VPN - L2TP/IPsec

L2TP/IPSec is a common general-purpose VPN protocol that work with most platforms. For example, computers running Windows, macOS, iPhones, and Android devices all support this type of VPN out-of-the-box. This type of VPN is a little bit oldschool, in that it is typically found in enterprise corporate environments, which is part of what makes it so ubiquitous. For this reason, we have decided to provide and endpoint of this protocol.

Technically speaking, it is a combination of L2TP and IPsec protocol standards. The former protocol (L2TP, the Layer 2 Tunneling Protocol) carries a "call" over the Internet, while the latter protocol suite (IPsec, Internet Protocol Security) encrypts the call and authenticates the participants. Although it's possible to use either protocol without the other, L2TP and IPsec are most often used together. This is because L2TP creates a connection but does not secure the connection, while IPSec protects traffic but does not itself create a network connection to carry traffic.

L2TP/IPsec VPNs are often slower than more modern protocols because the implementation on smaller routers is usually implemented in software which makes heavy use of the router's CPU. On the other hand, more expensive routers often have a custom chip to speed-up IPSec encryption, actually making L2TP/IPsec the fastest choice in certain cases.

The best performance you can reasonably expect from on small routers is around ~100Mbps.

Device support

The L2TP/IPsec protocol enjoys very wide support across platforms and vendors.

- Windows devices: Should work - Guide Here
- Apple devices: iPhones, Mac laptops, natively - iOS Guide Here, macOS Guide Here
- Android devices: Most yes - Guide Here
- Linux: Yes, complicated without GUI, works well with Network Manager - Guide Here
- Ubiquiti EdgeOS: Yes, but slow - Guide Here
- Mikrotik devices: All (various speeds) - No Guide Yet
- OpenWRT: Should work - No Guide Yet

Endpoints

This section provides connection information for NYC Mesh VPN endpoints that use L2TP/IPsec.

Supernode 1

- Server domain name: `l2tpvpn.sn1.mesh.nycmesh.net`
- Supported connection methods:
  - Anonymous:
    - Username:
    - Password:
    - Pre-shared key/secret:
  - OSPF Node-Peering (Same, connect as above, use OSPF afterwards.)

Supernode 3

- Server domain name: `l2tpvpn.sn3.mesh.nycmesh.net`
- Supported connection methods:
  - Anonymous:
    - Username:
    - Password:
    - Pre-shared key/secret:
  - OSPF Node-Peering (Same, connect as above, use OSPF afterwards.)

Account

- To request a vpn access

N.B. Your account will be specific to SN1 or SN3 - note in the comments if you have a preference

Connection Guides

Please follow the below connection guides for each platform.

Windows 10

Expand to view instructions...

1. Click on Start (Title menu) and type VPN
2. Click on on Change Virtual Private Networks (VPN)
3. Click on the plus button (Add a VPN connection)
4. Choose VPN provider (Microsoft by default)
5. Connection name (Name it whatever you want)
6. Server name or address `l2tpvpn.sn1.mesh.nycmesh.net`
7. VPN Type: L2TP/IPsec with pre-shared key
8. Pre-shared key: `nycmeshnet`
9. Type of sign-in info: User name and password
10. Username: `your personal user name`
11. Password: `your personal password`
12. Check box to remember password so you don't have to type this everytime
13. Click save
14. Click on newly created VPN connection and click connect

macOS

Expand to view instructions...

See Apple Support: Set up a VPN Connection. Be sure to use the appropriate authentication credentials for your connection. For an anonymous connection, enter `your personal user name` as the "account name" and `your personal user name` in the User Authentication's "password" field *and* 'nycmeshnet' the Machine Authentication's "Shared Secret" field.

Apple iOS

Expand to view instructions...

These instructions refer to Apple-branded handheld devices such as the iPhone and iPad.

1. Go to Settings
2. Tap on VPN
3. Tap on Add VPN Configuration
4. Tap on Type and choose L2TP
5. Description (Anything you want)
6. Server: `l2tpvpn.sn1.mesh.nycmesh.net`
7. Account: `your personal user name`
8. Leave RSA SecurID off

9. Password `your personal password`
10. Secret: `nycmeshnet`

See also How-To Geek: How to Connect to a VPN From Your iPhone or iPad § Connect to IKEv2, L2TP/IPSec, and Cisco IPSec VPNs in iOS.

Android

Expand to view instructions...

See How-To Geek: How to Connect to a VPN on Android § Android's Built-In VPN Support.

