By: Alexander Ho

Professor: Khanh Dinh

12/3/2021

VPN Project Report

In this project, the solution I found was to setup a full-tunnel VPN connection to my home router by using a Raspberry Pi and installing an open-source software called WireGuard. A Raspberry Pi is a low-cost small computer that can be connected to a monitor and used as a normal computer. WireGuard provides strong encryption by using a process called crypto key routing. Crypto key routing is a mechanism that associates public encryption keys with a list of VPN IP tunnel addresses which are allowed inside the tunnel. For this project, the Raspberry Pi model I used was a Raspberry Pi 3 B+. When installing a VPN designed for Raspberry Pi, the software I used is called PiVPN. PiVPN is an open-source software that sets up a VPN server using the OpenVPN and WireGuard VPN server software. When using the two VPN software, the configuration makes the installation process easy and simple to install.

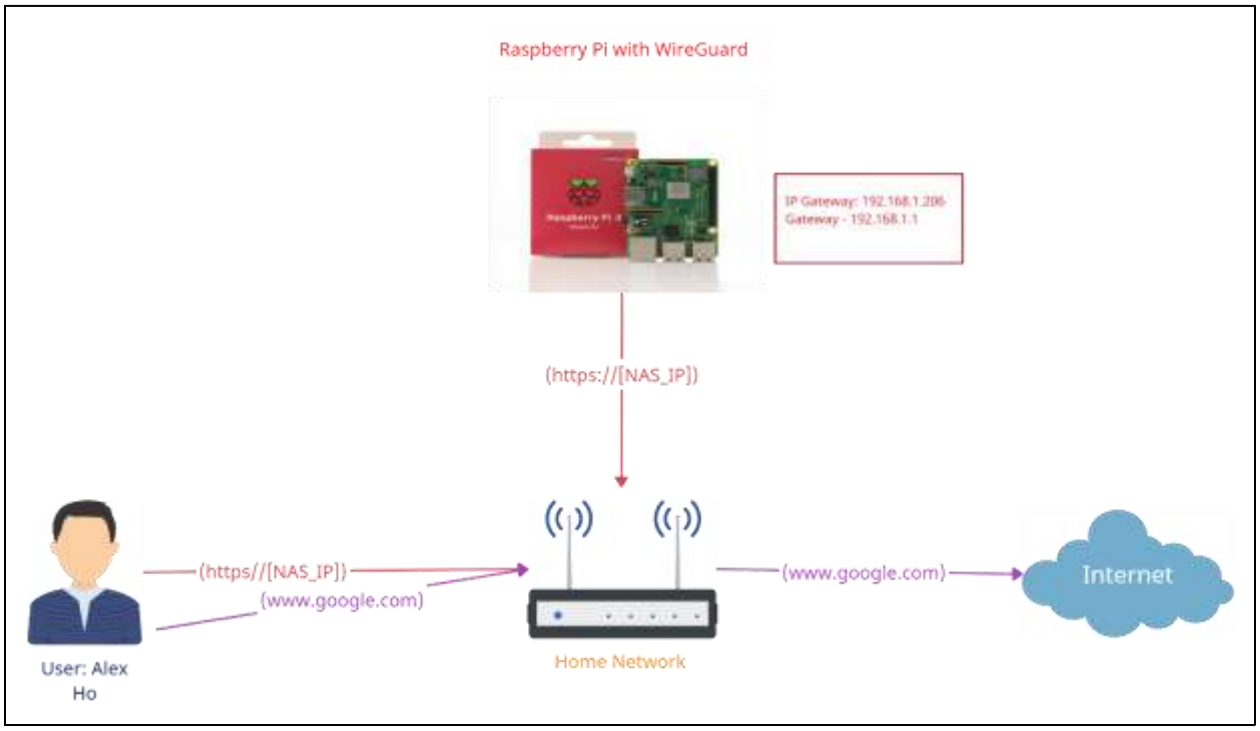
When setting up PiVPN, I only needed to execute the script “curl -L https://install.pivpn.io | bash” within the terminal of the raspberry pi. During the setup, I was asked to set up either a DHCP reservation or static IP address for the home VPN. I chose to use the DHCP reservation IP address that my router was automatically assigned. Next, I was prompted to select a local user but there is only one user on the Raspberry Pi which is the default pi user. During the installation, I was asked whether I wanted to install WireGuard or OpenVPN. I chose WireGuard for this configuration. Within the configuration, it prompted me to enter a port number, but I kept the default port number the same. The port number I used was port 51820 using a UDP protocol. Next, I was prompted to choose a DNS provider for the VPN client, and I selected Google. Lastly, I was prompted to set up whether to use my public IP address or DNS entry. I chose to use the public IP address assigned to my home router. I believe when selecting a DNS entry will make the VPN connection of WireGuard a split tunnel connection. At the end of the configuration, I was prompted to reboot the Raspberry Pi to ensure there were no issues with the installation of WireGuard.

