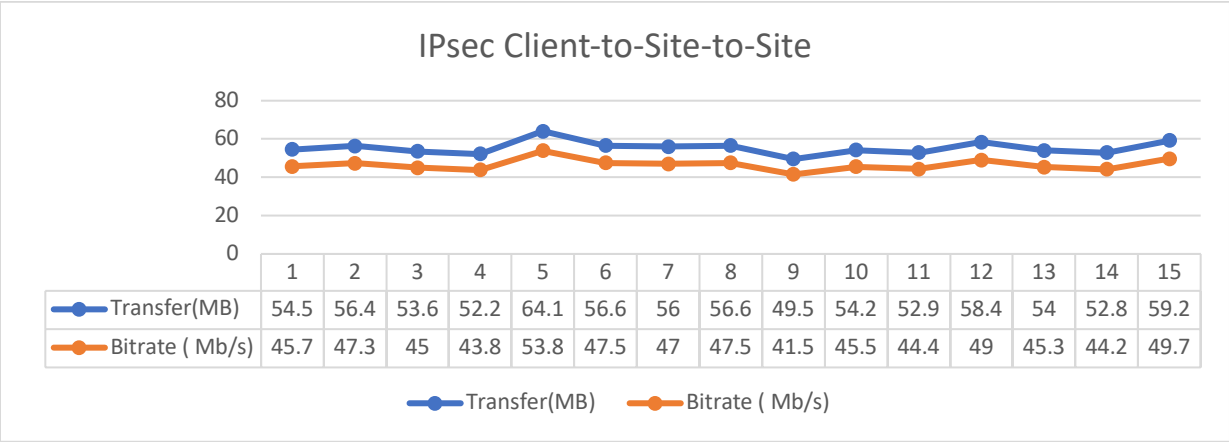
IPsec Site-to-Site performance is quite high as well, although not as high as Wireguard with an average data transferred of 270.93MB and average transfer rate of 226.8MB/s. However, there is a major dip in test 3 which results in a major difference between the maximum and minimum values for both transferred and bitrate. Despite this, minimum data transfer of 198 and 166 for transfer rate is still higher than average of OpenVPN site-to-site at 99.11MB and 82.86MB/s.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Transfer(MB)	107	103	98.9	92	110	87.1	111	99.2	87	91.8	93.2	71.8	44.2	31.6	56.5
Bitrate (Mb/s)	89.6	86	82.9	77	92.7	73.1	93.1	83.3	73	77	78.2	60.2	37.1	26.5	47.4

Figure 25 Throughput Tests - IPsec Client-to-Site

For the Client to site performance test, there are some major dips in performance in the last couple tests. It appears to consistently be above 60 MB transferred and 60Mb/s transferred. Nonetheless averages were 85.62MB and 71.81Mb/s for data transferred and transfer rates respectively.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Transfer(MB)	54.5	56.4	53.6	52.2	64.1	56.6	56	56.6	49.5	54.2	52.9	58.4	54	52.8	59.2
Bitrate (Mb/s)	45.7	47.3	45	43.8	53.8	47.5	47	47.5	41.5	45.5	44.4	49	45.3	44.2	49.7

Figure 26 Throughput Tests - Client-Site-to-Site

Client to site-to-site performance was much more consistent with averages of 55.4MB transferred and 46.48MB/s transfer rate. Range between Max and minimum transferred data and bitrate was 14.6MB and 12.3Mb/s. However this much lower than expected much like the client-to-site performance tests.