GNU/Linux - GNOME/Network Manager

Expand to view instructions...

Using GNOME/Network Manager:

1. Make sure you have the L2TP/IPsec NetworkManager plugin installed (`NetworkManager-l2tp-gnome` on Fedora)
2. Add a new VPN of type 'Layer 2 Tunneling Protocol'
3. Gateway: `l2tpvpn.sn1.mesh.nycmesh.net`
4. Username: `your personal user name`
5. Password: `your personal passwrod` (you may have to click a question mark on the right of the textbox to allow saving the password)
6. Click "IPsec Settings"
7. Check "Enable IPsec tunnel to L2TP host"
8. Pre-shared key: `nycmeshnet`
9. Save & Connect

Ubiquiti EdgeOS

Expand to view instructions...

This example EdgeRouter configuration will let clients on your LAN reach the mesh. It requires at least EdgeOS 2.0.9 which adds support for connecting to L2TP/IPsec VPNs. You will need to

be familiar with the EdgeOS CLI.

Note: there is a bug in EdgeOS's PPP configuration that prevents EdgeRouter from connecting to the NYC Mesh VPN. Make sure to configure the workaround scripts on your EdgeRouter.

Here is a minimal configuration for connecting to the Supernode 1 VPN.

First, enter configuration mode:

```
configure
```

Then, configure the L2TP client interface (you should be able to copy and paste all the lines in this block at once):

```
set interfaces l2tp-client l2tpc0 server-ip l2tpvpn.sn1.mesh.nycmesh.net
set interfaces l2tp-client l2tpc0 description "NYC Mesh VPN (SN1)"
set interfaces l2tp-client l2tpc0 authentication user-id 'your personal user name'
set interfaces l2tp-client l2tpc0 authentication password your personal password'
set interfaces l2tp-client l2tpc0 require-ipsec
```

Next, configure the IPsec tunnel:

```
set vpn ipsec esp-group NYC_MESH mode transport
set vpn ipsec esp-group NYC_MESH pfs disable
set vpn ipsec esp-group NYC_MESH proposal 1 encryption aes256
set vpn ipsec esp-group NYC_MESH proposal 1 hash sha1
set vpn ipsec ike-group NYC_MESH dead-peer-detection action restart
set vpn ipsec ike-group NYC_MESH proposal 1 encryption aes256
set vpn ipsec ike-group NYC_MESH proposal 1 hash sha1
set vpn ipsec site-to-site peer l2tpvpn.sn1.mesh.nycmesh.net description "NYC Mesh VPN
(SN1)"
set vpn ipsec site-to-site peer l2tpvpn.sn1.mesh.nycmesh.net authentication mode pre-
shared-secret
set vpn ipsec site-to-site peer l2tpvpn.sn1.mesh.nycmesh.net authentication pre-shared-
secret nycmeshnet
set vpn ipsec site-to-site peer l2tpvpn.sn1.mesh.nycmesh.net local-address default
set vpn ipsec site-to-site peer l2tpvpn.sn1.mesh.nycmesh.net ike-group NYC_MESH
set vpn ipsec site-to-site peer l2tpvpn.sn1.mesh.nycmesh.net tunnel 1 esp-group NYC_MESH
set vpn ipsec site-to-site peer l2tpvpn.sn1.mesh.nycmesh.net tunnel 1 local port l2tp
set vpn ipsec site-to-site peer l2tpvpn.sn1.mesh.nycmesh.net tunnel 1 protocol udp
```

```
set vpn ipsec site-to-site peer l2tpvpn.sn1.mesh.nycmesh.net tunnel 1 remote port l2tp
```

