Oracle Linux

Configuring Virtual Private Networks

F45242-08
January 2024

Oracle Linux Configuring Virtual Private Networks,

F45242-08

Contents

Preface

Documentation License iv
Conventions iv
Documentation Accessibility iv
Access to Oracle Support for Accessibility iv
Diversity and Inclusion v

1 About Virtual Private Networks

2 Configuring a VPN by Using WireGuard

Installing WireGuard 2-1
Configuring WireGuard 2-1
Enabling the WireGuard Tunnel 2-3
Testing the WireGuard Tunnel 2-5
Shutting Down the WireGuard Tunnel 2-6

3 Configuring a VPN by Using Libreswan

Installing Libreswan 3-1
Configuring IPsec VPN 3-1
Creating a Host to Host Connection 3-1
Creating a Site to Site Connection 3-3
Verifying the Status of VPN Services 3-4

Preface

Oracle Linux: Configuring Virtual Private Networks describes how to use virtual private networks (VPNs) in Oracle Linux to deploy tunneled connections to remote systems.

Documentation License

The content in this document is licensed under the Creative Commons Attribution–Share Alike 4.0 (CC-BY-SA) license. In accordance with CC-BY-SA, if you distribute this content or an adaptation of it, you must provide attribution to Oracle and retain the original copyright notices.

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at https://www.oracle.com/corporate/accessibility/.

For information about the accessibility of the Oracle Help Center, see the Oracle Accessibility Conformance Report at https://www.oracle.com/corporate/accessibility/templates/t2-11535.html.

Access to Oracle Support for Accessibility

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit https://www.oracle.com/corporate/accessibility/learning-support.html#support-tab.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1

About Virtual Private Networks

Note:

This document includes content that was tested against Oracle Linux 8 and Oracle Linux 9, but applies to most Oracle Linux releases, and might also apply to other distributions.

VPNs are used to enable remote access between endpoints and provide site-to-site connections that simulate a larger network beyond the limitations of a deployed physical network. Encryption is applied to secure the traffic traversing the tunneled connections.

Implementations of VPNs have varied over time through the release of different VPN protocols, technologies, and applications. Oracle Linux supports two VPN technologies: IPsec OpenVPN implementation in Libreswan and WireGuard®.

WireGuard differs from OpenVPN in that OpenVPN uses certificates for identification and encryption. WireGuard uses public key encryption for those tasks. Secure key generation and management is handled **in the** background, and an option is available to preshare a key for an added layer of security.

For more information, see the following resources:

- WireGuard: https://www.wireguard.com
- Libreswan: https://libreswan.org

2

Configuring a VPN by Using WireGuard

WireGuard is a cross-platform technology that enables you to create a VPN setup that's simple, fast, but secure through its implementation of the latest cryptography.

Note:

WireGuard is supported beginning with Unbreakable Enterprise Kernel Release 6 Update 3. Ensure that the system is upgraded to this release or later before proceeding to the steps in this chapter. For more information, see Unbreakable Enterprise Kernel Release 6 Update 3: Release Notes (5.4.17-2136).

Installing WireGuard

To configure a VPN with WireGuard, download the package with the following command:

```
sudo dnf install -y wireguard-tools
```

You must install WireGuard on the server and all its clients.

Configuring WireGuard

For simplicity, the following sections describe how to deploy WireGuard by using two hosts as examples. One host functions as the VPN server while the other is a client.

To use WireGuard, you need the following requirements:

- IP addresses of both hosts. Use the `ip addr sh` command to obtain this information. For the procedures that follow, the IP address of the server is 10.0.0.1 and the IP address of the client is 10.0.0.2.
- Private IP addresses to be assigned to the WireGuard interfaces of both hosts. For the procedures that follow, the private IP address of the server is 192.168.2.1 and the private IP address of the client is 192.168.2.2.
- Name of the WireGuard interface. For the procedures that follow, the Wireguard interface name of both server and client is `wg0`.
- Console connections to the client.

  Instructions for setting up console connections is outside the scope of this documentation. To use a VNC server for remote connections, see https://docs.oracle.com/en/learn/install-vnc-oracle-linux/#introduction. If you're using an instance in Oracle Cloud Infrastructure, see https://docs.oracle.com/iaas/Content/Compute/References/serialconsole.htm#Instance_Console_Connections.