Speed Test Conclusions

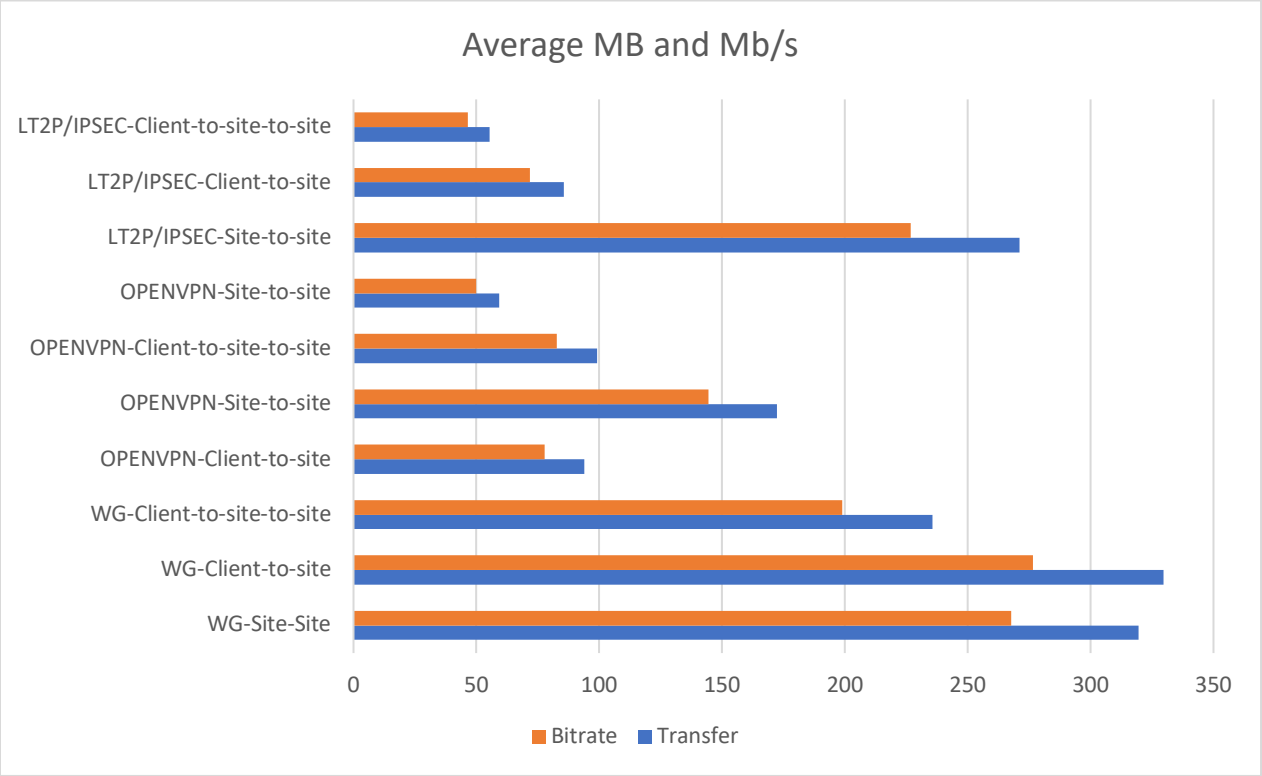


Figure 27 Throughput Tests - Overall Average Speed

The expectation going into the tests was that Wireguard would be faster than its alternatives. Based on the results this appears to be the case. However, as noted previously, there were some notable performance regressions when using L2TP versus GRE which resulted in poorer than expective outcomes even when compared to OpenVPN. Conversely, IPsec and GRE quite performant but still not as performant as Wireguard. However, it is still slower when compared to Wireguard in most of the scenarios. Evidently, Wireguard is demonstrably performant.

Drawbacks and Other Considerations

Vendor Support

Wireguard implementation outside of standalone servers Linux and windows deployments, Vyos, and pfSense and its derivatives are minimal. Major vendors, such as Cisco and Palo Alto, are more focused on compatibility rather than pure speed. While there is support for Wireguard support for Mikrotik, pfSense, OpenWRT and OPNsense support as well as Linux and Windows servers also available, implementation is not with controversy(Grigoryev, 2019; MikroTik, 2021; OPNsense, 2021). In 2020, there was a major issue with its implementation in OpenBSD on which pfSense is based. Donenfeld was quoted as being disappointed with its implementation on in pfSense and OpenBSD with the code ultimately pulled from the OpenBSD 13 release(Anderson, 2021). While one could argue that simply not implementing Wireguard on pfSense or OpenBSD13 would resolve the issue, this shrinks the level of OS/Device support of an already limited pool of vendors that support the protocol. Furthermore, given its opinionized nature of its primitives, it could result in devices or sites being disconnected behind simply because their vendor of choice has a broken implementation of Wireguard that cannot be updated.

Flexibility

Wireguard simplicity comes at the cost of flexibility. Because Wireguard is opinioned in its cryptography, if the maintainers were to change the cryptographic primitives, upgrades would be necessary across Wireguard infrastructure. While this is less problematic for networks in which an admin has full control over, it is much less tenable in other situations. An example would when implementing Wireguard and working with multiple partners with different upgrade cycles. Unlike Wireguard, OpenVPN and IPsec that allow cipher negotiation to meet the needs of peers regardless of their current implementations(Tremer, 2020). This would allow for gradual roll outs of changes to ciphers rather than needing organize with partners to make a coordinated upgrade.