Finally, WireGuard has successfully been installed, I created a VPN profile. When creating a VPN profile, I executed the command “sudo pivpn add alex”. The VPN profile will generate client keys, config file and update the server config file. After creating the profile, the most secure and safest way to access WireGuard on a different device such as a laptop and desktop computer is to put the generated key file on a USB flash drive. The generated key file is “alex.conf” and it cannot be shared with anyone because the “.conf” file is the key to the WireGuard. The WireGuard open-source software also has an application for IOS and Android so that any mobile device can use Wire Guard’s open VPN. Before I start installing WireGuard on my devices and connect to the VPN the most important step is to set up a port forwarding rule. The internet provider for my home network is Verizon Fios and the model number is G3100. Within my home router control panel, to configure a port forwarding rule I needed to navigate to the advanced tab in the header of the control panel. The port forwarding configuration is within the firewall section of my home router control panel. I configured a port forward rule using UDP protocol and port number “51820” that I used before when setting up the configuration of WireGuard.

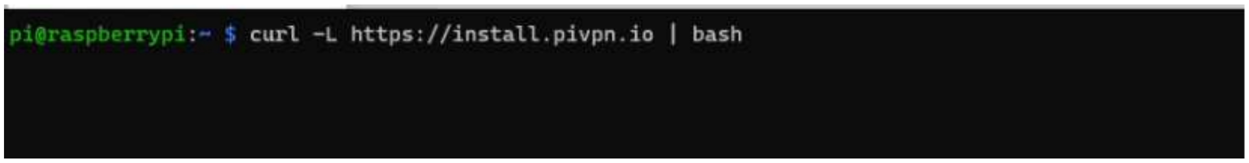
Within the open terminal, I executed a QR command to generate a QR code which makes it very secure and easy to get access onto your phone. I completed the port forwarding configuration for my router, so I tried getting a VPN connection to my home router using WireGuard on my phone. I downloaded the application and scanned the QR code from the open terminal and was able to get the key to the VPN of the user “alex” that I have successfully created. After the installation, I connected my phone to a hotspot and connected to the VPN. I successfully created a VPN using Raspberry Pi and was able to do a full tunnel VPN connection to my home router. I also installed the WireGuard software on my laptop and used a flash drive to get the key, “alex.conf” file and was able to get a full-tunnel connection to my home VPN too.

In conclusion, the solution that I found when creating a VPN that can connect to my home router is called a full tunnel connection. I used a Raspberry Pi as the host computer to set up the VPN using the WireGuard VPN software. I configured the WireGuard VPN and simply set up the software on my devices such as my phone, laptop, and desktop computer. Overall, I also did encounter some challenges such as configuring the “alex.conf” file and port forwarding the Raspberry pi host to my router.

<u>Topology of the full tunnel connection using Raspberry Pi with WireGuard installed.</u>

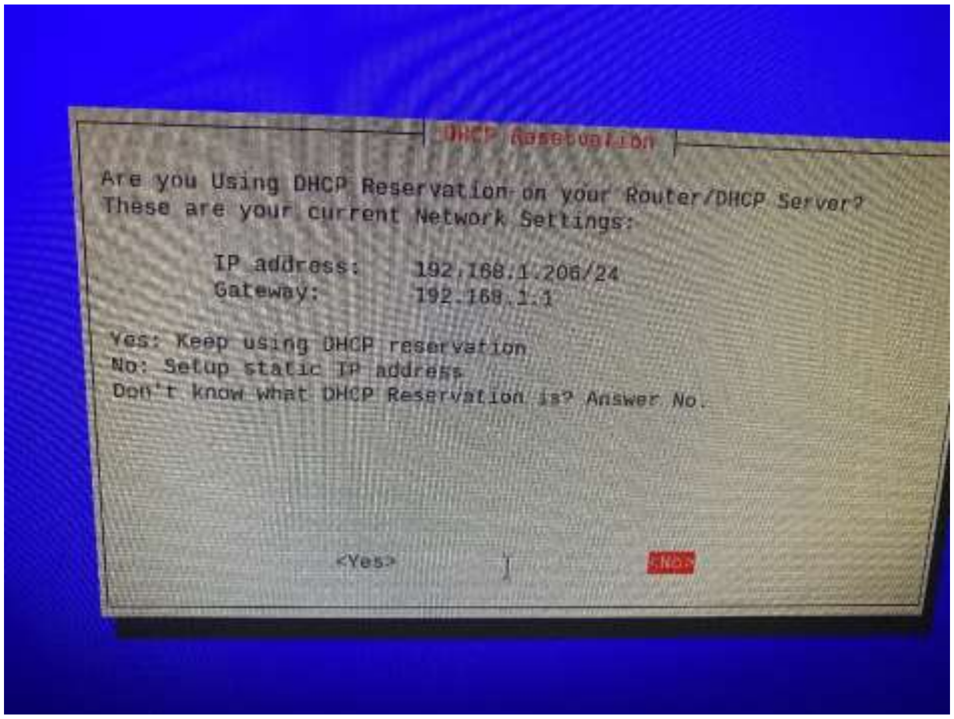


<u>Screenshot of executing a script to install PiVPN within the open terminal on my Raspberry Pi</u>