Here's what your final configuration should look like. You can view it with `show interfaces l2tp-client` and `show vpn`. There are more settings here than you typed in above. That's ok. The additional settings are part of the default L2TP/IPsec config.

```
interfaces {
    l2tp-client l2tpc0 {
        authentication {
            password 'your personal password'
            user-id your personal user name'
        }
        default-route auto
        description "NYC Mesh VPN (SN1)"
        mtu 1400
        name-server auto
        require-ipsec
        server-ip l2tpvpn.sn1.mesh.nycmesh.net
    }
}
vpn {
    ipsec {
        allow-access-to-local-interface disable
        auto-firewall-nat-exclude disable
        esp-group NYC_MESH {
            compression disable
            lifetime 3600
            mode transport
            pfs disable
            proposal 1 {
                encryption aes256
                hash sha1
            }
        }
        ike-group NYC_MESH {
            dead-peer-detection {
                action restart
                interval 30
                timeout 120
            }
```

```
            ikev2-reauth no
            key-exchange ikev1
            lifetime 28800
            proposal 1 {
                dh-group 2
                encryption aes256
                hash sha1
            }
        }
        site-to-site {
            peer l2tpvpn.sn1.mesh.nycmesh.net {
                authentication {
                    mode pre-shared-secret
                    pre-shared-secret nycmeshnet
                }
                connection-type initiate
                description "NYC Mesh VPN (SN1)"
                ike-group NYC_MESH
                ikev2-reauth inherit
                local-address default
                tunnel 1 {
                    allow-nat-networks disable
                    allow-public-networks disable
                    esp-group NYC_MESH
                    local {
                        port l2tp
                    }
                    protocol udp
                    remote {
                        port l2tp
                    }
                }
            }
        }
    }
}
```

PPP configuration workaround

There is a bug in EdgeOS's PPP configuration that prevents EdgeRouter from connecting to the NYC Mesh VPN. Before you commit your VPN configuration, add the following scripts to your EdgeOS device:

The first script is located at `/config/scripts/post-config.d/post-commit-hooks.sh`. It's a helper that lets you run scripts every time you commit a new configuration.

```
#!/bin/sh

set -e

if [ ! -d /config/scripts/post-commit.d ]; then
  mkdir -p /config/scripts/post-commit.d
fi

if [ ! -L /etc/commit/post-hooks.d/post-commit-hooks.sh ]; then
  sudo ln -fs /config/scripts/post-config.d/post-commit-hooks.sh /etc/commit/post-hooks.d
fi

run-parts --report --regex '^[a-zA-Z0-9._-]+$' /config/scripts/post-commit.d
```

Make it executable and then run it:

```
chmod +x /config/scripts/post-config.d/post-commit-hooks.sh
/config/scripts/post-config.d/post-commit-hooks.sh
```

The second script fixes the PPP configuration for your L2TP tunnel so that you can successfully connect. It is located at `/config/scripts/post-commit.d/fixup-l2tpc0.sh`. **Note: this is a different directory from the previous script.**

```
#!/bin/bash

set -e

DEVICE=l2tpc0

CONFIG=/etc/ppp/peers/$DEVICE

if [ ! -f $CONFIG ]; then
  exit
fi
```

```
if grep ^remotename $CONFIG > /dev/null; then
  exit
fi

echo "remotename xl2tpd" | sudo tee -a $CONFIG > /dev/null
```

Make it executable:

```
chmod +x /config/scripts/post-commit.d/fixup-l2tpc0.sh
```

MTU workaround

Additionally, EdgeOS has a bug where pppd fails to correctly set the MTU of the L2TP interface. This is a problem if you plan to use OSPF over the VPN because OSPF requires that both peers agree on an MTU.

This script, located at `/config/scripts/ppp/ip-up.d/l2tp-fix-mtu` works around this issue by manually setting the MTU after the connection comes up.

```
#!/bin/sh

set -e

MTU=$(grep mtu /etc/ppp/peers/$PPP_IFACE | awk '{ print $2 }')

if echo $PPP_IFACE | grep -Eq ^l2tpc[0-9]+; then
  ip link set dev $PPP_IFACE mtu $MTU
fi
```

Make it executable, and commit your configuration:

```
chmod +x /config/scripts/ppp/ip-up.d/l2tp-fix-mtu
commit
```

If all goes well, you should be connected to the VPN and be able to reach the other end of the tunnel:

```
ubnt@edgerouter# run show interfaces l2tp-client
Codes: S - State, L - Link, u - Up, D - Down, A - Admin Down
Interface    IP Address                        S/L  Description
```

```
---------    ----------                          ---  -----------
l2tpc0       10.70.72.68                         u/u  NYC Mesh VPN (SN1)
ubnt@edgerouter# ping 10.70.72.1
PING 10.70.72.1 (10.70.72.1) 56(84) bytes of data.
64 bytes from 10.70.72.1: icmp_seq=1 ttl=64 time=6.15 ms
64 bytes from 10.70.72.1: icmp_seq=2 ttl=64 time=6.42 ms
64 bytes from 10.70.72.1: icmp_seq=3 ttl=64 time=4.98 ms
^C
--- 10.70.72.1 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2007ms
rtt min/avg/max/mdev = 4.985/5.854/6.424/0.627 ms
```

