Setting up VPN on Raspberry Pi 4

Written by Daniel Horan

05/25/2023

Step 1: Download OS for your Raspberry Pi 4
*I have a basic Raspberry Pi OS from https://www.raspberrypi.com/software/

Step 2: Make sure you have a stable internet connection (preferably a direct ethernet connection)

Step 3: Set firewall on your Raspberry Pi:

a) Update the system through terminal:
sudo apt-get update
sudo apt-get upgrade

b) Install iptables:
sudo apt-get install iptables

1) Configure iptables:
You will now set up rules for iptables. The three default chains are INPUT (for incoming traffic), OUTPUT (for outgoing traffic), and FORWARD (for forwarded traffic). Here are the rules you might set up
sudo apt-get install iptables
*iptables is a user-space utility program that allows a system administrator to configure the IP packet filter rules of the Linux kernel firewall, implemented as different Netfilter modules. The filters are organized in different tables, which contain chains of rules for how to treat network traffic packets.
*nftables is a subsystem of the Linux kernel providing filtering and classification of network packets/datagrams/frames. It has been designed to replace the existing iptables, ip6tables, arptables, and ebtables
(During this step I had errors "***update-alternatives: error: alternative /usr/sbin/arptables-legacy for adorable not registered; not setting***" and "***update-alternatives: error: alternative /usr/sbin/ebtables-legacy for adorable not registered; not setting***". This was fixed by installing **arptables** and **ebtables**:
sudo apt-get install arptables
sudo apt-get install ebtables)

2) Allow Incoming VPN connections:
This rule allows incoming connections that are either starting a new connection or part of an already established connection, to UDP port 1194
sudo iptables -A INPUT -p udp --dport 1194 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
(During this step I had an error "***iptables/1.8.7 Failed to initialize nft: Protocol not supported***" which I was able to fix by switching iptables from "nft" to "legacy":
sudo update-alternatives --set iptables /usr/sbin/iptables-legacy
sudo update-alternatives --set ip6tables /usr/sbin/ip6tables-legacy
sudo update-alternatives --set arptables /usr/sbin/arptables-legacy

sudo update-alternatives --set ebtables /usr/sbin/ebtables-legacy)

3) Allow outgoing VPN responses:
   This rule allows all outgoing traffic. It doesn't specify a protocol or port, so it applies to all outgoing packets
   sudo iptables -A OUTPUT -p udp --sport 1194 -m conntrack --ctstate ESTABLISHED -j ACCEPT
4) Allow all outgoing traffic:
   This rule allows all outgoing traffic. It doesn't specify a protocol or port, so it applies to all outgoing packets
   sudo iptables -A OUTPUT -j ACCEPT
5) Deny all incoming traffic:
   This rule drops all incoming traffic that doesn't match any of the previous rules. Because it's appended to the end of the INPUT chain, it only applies if no previous rule has matched a packet
   sudo iptables -A INPUT -j DROP

c) Save your iptable rules:
   1) Install the iptables-persistent package (press 'yes' when asks):
      sudo apt-get install iptables-persistent
   2) if you make any changes to your iptables, you can always save them with this command:
      sudo netfilter-persistent save

d) Check your iptable rules:
   sudo iptables -L

Step 4: Install VPN software:
*I was somewhat familiar only with OpenVPN and WireGuard. For this project, I decided to choose WireGuard for these reasons:

Security: WireGuard is designed to be simple and easy to audit, which can be a significant advantage from a security perspective. It uses modern, state-of-the-art cryptographic protocols (e.g., Curve25519 for key exchange, ChaCha20 for encryption, Poly1305 for data authentication, and BLAKE2 for hashing).

Simplicity: WireGuard aims to be much simpler than OpenVPN, with less code and fewer options. This makes it easier to set up and manage, and less likely to have security vulnerabilities. However, its simplicity also means it has fewer features and less flexibility than OpenVPN.

Performance: WireGuard tends to be faster and more efficient than OpenVPN, thanks to its modern cryptographic algorithms and streamlined design.

Newer and less widely adopted: WireGuard is newer and less widely adopted than OpenVPN. While it is quickly gaining in popularity, it is not as compatible with older devices and systems

a) Install software:
   sudo apt-get update
   sudo apt-get install wireguard

b) Generate Crypto Keys:
WireGuard uses public key cryptography for secure communication. You'll need to generate a pair of keys (one private and one public) for each device that will connect to your VPN
umask 077
wg genkey | tee privatekey | wg pubkey > publickey
(During this step I got a few errors such as
"**W: Failed to fetch http://deb.debian.org/debian/dists/bullseye/InRelease Temporary failure resolving 'deb.debian.org'**";
"***W: Failed to fetch http://security.debian.org/debian-security/dists/bullseye-security/InRelease Temporary failure resolving 'security.debian.org'***";
"**W: Failed to fetch http://archive.raspberrypi.org/debian/dists/bullseye/InRelease Temporary failure resolving 'archive.raspberrypi.org'**"
To solve this problem:
i) Open text editor with command: ***sudo nano /etc/resolv.conf***
ii) Add lines:
nameserver 8.8.8.8
nameserver 8.8.4.4
These lines tell your system to use Google's public DNS servers.
iii) '***Ctrl+O***' to save the file, then **'Enter'** to confirm the filename, and finally ***'Ctrl+X***' to exit nano.
iv) Reboot the system: ***sudo reboot***
Before doing these steps, check your connection because steps i-iii might be unnecessary
Might be helpful: Check the network configuration: If you have a more complex network setup (for example, if you're using static IP addresses or if you have multiple network interfaces), there might be an issue with your network configuration. You can do it by running: ***ip addr show*
This command will display information about your network interfaces and their current configuration. Look for any anomalies, such as missing or incorrect IP addresses, subnet masks, or gateways.**

c) Configure WireGuard:
    1) Generate WireGuard Keys:
        a) Generate a private key:
        wg genkey
        Save the private key to a separate file
        b) Generate a public key:
        echo privatekey | wg pubkey
        *Replace '***privatekey***' in this command with your pregenerated private key. Save the public key to a separate file as well
    2) Configure WireGuard Interface:
        a) Open a text editor:
        sudo nano /etc/wireguard/wg0.conf
        b) Add these lines to your file:
        [Interface]
        Address = 10.200.200.1/24