On the server

1. Edit the `/etc/sysctl.conf` file with the following changes:

   ```
   net.ipv4.ip_forward = 1
   ```

2. Run the `sysctl` command to reread the `/etc/sysctl.conf` file.

   ```
   sudo sysctl -p
   ```

   You can disregard the error messages that might appear about the command not being able to perform `stat`.

3. Create a directory where you can store the WireGuard key pair, for example:

   ```
   mkdir ~/.wireguard
   cd ~/.wireguard
   umask 077
   ```

4. In the directory you created, generate a WireGuard cryptographic key pair.

   You can specify any preferred names for the files to contain the server's private and public keys.

   ```
   wg genkey | tee privatekey | wg pubkey > publickey
   ```

5. Obtain the server's private and public keys and store them in a temporary storage.

   ```
   cat ~/.wireguard/privatekey
   cat ~/.wireguard/publickey
   ```

On the client

1. Create a directory where you can store the WireGuard key pair, for example:

   ```
   mkdir ~/.wireguard
   cd ~/.wireguard
   umask 077
   ```

2. In the directory you created, generate a WireGuard cryptographic key pair.

   You can specify any preferred names for the files to contain the client's private and public keys.

   ```
   wg genkey | tee privatekey | wg pubkey > publickey
   ```

3. Obtain the client's private and public keys and store them in a temporary storage.

   ```
   cat ~/.wireguard/privatekey
   cat ~/.wireguard/publickey
   ```

4. Edit the `/etc/wireguard/wg0.conf` file to contain the following:

   ```
   [Interface]
   Address = 192.168.2.2/24
   SaveConfig = true
   ListenPort = 60477
   ```

```
PrivateKey = client's private key

[Peer]
PublicKey = server's public key
AllowedIPs = 0.0.0.0/0, ::/0
Endpoint = 10.0.0.1:51820
```

On the server

1. Edit the `/etc/wireguard/wg0.conf` file to contain the following:

```
[Interface]
Address = 192.168.2.1/24
SaveConfig = true
PostUp = iptables -A FORWARD -i wg0 -j ACCEPT; iptables -t nat -A
POSTROUTING -o eno1 -j MASQUERADE
PostDown = iptables -D FORWARD -i wg0 -j ACCEPT; iptables -t nat -D
POSTROUTING -o eno1 -j MASQUERADE
ListenPort = 51820
PrivateKey = server's private key
[Peer]
PublicKey = client's public key
AllowedIPs = 192.168.2.2/32
Endpoint = 10.0.0.2:60477
```

Note:

In the `PostUp` and `PostDown` lines of the example, the server's interface name is `eno1`. Ensure that you specify the correct interface name of your server for those same lines. To display the server's IP interface, run the `ip addr sh` command.

Enabling the WireGuard Tunnel

1. Start the tunnel on the server.

```
wg-quick up wg0


[#] ip link add wg0 type wireguard
[#] wg setconf wg0 /dev/fd/63
[#] ip link set mtu 1420 up dev wg0
[#] ip -4 route add 192.168.2.2/32 dev wg0
[#] iptables -A FORWARD -i wg0 -j ACCEPT; iptables -t nat -A POSTROUTING
 -o eno1 -j MASQUERADE; ip6tables -A FORWARD -i wg0 -j ACCEPT; ip6tables
-t nat -A POSTROUTING -o eno1 -j MASQUERADE
```

2. Check the status of the tunnel in the server.

   ```
   sudo wg
   ```

   ```
   interface: wg0
      public key: server's public key
      private key: (hidden)
      listening port: 51820

   peer: client public's key
     endpoint: 10.0.0.2:60477
     allowed ips: 192.168.2.2/32
   ```

3. On a separate terminal window, log in to the client by using a console connection.

   From this point on, all network traffic is routed through the tunnel. If you're logged in to the client through SSH and then issue the command to start the tunnel, that connection becomes lost. You can only reconnect to the client through a console connection.