Nonetheless, this level of flexibility increases complexity. As Pennarun highlights in his critique of Tremer position, IPsec was already considered too complex in 2003 to be secure and has only gotten more complexity in order to meet the needs of the industry(Pennarun, 2020). In

contrast, Wireguard has an impressive 4000 LOC and have received high praise from even Linux maintainer Linus Torvalds(*Linux-Kernel Archive: Re: [GIT] Networking*, n.d.) This is contrast to the 600,000 to 400,000 LOC of OpenVPN and IPsec respectively(Salter, 2018). Naturally, this smaller code base makes it easier to maintain and audit.

Administrative Overhead

Overall, the initial administrative overhead in VyOS for both OpenVPN and L2TP/IPsec is relatively high when compared to Wireguard. Whereas Wireguard only required the generation of a key pair and setting up of the tunnel network settings, both OpenVPN and IPsec significantly more configuration. For OpenVPN, this included the need to generation and export of certificates which involves the need to maintain a PKI. IPsec and L2TP also required a non-insignificant more configuration when compared to Wireguard. This involved setting IKE and ESP parameters, as well as several additional firewall entries.

However, Wireguard configuration might not be as scalable when it comes to managing large number of remote users. Because identities are handled entirely through key pairs, there is a lack of built-in integration with industry standards for AAA such as RADIUS or use of LDAP for identity management. While there are some advanced management tools such as wg-manager being worked on and more rudimentary tools built in VyOS to manage Wireguard mobile configuration, it does not compare to the use of multitude of mature products for managing RADIUS or LDAP and integration with OpenVPN and IPsec (Andersen, 2020/2021; *Wireguard*, n.d.).

Another drawback is addressing. As seen in our setup, both OpenVPN and IPsec have built in methods for assigning addresses from a specified pool. This is not the case with Wireguard which requires manual assignment when configuring peers. While there has been some development regarding implementing a similar feature via wg-dynamic, progress has been limited. The last public commit to this project was in 2019 and is expressly stated to be a work in progress making it not suitable for production deployment(*Wg-Dynamic - Dynamic Configuration*

Daemons for WireGuard, n.d.). As such, as additional clients are added, configuration could become unwieldy could be a source of misconfiguration if not careful.

Another issue that Tremor brings up is that Wireguard is lacking when it comes to dealing with dynamic IP addresses for the server side. While this may have been true during the early implementations, Wireguard can be set to use DNS names per Pennarun. Pennarun does concede that Wireguard clients will need to restart their clients if the server has its IP addresses changed; DNS lookup only occurs at client launch(Pennarun, 2020).