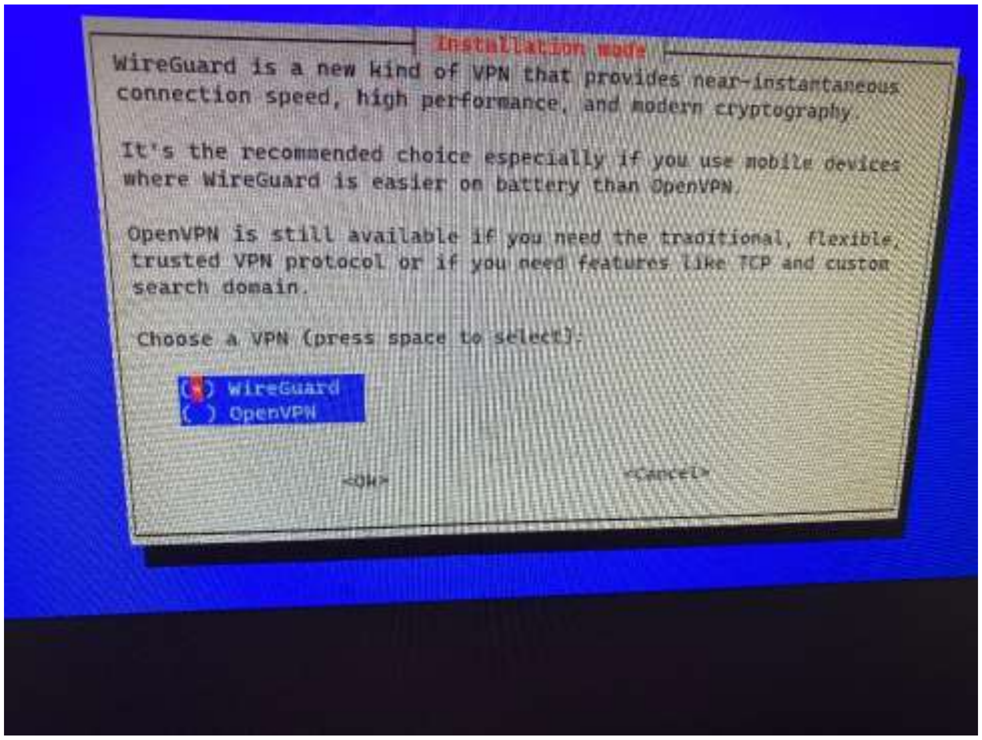
```
pi@raspberrypi:~ $ curl -L https://install.pivpn.io | bash
```

Screenshot of the Raspberry Pi as a DHCP Reservation



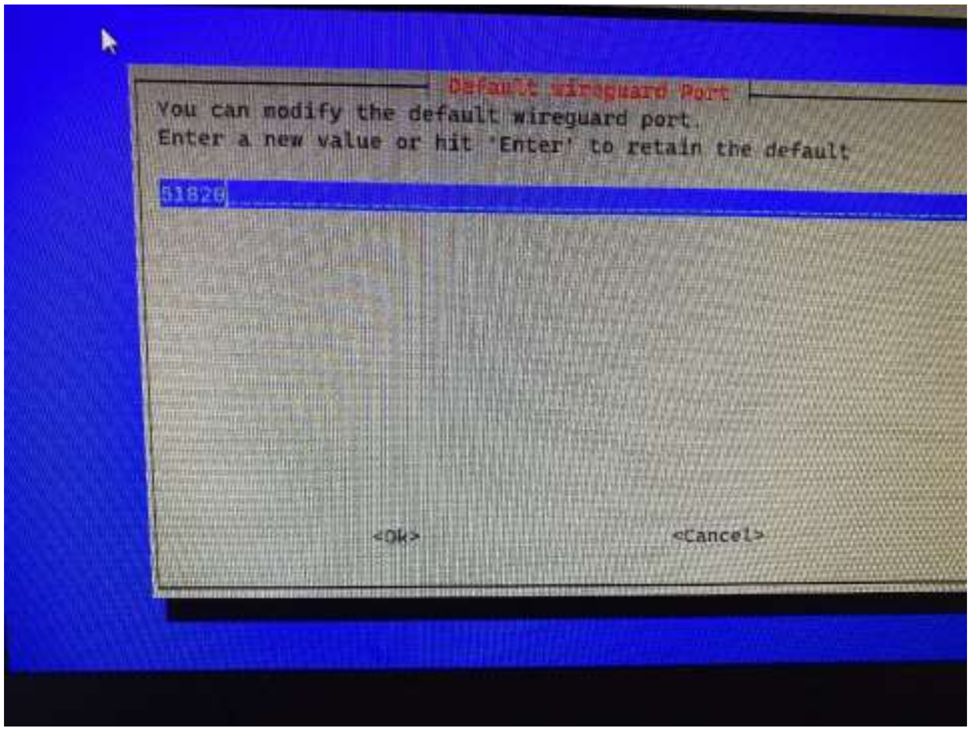
```
DHCP Reservation
Are you Using DHCP Reservation on your Router/DHCP Server?
These are your current Network Settings:

        IP address:    192.168.1.206/24
        Gateway:       192.168.1.1

Yes: Keep using DHCP reservation
No: Setup static IP address
Don't know what DHCP Reservation is? Answer No.

              <Yes>                    <No>
```

