TALLINN UNIVERSITY OF TECHNOLOGY
School of Information Technologies

Aleksandr Trohhatšov 192931IVSB

A Low-Cost Secure Connection Tunnel Solution for Raspberry Pi Zero Computers

Bachelor's thesis

Supervisor: Aleksei Talisainen
MSc (Lecturer)

Tallinn 2022

TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Aleksandr Trohhatšov 192931IVSB

Odav ja turvaline ühendustunnelilahendus Raspberry Pi Zero arvutitele

Bakalaureusetöö

Juhendaja: Aleksei Talisainen
MSc (Lektor)

Tallinn 2022

Annotatsioon

Odav ja turvaline ühendustunnelilahendus Raspberry Pi Zero arvutitele

Tänu sellele, kuidas tänapäeva maailm on arenenud, ei saa inimene iga üksikut tehnoloogiat kogu aeg endaga kaasas kanda. Seetõttu on ülimalt tähtis, et meil oleks võimalus ühendada iga päev kaasas kantav tehnoloogia süsteemidega, millel pole silmapaistvat kaasaskantavust ja mis tuleb seetõttu jätta staatilisele kohale. Selle lõputöö eesmärk on tuua välja sobiv ja kaubanduslikult saadav lahendus VPN-tunneli jaoks väljastpoolt eemal asuvasse privaatvõrku, samuti toota Ansible baasil hõlpsasti juurutatav installiskript koos teenuse jälgimisega. , alternatiivina laialt saadaolevatele Bashi-põhistele skriptidele. VPN-protokolli jaoks valiti Wireguard selle tõhususe ja selle juurutamiseks kasutatava platvormi jaoks Raspberry Pi Zero 2 W selle juurdepääsetavuse ja taskukohasuse tõttu. Vastav Ansible skript on juurutatud.

Abstract

A Low-Cost Secure Connection Tunnel Solution for Raspberry Pi Zero Computers

With how the modern world has developed, a person cannot always carry every single piece of technology with him. Therefore, it is paramount to have a way to link the technology we carry with us every day with the systems that do not have outstanding portability and thus have to left at a static position. The aim of this thesis is to highlight a suitable and commercially available solution for a VPN tunnel to a remotely situated private network from outside of it, as well as produce an easily deployable install script, complete with monitoring for the service, based on Ansible, as an alternative to the widely available Bash-based scripts. For the VPN protocol, Wireguard has been chosen for its efficiency and for the platform to deploy it on, a Raspberry Pi Zero 2 W has been chosen due to its accessibility and affordability. The appropriate Ansible script has been deployed. The thesis is in English and consists of 33 pages, 9 chapters, 9 figures and 1 table.

List of tables and figures

List of abbreviations and terms

ACL	Access Control List
AES	Advanced Encryption Standard
ARM	Advanced RISC Machines
CA	Certificate Authority
CGNAT	Carrier Grade Network Address Translation
CISC	Complex Instruction Set Computer
CPU	Central Processing Unit
DDNS	Dynamic Domain Name System
GPIO	General Purpose Input/Output.
HDMI	High-Definition Multimedia Interface
HDR	High Dynamic Range
HKDF	HMAC-Based Key Derivation Function
HMAC	Hash-Based Message Authentication Code
HTTPS	Hypertext Transfer Protocol Secure
ISP	Internet Service Provider
MOBIKE	Mobile Internet Key Exchange
NAS	Network-Attached Storage
NAT	Network Address Translation
RISC	Reduced Instruction Set Computer
RSA-4096	Rivest-Shamir-Adleman on 4096 bits
SATA	Serial AT Attachment
SBC	Single-board Computer
SHA-512	Secure Hash Algorithm on 512 bits
SoC	System-on-a-chip
SSH	Secure Shell
SSL	Secure Sockets Layer
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
USB	Universal Serial Bus
VPN	Virtual Private Network
WAN	Wide-Area Network
WSL	Windows Subsystem for Linux

CONTENTS

Contents

1. Introduction

As time keeps moving, we as humans invent increasingly sophisticated solutions for aiding us in our daily lives. As we mostly use these devices at home, they often must stay there, causing inconvenience by not having the necessary tools available wherever the person is. This is where a secure connection like a VPN (Virtual private network) to a private network is vital. It would allow for access to your private network and the services hosted on that network. Due to this, another issue instantly materialises. What if you, as a home user, want that kind of connection but do not desire expensive equipment and yet want total control over the installation, as well as the ability to deploy changes in minutes and be able to quickly respond to a service outage and recover your server if your hardware or software fails. This is what this thesis aims to highlight and examine.

2. What is a Raspberry Pi?

Devices like the Raspberry Pi are an evolution of the popular miniaturisation concept. Created by the Raspberry Pi Foundation, a UK-based charity organisation, in 2012, this is the most popular single-board computer in the industry. What makes single-board computers so incredibly unique is that instead of using multiples of large components that you connect to a motherboard for those components to communicate with each other, everything is already integrated into this board from the start.

Most single-board computers use ARM architecture [1]. It is an architecture based on the RISC instruction set, which is most used in mobile devices like smartphones, as opposed to the more mature standard of CISC. This is a crucial fact because RISC was an instruction set that has been built from its inception to maximise efficiency and to minimise heat and power consumption. This allows for less stringent power delivery requirements, as well as the ability to integrate more of the components into a special System-on-a-chip (SoC), which also adds a performance benefit, due to lowering access latency between system components. The Raspberry Pi takes advantage of this concept to its fullest.