4. Start the tunnel on the client.

   ```
   wg-quick up wg0
   ```

   ```
   [#] ip link add wg0 type wireguard
   [#] wg setconf wg0 /dev/fd/63
   [#] ip -4 address add 192.168.2.2/24 dev wg0
   [#] ip link set mtu 1420 up dev wg0
   [#] ip -6 route add ::/0 dev wg0 table 51820
   [#] ip -6 rule add not fwmark 51820 table 51820
   [#] ip -6 rule add table main suppress_prefixlength 0
   [#] ip6tables-restore -n
   [#] ip -4 route add 0.0.0.0/0 dev wg0 table 51820
   [#] ip -4 rule add not fwmark 51820 table 51820
   [#] ip -4 rule add table main suppress_prefixlength 0
   [#] sysctl -q net.ipv4.conf.all.src_valid_mark=1
   [#] iptables-restore -n
   ```

5. Check the status of the client.

   ```
   sudo wg
   ```

   ```
   interface: wg0
      public key: client's public key
      private key: (hidden)
      listening port: 60477
      fwmark: 0xca6c

   peer: server public's key
     endpoint: 10.0.0.1:51820
     allowed ips: 0.0.0.0/0, ::/0
   ```

You can use other commands to check the tunnel information, such as `ip addr sh` and `ip link`, for example:

```
ip addr sh
```

```
...
3: wg0: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 8920 qdisc noqueue state UNKNOWN
group default
   qlen 1000 link/none
   inet 192.168.2.1/24 scope global wg0
     valid_lft forever preferred_lft forever
```

```
ip link
```

```
...
3: wg0: <POINTOPOINT,NOARP,UP,LOWER_UP> mtu 8920 qdisc noqueue state UNKNOWN
mode DEFAULT group
   default qlen 1000 link/none
```

Testing the WireGuard Tunnel

On the client, use the `ping` command to check communications with the WireGuard server, for example:

```
ping -c 3 192.168.2.1
```

```
ping -c 3 192.168.2.1

PING 192.168.2.1 (192.168.2.1) 56(84) bytes of data.
64 bytes from 192.168.2.1: icmp_seq=1 ttl=64 time=0.640 ms
64 bytes from 192.168.2.1: icmp_seq=2 ttl=64 time=0.707 ms
64 bytes from 192.168.2.1: icmp_seq=4 ttl=64 time=0.643 ms

--- 192.168.2.1 ping statistics ---
3 packets transmitted, 4 received, 0% packet loss, time 2088ms
rtt min/avg/max/mdev = 0.643/0.690/0.713/0.038 ms
```

Likewise, the `wg` command indicates whether a connection to the WireGuard tunnel is established, as shown in the following example:

```
sudo wg
```

```
interface: wg0
  public key: client's public key
  private key: (hidden)
  listening port: 60477
  fwmark: 0xca6c
```

```
peer: server's public key
  endpoint: 10.0.0.1:51820
  allowed ips: 0.0.0.0/0, ::/0
  latest handshake: 44 seconds ago
  transfer: 28.22 KiB received, 48.64 KiB sent
```

The `handshake` and `transfer` output show the traffic in the tunneled connection.

If communications between the server and the client can't be established, check if the firewall might be blocking access to the ports that have been set up for WireGuard. You might need to open the ports to link both server and client. For example, you would type the following commands on the server:

```
sudo firewall-cmd --add-port=51820/udp --permanent
sudo firewall-cmd --reload
```

Shutting Down the WireGuard Tunnel

To shut down the WireGuard tunnel, type the following command on both server and client.

```
wg-quick down wireguard-interface
```

After shutting down, you can optionally check the status of the tunnels by using one of the commands that have been provided elsewhere in this document, such as `wg`, `ip addr`, or `ip link`.

3
Configuring a VPN by Using Libreswan

Libreswan is the software that implements VPN by using the IPsec protocol and the Internet Key Exchange (IKE) standards.