Commercial Providers

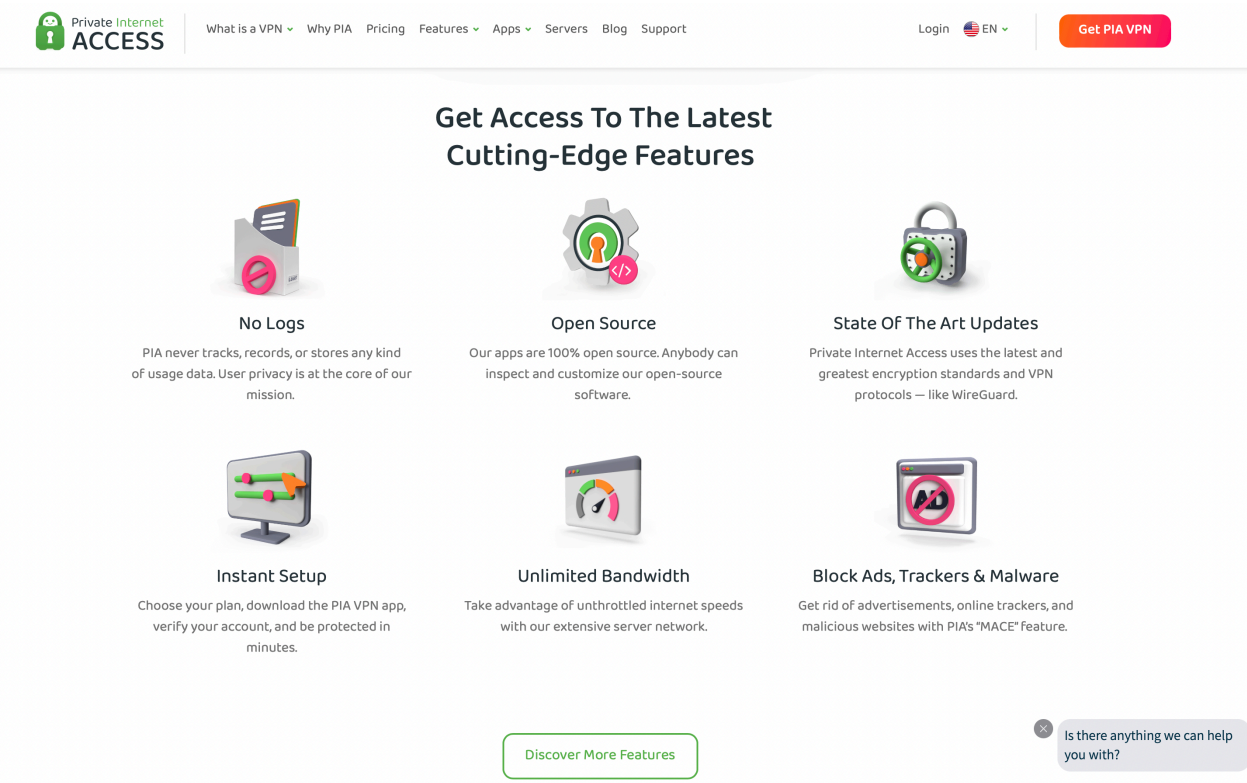


Figure 28 PIA Features from Website

Although not entirely within the scope of this paper, exposure of Wireguard through consumer providers of VPN in the general public could potentially result in adoption in corporate environments due the increased mindshare. Per Josh's post *Which VPNs Use WireGuard?* on All Things Secured, there are 19 vendors that support Wireguard as an alternative connection method(All Things Secured, 2021). Those include NordVPN, Surfshark, Private Internet Access. Those who are using consumer VPNs are likely more technically savvy. A subset of that group is

also likely working in the IT industry. Furthermore, that some of the vendors such as NordVPN prominently feature Wireguard as part of its feature set. Increased mind share might lend to increased support for the protocol both in the consumer space but also corporate IT.

Conclusion

To answer the question of whether Wireguard could replace current OpenVPN or IPsec deployments, no in most scenarios. Although from a performance standpoint Wireguard takes a leadership position, it is still lacking in some features that make IPsec or OpenVPN desirable. Those include integration with established authentication methods such as RADIUS, protocol flexibility and adoption by major appliance vendors. These features make OpenVPN and IPsec easier to scale and deploy against existing infrastructure. Conversely, Wireguard can excel in small-scale and simpler topologies with fewer parties involved. In these small-scale environments, Wireguard simplicity and ease of setup is a clear benefit over otherwise complex setup for IPsec and OpenVPN whose complexity to not provide any true benefit. Furthermore, if solutions from Microtik or OPNSense, or pfSense are already in place, Wireguard is also worthy alternative as said solutions have Wireguard implementation built in.

References

All Things Secured, J. (2021, November 1). *Which VPNs Use WireGuard? (As of November 2021)*. All Things Secured. https://www.allthingssecured.com/vpn/wireguard-vpn-list/

Andersen, P.-A. (2021). *Wg-manager* [Python]. https://github.com/perara/wg-manager (Original work published 2020)

Anderson, T. (2021, March 23). *FreeBSD 13.0 to ship without WireGuard support as dev steps in to fix "grave issues" with initial implementation*. https://www.theregister.com/2021/03/23/freebsd_130_no_wireguard/

Donenfeld, J. A. (n.d.). Wireguard. *Protocols and Cryptography*. https://www.wireguard.com/protocol/

Donenfeld, J. A. (2017). WireGuard: Next Generation Kernel Network Tunnel. *Proceedings 2017 Network and Distributed System Security Symposium*. Network and Distributed System Security Symposium, San Diego, CA. https://doi.org/10.14722/ndss.2017.23160

Grigoryev, V. (2019, August 2). *WireGuard*. OpenWrt Wiki. https://openwrt.org/docs/guide-user/services/vpn/wireguard/start

IPsec—VyOS 1.3.x (equuleus) documentation. (n.d.). Retrieved November 22, 2021, from https://docs.vyos.io/en/equuleus/configuration/vpn/ipsec.html?highlight=ipsec

L2TP — VyOS 1.3.x (equuleus) documentation. (n.d.). Retrieved November 22, 2021, from https://docs.vyos.io/en/equuleus/configuration/vpn/l2tp.html?highlight=ipsec

Linux-Kernel Archive: Re: [GIT] Networking. (n.d.). Retrieved November 18, 2021, from http://lkml.iu.edu/hypermail/linux/kernel/1808.0/02472.html

MikroTik. (2021). *WireGuard—RouterOS - Documentation*. https://help.mikrotik.com/docs/display/ROS/WireGuard