Screenshot of choosing WireGuard as the VPN server

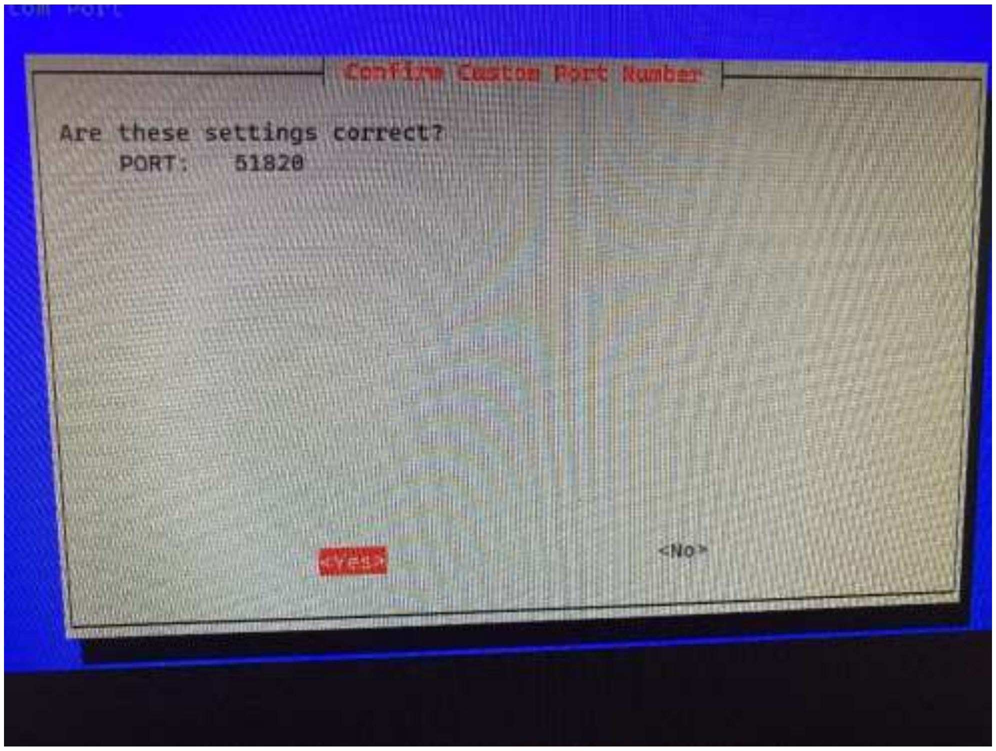


```
Installation mode
WireGuard is a new kind of VPN that provides near-instantaneous
connection speed, high performance, and modern cryptography.

It's the recommended choice especially if you use mobile devices
where WireGuard is easier on battery than OpenVPN.

OpenVPN is still available if you need the traditional, flexible,
trusted VPN protocol or if you need features like TCP and custom
search domain.

Choose a VPN (press space to select):

     (*) WireGuard
     ( ) OpenVPN

            <Ok>                    <Cancel>
```

Screenshot of using the default WireGuard port 51820

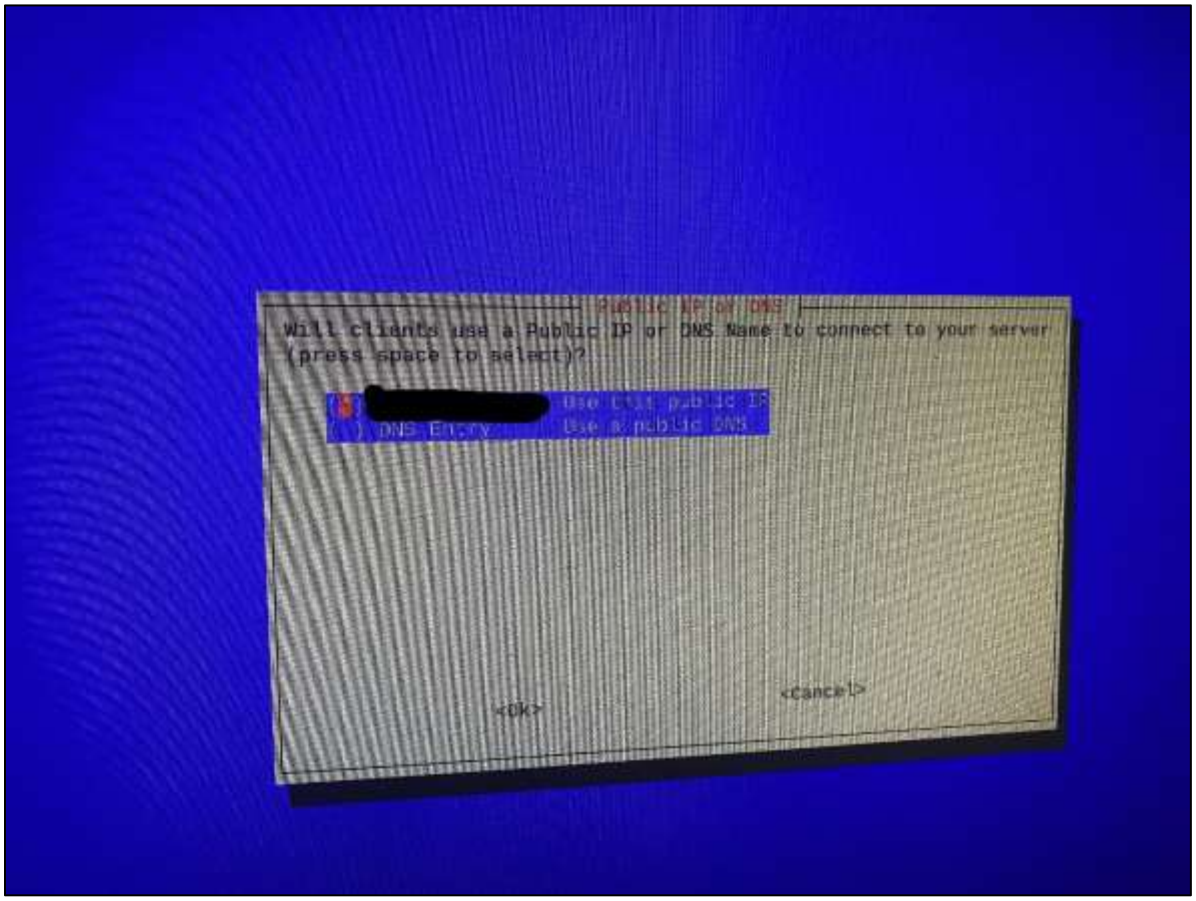


```
Default wireguard Port
You can modify the default wireguard port.
Enter a new value or hit 'Enter' to retain the default

51820
```