Installing Libreswan

To configure an IPsec VPN with Libreswan, download the package as follows:

1. Ensure that the AppStream repository is enabled.
2. Install Libreswan.

   ```
   sudo dnf install -y libreswan
   ```

3. Start `ipsec` as a persistent service.

   ```
   sudo systemctl enable ipsec --now
   ```

4. Add the `ipsec` service to the firewall service.

   ```
   sudo firewall-cmd --add-service="ipsec"
   sudo firewall-cmd --runtime-to-permanent
   ```

Configuring IPsec VPN

VPN configurations range from basic setups such as one between hosts to complex ones that involve entire sites.

Creating a Host to Host Connection

Regardless of the types of VPN connections that you want to configure, a common but important step involves obtaining RSA keys that would enable connections between endpoints.

On a host-to-host connection, for example, do the following:

1. Generate an RSA key pair by running the following command.

   ```
   sudo ipsec newhostkey
   ```

   ```
   Generated RSA key pair with CKAID
   6e6e724aa180b071128632dc09c7d2b25a852d7e was stored in the NSS database
   ```

   The command generates an RSA key pair with a specific `ckaid` value.

   You must run the command on **both** hosts.

2. On the first host, display the `leftrsasigkey` key.

   > **Note:**
   >
   > Typically in `libreswan`, the *left* designation refers to the local host, while the *right* designation refers to the remote host. However, because both hosts are peers, the designations can be used interchangeably.

   ```
   sudo ipsec showhostkey --list
   ```

   From the output, identify the `ckaid`, for example, `6e6e724aa180b071128632dc09c7d2b25a852d7e` and use it to display the `leftrsasigkey` key, as follows:

   ```
   sudo ipsec showhostkey --left --ckaid
   6e6e724aa180b071128632dc09c7d2b25a852d7e


     # rsakey AwEAAaxdf

   leftrsasigkey=0sAwEAAaxdfaCPrZ72pAm1kjvhAQHHLn3Wg3gAu1Z0U+3FWeh7FN+b
   Htfy
   ...
   9f8=
   ```

3. On the second host, display the `rightrsasigkey` key.

   ```
   sudo ipsec showhostkey --list
   ```

   From the output, identify the `ckaid`, for example, `5dddc2334515702c3a605bc00daed1e44e18767d` and use it to display the `rightrsasigkey` key, as follows:

   ```
   ipsec showhostkey --right --ckaid
   5dddc2334515702c3a605bc00daed1e44e18767d


     # rsakey AwEAAb1nC

   rightrsasigkey=0sAwEAAdSSYrNO2QOY8RXgLlJZilBokPb9cFzCbU+VYY7eFcoZMmV
   WPVI
   ...
   zu+/7BE5kjXHAAI1fvYha+CFbuh6KYAlpoHvX81ALusfQs+6wwTsde5jlfcrXNlqX
   ```

4. On each host, create a configuration file in `/etc/ipsec.d`, for example, `host2host.conf`, with the following entries:

   ```
   conn tunnel-name
       leftid=@host1-tunnel-id
       left=host1-IPaddress
   ```

```
leftrsasigkey=host1-leftrsasigkey
rightid=@host2-tunnel-id
right=host2-IPaddress
rightrsasigkey=host2-rightrsasigkey
authby=rsasig
```

For more information about the configuration file and other parameters you can set, see the `ipsec.conf(5)` manual page.

5. Restart the IPsec service.

   ```
   sudo systemctl restart ipsec
   ```

6. Start `libreswan`.

   ```
   sudo ipsec setup start
   ```

7. Load the VPN tunnel connection.

   ```
   sudo ipsec auto --add tunnel-name
   ```

8. Establish the tunnel connection.

   ```
   sudo ipsec auto --up tunnel-name
   ```

9. Start the tunnel automatically when the `ipsec` service is started by adding the following line to the configuration file:

   ```
   auto=start
   ```

Creating a Site to Site Connection

A VPN connection between sites means that a connection is established between two networks. When you configure a pair of hosts for this type of connection, the hosts effectively become gateways through which traffic can enter or exit to access other hosts in the network.

To configure a site to site VPN, a configured host to host VPN must already be existing and operational as described in Creating a Host to Host Connection.

Note:

You can configure a Site-to-Site VPN between your on-premises network and an Oracle Cloud Infrastructure virtual cloud network (VCN) using Libreswan. Steps for setting up a Site-to-Site VPN to OCI are described in the Libreswan Oracle Cloud Infrastructure documentation.