SaveConfig = true
PostUp = iptables -A FORWARD -i %i -j ACCEPT; iptables -A FORWARD -o %i -j ACCEPT; iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
PostDown = iptables -D FORWARD -i %i -j ACCEPT; iptables -D FORWARD -o %i -j ACCEPT; iptables -t nat -D POSTROUTING -o eth0 -j MASQUERADE
ListenPort = 51820
PrivateKey = YourRaspberryPiPrivateKey
*Replace '***YourRaspberryPiPrivateKey***' with pregenerated private key*
These settings configure the WireGuard VPN to pass traffic from the VPN to your Raspberry Pi's internet connection. The Address line specifies the IP address range for the VPN. ListenPort is the port where your Raspberry Pi will listen for connections.

c) Save and close the text editor with commands ***'Ctrl+O***', then ***'Ctrl+X'***

3) Enable and start WireGuard Interface:
   a) Enable the WireGuard interface to start on boot:
      sudo systemctl enable wg-quick@wg0
   b) Start the WireGuard interface:
      sudo systemctl start wg-quick@wg0
      *If you will get any errors at these steps, you can run commands:
      sudo systemctl status wg-quick@wg0.service
      (This will display the status of the wg-quick@wg0 service, including any error messages that can help identify why the service failed to start. If you can, please provide the output of this command, as it will help me understand the problem and provide a more accurate solution.)
      sudo journalctl -xe
      (This will provide a detailed system log that might include more information about the error.)

** To check the status of the WireGuard: ***sudo wg show****

4) Configure Client:
   a) Install WireGuard on the client machine:
      This can be done from the official WireGuard website:
      https://www.wireguard.com/install/
   b) Generate Keys for the Client:
      wg genkey
      This will generate a private key for the client
      echo privatekey | wg pubkey
      This will generate a public key for the client
   c) echo privatekey | wg pubkey
   d) Create a Configuration File for the Client:
      *Create a separate file on your Raspberry Pi. This file should be saved as ***'name.conf'*** where ***name*** you choose yourself. In that file add these lines:
      [Interface]
      Address = 10.200.200.2/32

PrivateKey = YourClientPrivateKey

[Peer]
PublicKey = YourRaspberryPiPublicKey
Endpoint = YourRaspberryPiPublicIP:51820
AllowedIPs = 0.0.0.0/0
Where '***YourClientPrivateKey***' is replaced with your pre generated clients private key, '***YourRaspberryPiPublicKey***' is replaced with your pre generated clients public key, '***YourRaspberryPiPublicIP***' is replaced with your public IP (can use this website https://www.whatismyip.com/ or command ***curl icanhazip.com***). The **AllowedIPs** line tells WireGuard to send all network traffic through the VPN. The **0.0.0.0/0** value means that all the client's internet traffic will be sent over the VPN. If you only want to use the VPN for accessing specific networks, you would list their IP ranges here instead.*

e) Add the Client to the Raspberry Pi Configuration:
i) Open the Raspberry Pi Configuration File for Editing:
sudo nano /etc/wireguard/wg0.conf
ii) Add the Client Information:
At the end of the file add this lines:
[Peer]
PublicKey = YourClientPublicKey
AllowedIPs = 10.200.200.2/32
Where '***YourClientPublicKey***' is replaced with your pre generated clients private key. The **AllowedIPs** line should contain the IP address you assigned to the client in the client configuration file.
iii) Save and close the text editor with commands ***'Ctrl+O***', then ***'Ctrl+X***'
iv) Restart WireGuard:
sudo systemctl restart wg-quick@wg0

f) Start WireGuard:
sudo systemctl start wg-quick@wg0

Step 5: Setting up WireGuard app:

a) Import the Client Configuration:
Open the WireGuard application on your client device and import the client configuration file that you created earlier. This is usually done by clicking "Add Tunnel" or "Import Tunnel from File" in the WireGuard application and then selecting your .conf file.

b) Connect to the VPN:
In the WireGuard application on your client device, click "Activate" or "Connect" for the tunnel you just added.

c) Check the Connection:
You can check the connection status in the WireGuard application. If the VPN is working correctly, it should say that the tunnel is active.

d) Test the VPN:
To make sure the VPN is working as expected, you can try accessing the internet or a service on your local network. For a more thorough test, you can check your public

IP address to see if it matches the IP address of your Raspberry Pi (or the external IP address of your network, if you're connecting from outside). Please note that the IP address should only match if you've configured your VPN to route all traffic through the VPN (i.e., if you've set AllowedIPs = 0.0.0.0/0 in your client configuration).

Helpful commands:
sudo systemctl stop wg-quick@wg0 - safely stops WireGuard service; can safely unplug your Raspberry Pi if needed

sudo systemctl disable wg-quick@wg0 - disables the service from starting at boot

sudo systemctl enable wg-quick@wg0- re-enables it so it starts at boot

sudo shutdown -h now- safely shuts down your Raspberry Pi