OpenVPN — VyOS 1.3.x (equuleus) documentation. (n.d.). Retrieved November 22, 2021, from https://docs.vyos.io/en/equuleus/configuration/interfaces/openvpn.html

OpenVPN Remote Access VPNs Using TLS on VyOS | Brezular's Blog. (n.d.). Retrieved November 22, 2021, from https://brezular.com/2019/12/18/openvpn-remote-access-vpns-using-tls-on-vyos/

OPNsense. (2021). *WireGuard Site-to-Site Setup—OPNsense documentation*. https://docs.opnsense.org/manual/how-tos/wireguard-s2s.html

Pennarun, A. (2020, April 23). *Why not "Why not WireGuard?"* Tailscale. https://tailscale.com/blog/why-not-why-not-wireguard/

Salter, J. (2018, August 26). *WireGuard VPN review: A new type of VPN offers serious advantages*. Ars Technica. https://arstechnica.com/gadgets/2018/08/wireguard-vpn-review-fast-connections-amaze-but-windows-support-needs-to-happen/

Tremer, M. (2020, February 18). *Why Not WireGuard—The IPFire Blog*. IPFire Blog. https://blog.ipfire.org/post/why-not-wireguard

wg-dynamic—Dynamic configuration daemons for WireGuard. (n.d.). Retrieved November 23, 2021, from https://git.zx2c4.com/wg-dynamic/

Wireguard. (n.d.). WireGuard — VyOS 1.3.x (Equuleus) Documentation. Retrieved November 18, 2021, from https://docs.vyos.io/en/equuleus/configuration/interfaces/wireguard.html

Wu, P. (2019). *Analysis of the WireGuard protocol*. 89.

Appendix

Peer Comments

Comments are from Calvin Hua Nguyen. He is former classmate at the British Columbia Institute of Technology who currently is working at D-Wave as an IT Support Specialist. Linked-in Profile: https://www.linkedin.com/in/calvin-hua-nguyen/

The following transcript is pulled from a Discord Chat:

TechDisconnect: really interesting read. Loved the breakdown on the packets and speed comparisons. I had read of the speed advantages of wireguard, but never really looked too deep, so cool to see the comparisons.

Honestly, I don't know enough to point out any major inaccuracies, but just had some questions on certain areas:
1. On page 6, you specified the following, which I think may be a typo:

Where IPSec requires configuring additional GRE or LT2P interfaces for site-to-site or client configurations, OpenVPN and IPsec do not require an additional tunneling protocol to be configured

I think IPsec is repeated twice there, when it should be wireguard in the latter half.

2. On page 13, I'm a bit confused on this line:

In the case that the UDP packet does hit the maximum MTU, the message is padded in multiples of 16 bytes. If the message is a keep alive, no padding is done and encrypted packet is 16 Bytes

If the packet hits the MTU, why would it need to be padded? Do you mean fragmented?
TechDisconnect: And man, configuring this traditionally on a router is so different than through Docker.
TechDisconnect: For my deployment, it's been smooth. The fact that my family defaults to the VPN outside and no one has complained, speaks wonders to reliability.
TechDisconnect: Oh yea, manually creating and maintaining keys for users is a job on it's own

Personal Comments

It was relatively easy to find information regarding VyOS and Wireguard. There were some existing research that I could draw upon and cited in this paper to help understand the protocol better. However, given the lack of maturity of the protocol there were some conflicting information even in the white paper penned by the author of the protocol. Certain sections of the message headers were inaccurate in terms of the size and external research papers and Wireshark captures revealed this discrepancy. Other issues I ran into was navigating VyOS in general. It took some time to figure out how to configure and manage it. Furthermore, there were some major breakages in GNS3 caused by python versioning that needed to be resolved before any implementation could be done. In retrospect, it could have been easier to avoid using GNS3 altogether and reduce the overhead from the GNS3VM.

Scope of the project did not change much. Initially the plan was to go with a full network simulation but because of the overhead from GNS3VM there were limited options for testing. VPCs GNS3 provided do not support iPerf3 testing. Testing fell back to running IPerf3 on the internal interface of site01 and site02 and omitting the VPC on each private subnet all together. Also had initially planned to go more in depth in the primitives used by IPsec and OpenVPN but was lacking time to do further research on that front.

As for the valuableness of this research, I believe it will be valuable long term. I have no doubt that Wireguard is contender the VPN space and shows potential in replacing OpenVPN and IPsec for site-to-site connections in the long term. Less so for client connections until there are easier methods of user management. As such, learning about the technology ahead of time and keeping up to date with it could prove useful should an appropriate scenario arise for implementing Wireguard over its alternatives.