To proceed with configuring a connection between sites, follow these steps:

1. Create a copy of the host to host configuration file to serve as the configuration file for the site to site connection, for example:

   ```
   sudo cp /etc/ipsec.d/host2host.conf /etc/ipsec.d/site2site.conf
   ```

   Copies must exist on both hosts.
2. Edit the new configuration file by adding subnet information, for example:

   ```
   conn subnet-name
     also=tunnel-name
     leftsubnet=subnet1-IP
     rightsubnet=subnet2-IP
     auto=start

   host connection information...
   ```

   > **Note:**
   >
   > The subnets can be in CIDR notation.

Verifying the Status of VPN Services

To check if the `ipsec` service is running, type this command:

```
sudo systemctl status ipsec
```

```
ipsec.service - Internet Key Exchange (IKE) Protocol Daemon for IPsec
   Loaded: loaded (/usr/lib/systemd/system/ipsec.service; enabled;
vendor prese>
   Active: active (running) since Mon 2021-04-26 02:27:39 PDT; 7h ago
     Docs: man:ipsec(8)
           man:pluto(8)
           man:ipsec.conf(5)
...
```

To check the correctness of the `ipsec` configuration, type this command:

```
sudo ipsec verify
```

```
Verifying installed system and configuration files

Version check and ipsec on-path                         [OK]
Libreswan 3.32 (netkey) on 5.4.17-2036.104.5.el8uek.x86_64
Checking for IPsec support in kernel                    [OK]
 NETKEY: Testing XFRM related proc values
         ICMP default/send_redirects                    [NOT DISABLED]
```

```
  Disable /proc/sys/net/ipv4/conf/*/send_redirects or XFRM/NETKEY will act
on or
 cause sending of bogus ICMP redirects!

         ICMP default/accept_redirects                  [NOT DISABLED]

  Disable /proc/sys/net/ipv4/conf/*/accept_redirects or XFRM/NETKEY will act
on
or cause sending of bogus ICMP redirects!

         XFRM larval drop                               [OK]
Pluto ipsec.conf syntax                                 [OK]
Checking rp_filter                                      [ENABLED]
 /proc/sys/net/ipv4/conf/all/rp_filter                  [ENABLED]
  rp_filter is not fully aware of IPsec and should be disabled
Checking that pluto is running                          [OK]
 Pluto listening for IKE on udp 500                     [OK]
 Pluto listening for IKE/NAT-T on udp 4500              [OK]
 Pluto ipsec.secret syntax                              [OK]
Checking 'ip' command                                   [OK]
Checking 'iptables' command                             [OK]
Checking 'prelink' command does not interfere with FIPS [OK]
Checking for obsolete ipsec.conf options                [OK]
```

To test the tunnel connections, install the `tcpdump` utility to monitor network traffic.

Run the following command on one of the peers to monitor traffic explicitly on the *interface*. The utility tracks Encapsulated Security Payload (ESP) packets and traffic traversing the UDP ports 500 and 4500 that are used by the ipsec service:

```
tcpdump -n -i interface esp or udp port 500 or udp port 4500
```

```
dropped privs to tcpdump
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on interface, link-type EN10MB (Ethernet), capture size 262144
bytes
10:05:53.578884 IP 10.147.25.195 > 10.147.25.196:
ESP(spi=0xcba1dd78,seq=0x2325), length 96
10:05:53.579353 IP 10.147.25.196 > 10.147.25.195:
ESP(spi=0x979dcdbe,seq=0x2325), length 124
10:05:56.585128 IP 10.147.25.195 > 10.147.25.196:
ESP(spi=0xcba1dd78,seq=0x2326), length 96
10:05:56.585527 IP 10.147.25.196 > 10.147.25.195:
ESP(spi=0x979dcdbe,seq=0x2326), length 124
...
```

The utility first reports traffic that's generated by the peers exchanging keys.

While the `tcpdump` is running, go to the other peer and perform a network operation, such as a network `ping`, to the first host. The host that's monitoring the traffic would report network activity over the VPN from the second peer.

Press Ctrl+c to end the operations on both peers.