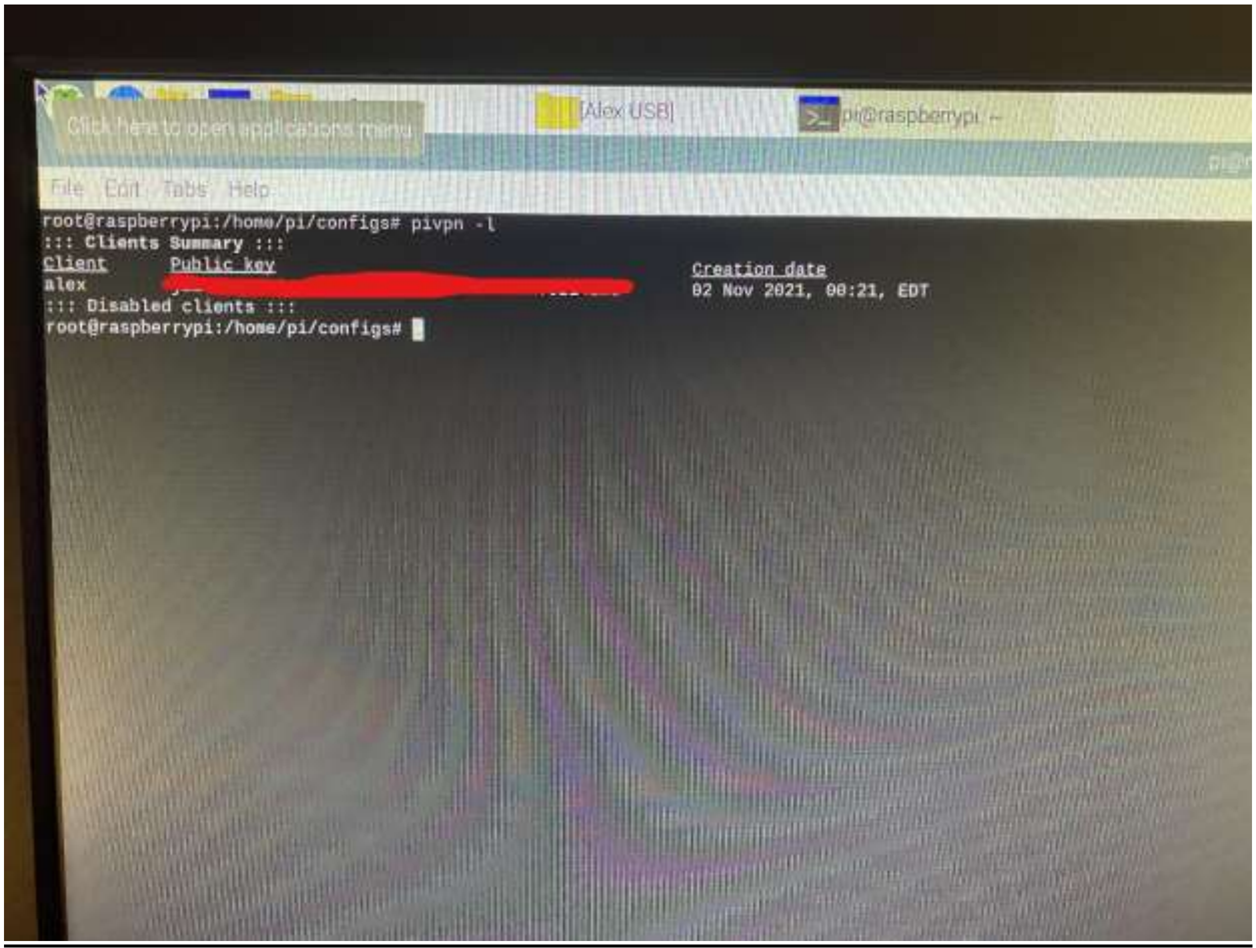
```
Confirm Custom Port Number

Are these settings correct?
    PORT:   51820
```

<u>Screenshot of using my public IP address which will create a full tunnel connection to my home router.</u>