Additionally, the MTU of your L2TP interface should be correctly set to 1400:

```
ubnt@edgerouter# ip link show l2tpc0
34: l2tpc0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1400 qdisc pfifo_fast state
UNKNOWN mode DEFAULT group default qlen 100
    link/ppp
```

Save your active configuration to the startup configuration so that your tunnel will still be there when you reboot, and exit configuration mode:

```
save
exit
```

Reaching the mesh through the VPN

So far, your EdgeRouter can reach the VPN server at the other end of the tunnel, but you can't reach any of the other devices on the mesh (try pinging `10.10.10.10`; you shouldn't be able to reach it). You can fix this by OSPF peering or by adding a static route. A static route is easiest.

In configuration mode, enter the following, and commit it:

```
set protocols static interface-route 10.0.0.0/8 next-hop-interface l2tpc0 description "NYC
Mesh"
```

In this configuration, your EdgeRouter will route traffic destined for the mesh's private IP network through the VPN and all your other traffic over your primary internet connection – this is sometimes called split VPN. If you use addresses from `10.0.0.0/8` for your LAN that overlap with addresses used by the mesh, the addresses on your LAN will take precedence and you will not be able to access those parts of the mesh.

Once you've installed the static route, you should be able to reach the rest of the mesh:

```
ubnt@edgerouter# ping 10.10.10.10
PING 10.10.10.10 (10.10.10.10) 56(84) bytes of data.
64 bytes from 10.10.10.10: icmp_seq=1 ttl=63 time=4.85 ms
64 bytes from 10.10.10.10: icmp_seq=2 ttl=63 time=4.28 ms
64 bytes from 10.10.10.10: icmp_seq=3 ttl=63 time=7.08 ms
^C
--- 10.10.10.10 ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2006ms
rtt min/avg/max/mdev = 4.286/5.408/7.082/1.207 ms
```

Remember to save your configuration to the startup config once it's working.

Using NAT to reach the mesh from a device on your network

You can now reach the mesh from your EdgeRouter, but you can't reach it from a device on your LAN like your laptop. The easiest way to do this is to use NAT:

```
set service nat rule 6000 outbound-interface l2tpc0
set service nat rule 6000 type masquerade
set service nat rule 6000 description "masquerade for NYC Mesh (SN1)"
```

Commit your config, and verify that you can reach the mesh from your laptop:

```
me@laptop$ ping 10.10.10.10
PING 10.10.10.10 (10.10.10.10): 56 data bytes
64 bytes from 10.10.10.10: icmp_seq=0 ttl=62 time=13.572 ms
64 bytes from 10.10.10.10: icmp_seq=1 ttl=62 time=10.603 ms
64 bytes from 10.10.10.10: icmp_seq=2 ttl=62 time=16.394 ms
^C
--- 10.10.10.10 ping statistics ---
3 packets transmitted, 3 packets received, 0.0% packet loss
round-trip min/avg/max/stddev = 10.603/13.523/16.394/2.364 ms
```

Once your configuration is working, save it to your startup config.

If you can ping 10.10.10.10 from the router, but not other devices on your LAN, you may need to reboot the router to clear the route cache.

Configuring .mesh DNS lookup

To use the .mesh top level domain to reach mesh services, you will need to change the DNS configuration on your EdgeRouter. The simplest way to do this is to configure your router's DHCP server to tell clients to use the mesh's recursive resolver (`10.10.10.10`) as their DNS server. But this causes a problem with our split VPN config: if your VPN connection goes down, you won't be able to resolve domain names, even if you're still connected to the public internet.

To fix this, you can configure EdgeOS's DNS forwarder to use the mesh's authoritative name server (`10.10.10.11`) for the .mesh TLD:

```
set service dns forwarding options server=/mesh/10.10.10.11
```

Additionally, you can configure the DNS forwarder to use the mesh's name server for reverse DNS lookups on `10.0.0.0/8`:

```
set service dns forwarding options rev-server=10.0.0.0/8,10.10.10.11
```

Make sure to configure the DHCP server to provide your router's LAN address as the recursive DNS resolver.

To be able to reach the .mesh TLD while SSH'd into your EdgeRouter, configure your EdgeRouter to use its local DNS forwarder as its primary DNS server:

```
set system name-server 127.0.0.1
```

Revision #18
Created 9 December 2023 04:39:51 by Willard Nilges
Updated 9 August 2024 02:44:14 by Lydon Thorpe