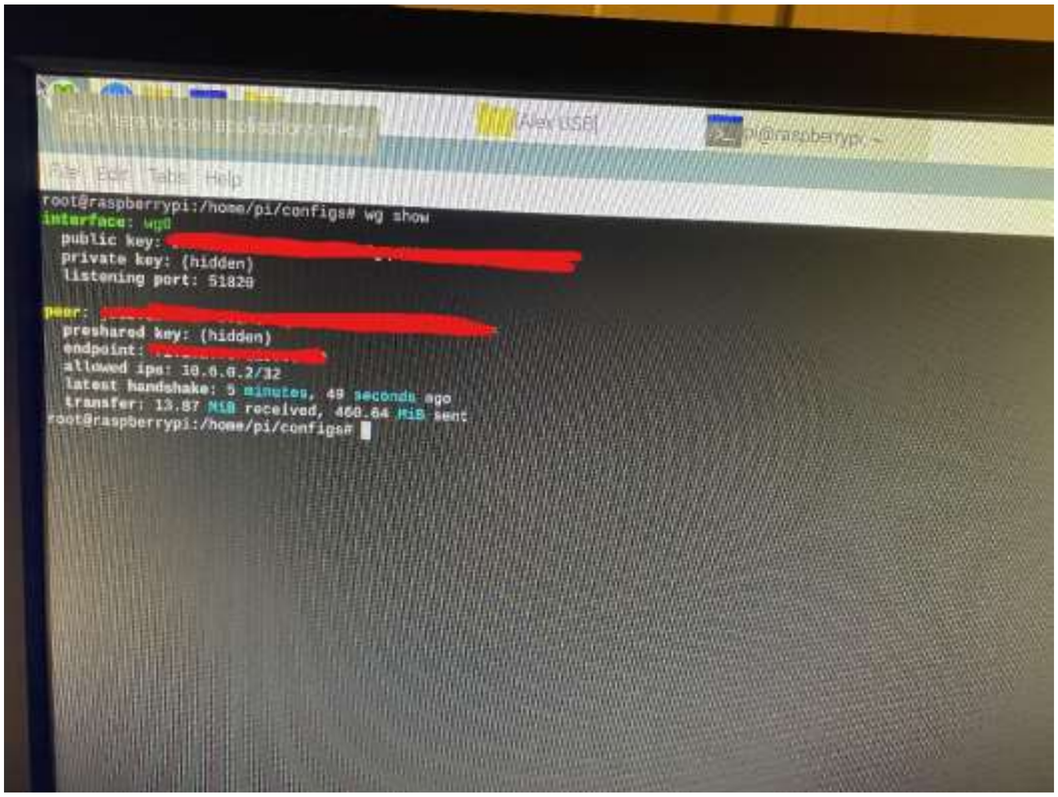
```
Public IP or DNS
Will clients use a Public IP or DNS Name to connect to your server
(press space to select)?

  (*) [redacted]        Use this public IP
  ( ) DNS Entry         Use a public DNS

            <Ok>                    <Cancel>
```

<u>Screenshot of using a PiVPN command to show the “alex” pi user creation date</u>



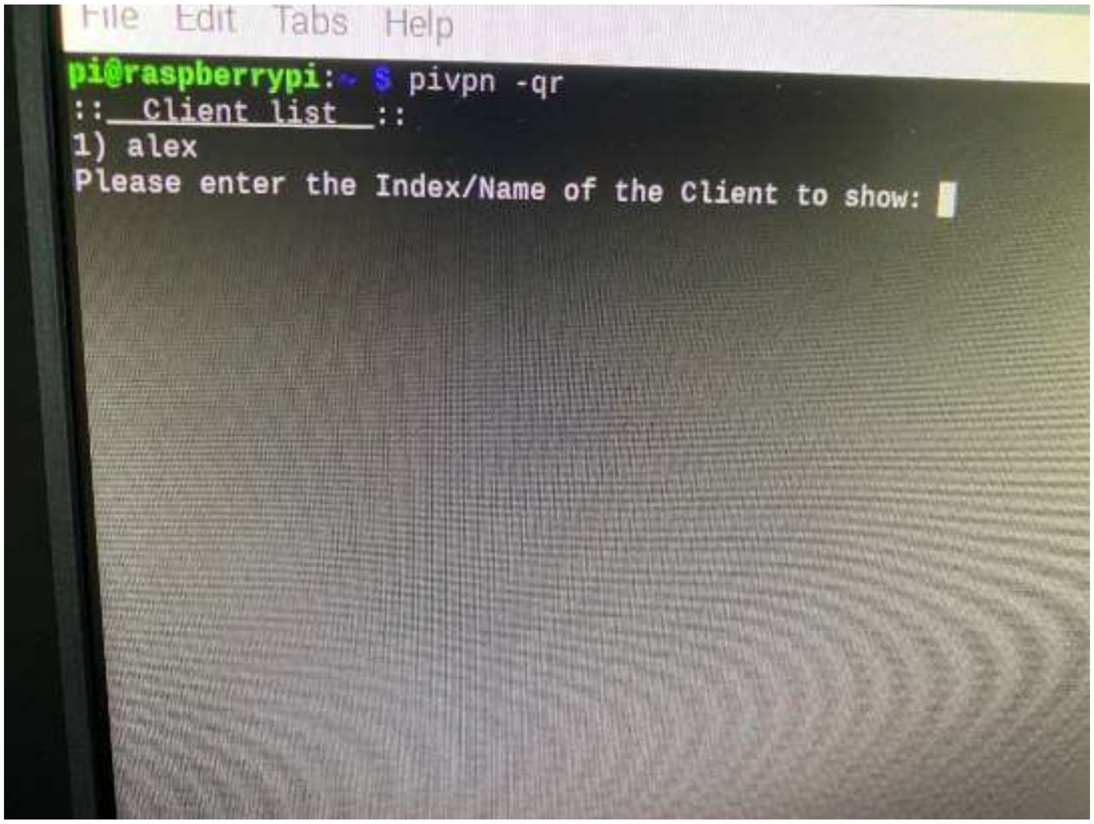
```
root@raspberrypi:/home/pi/configs# pivpn -l
::: Clients Summary :::
Client      Public key                                    Creation date
alex        [redacted]                                    02 Nov 2021, 00:21, EDT
::: Disabled clients :::
root@raspberrypi:/home/pi/configs#
```

Screenshot of WireGuard can show a connection



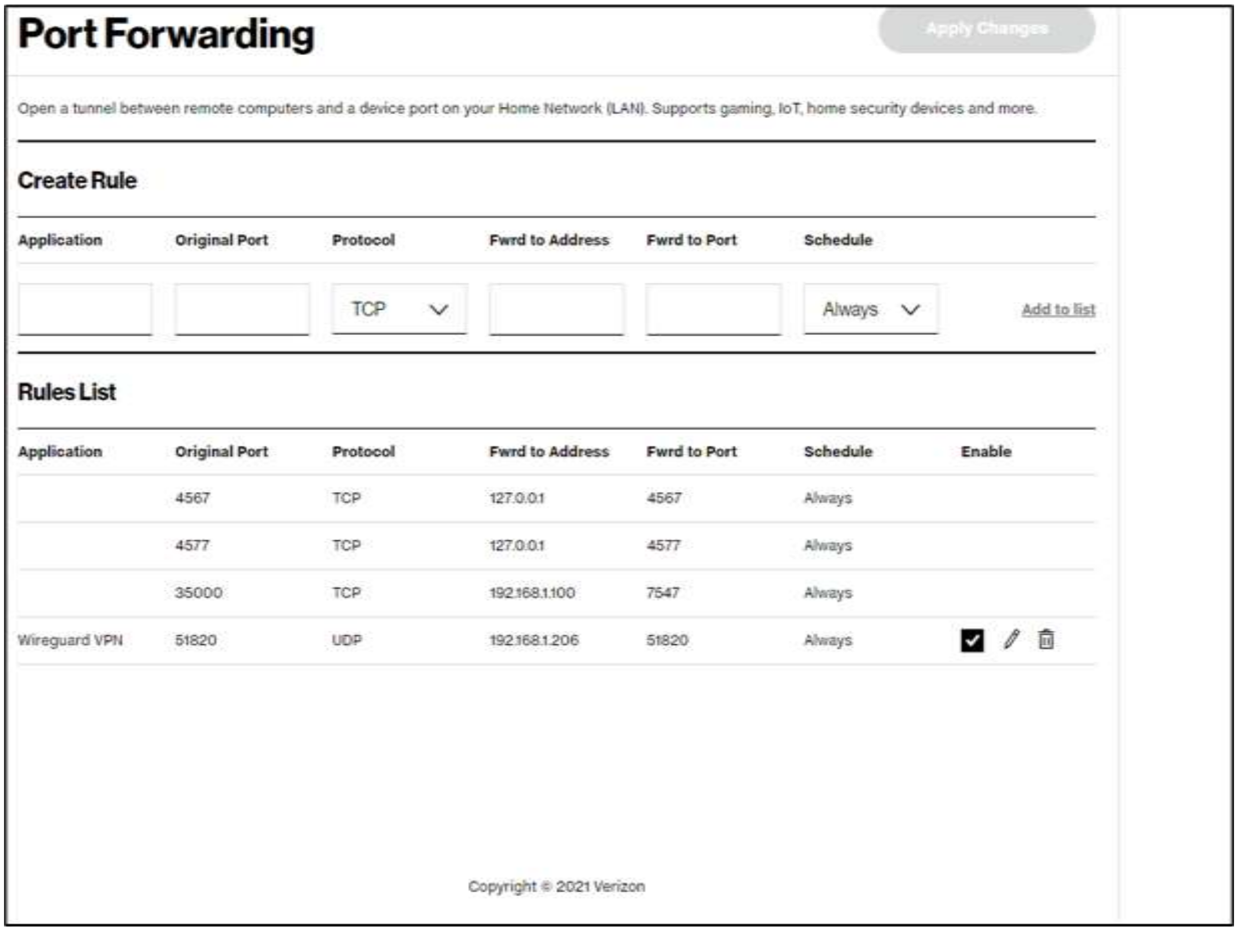
```
root@raspberrypi:/home/pi/configs# wg show
interface: wg0
  public key: [redacted]
  private key: (hidden)
  listening port: 51820

peer: [redacted]
  preshared key: (hidden)
  endpoint: [redacted]
  allowed ips: 10.6.0.2/32
  latest handshake: 5 minutes, 49 seconds ago
  transfer: 13.87 MiB received, 460.64 MiB sent
root@raspberrypi:/home/pi/configs#
```

Screenshot of executing the PiVPN QR code command



```
pi@raspberrypi:~ $ pivpn -qr
::  Client list  ::
1) alex
Please enter the Index/Name of the Client to show:
```

Screenshot of creating a port forwarding rule of protocol UDP using port number 51820



Port Forwarding

Open a tunnel between remote computers and a device port on your Home Network (LAN). Supports gaming, IoT, home security devices and more.

Create Rule

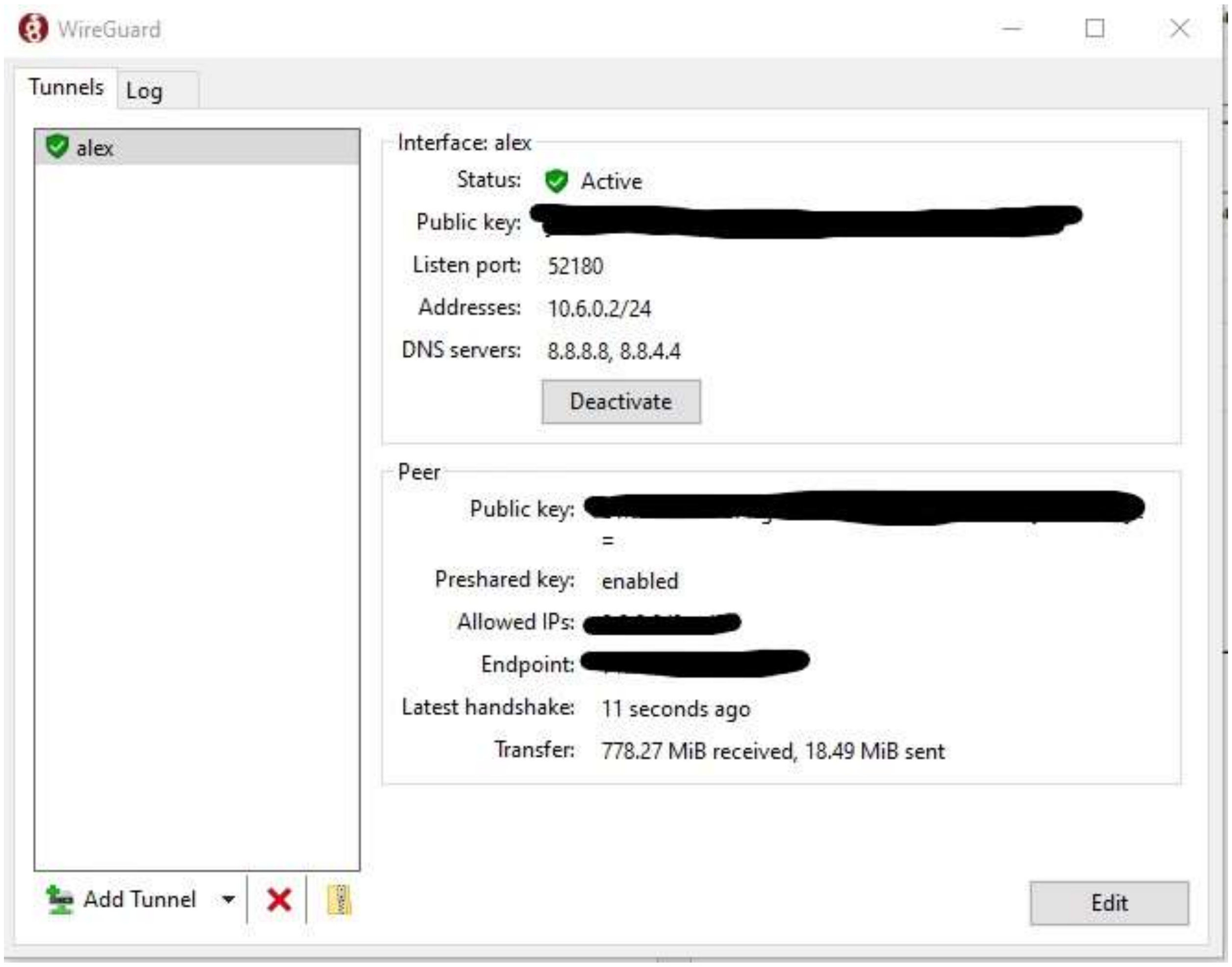
Application	Original Port	Protocol	Fwrd to Address	Fwrd to Port	Schedule	
		TCP			Always	Add to list

Rules List

Application	Original Port	Protocol	Fwrd to Address	Fwrd to Port	Schedule	Enable
	4567	TCP	127.0.0.1	4567	Always	
	4577	TCP	127.0.0.1	4577	Always	
	35000	TCP	192.168.1.100	7547	Always	
Wireguard VPN	51820	UDP	192.168.1.206	51820	Always	✓

Copyright © 2021 Verizon

Screenshot of the WireGuard Software



Interface: alex

Status: Active

Public key: [redacted]

Listen port: 52180

Addresses: 10.6.0.2/24

DNS servers: 8.8.8.8, 8.8.4.4

Peer

Public key: [redacted]
=

Preshared key: enabled

Allowed IPs: [redacted]

Endpoint: [redacted]

Latest handshake: 11 seconds ago

Transfer: 778.27 MiB received, 18.49 MiB sent

Screenshot of WireGuard Mobile App

2:29
Settings
WireGuard
+

alex's home vpn