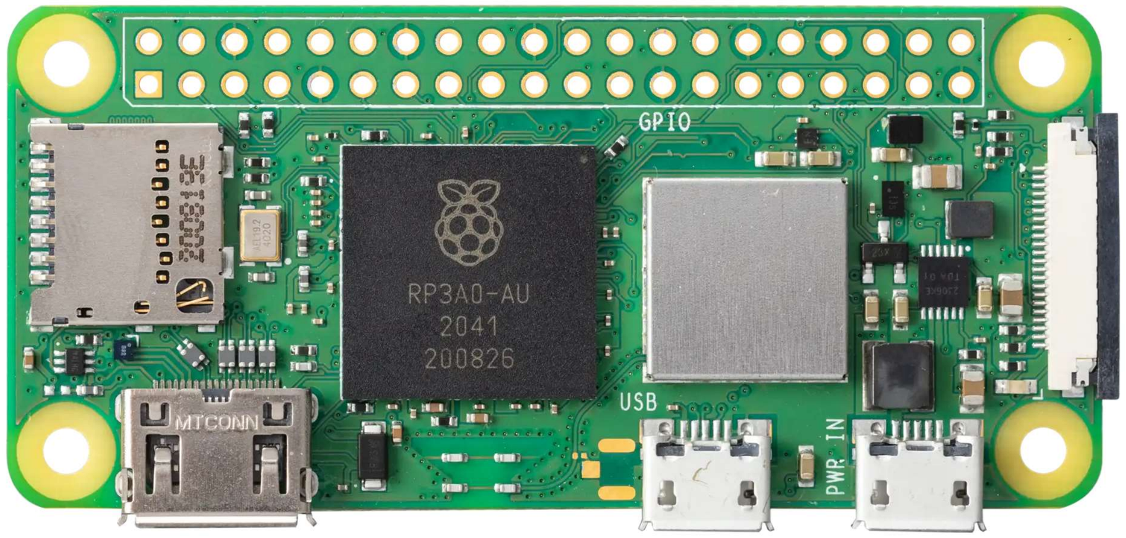


Figure 1. Raspberry Pi Zero 2 W

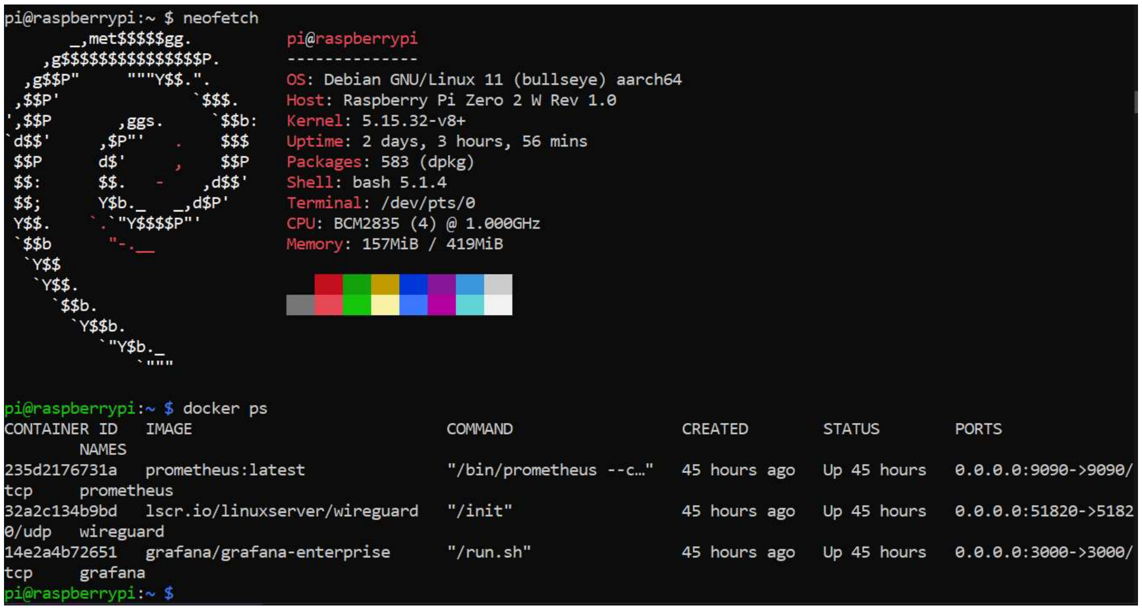
The Raspberry Pi Zero 2 W uses a custom Broadcom SoC, using 4 Cortex-A53 low-power ARM cores, with both the CPU and 512 MB of RAM integrated into one chip [2]. The board also contains 2 USB Micro-B ports and 1 micro-HDMI port on the side for connectivity. For storage, the board uses a Micro-SD storage card, which is used to store all the files we will require for the project itself. Power delivery is handled through the Micro-USB ports.

That is all the main aspects of the device that every buyer will receive. The following two aspects will depend on what model you have chosen, namely connectivity and the presence of GPIO pins on the device. The USB ports on the device in all cases support adapting them to an Ethernet connection to gain Internet connectivity, as per the USB specifications, although it is advantageous to pay about 20% more in most cases for a version of the board that contains a 2.4GHz 802.11 b/g/n Wi-Fi module. This will prove to be fruitful to some users because it will allow you to run your Raspberry-Pi as a ‘headless’ device. It should however be kept in mind that due to the Wi-Fi antennas being confined to the 2.4GHz band, internet speeds that can be achieved are limited to around 25 Mbps, which may become a system bottleneck if the user requires high bandwidth.

A headless device [3] or service is a term that refers to a server or application that is designed from the ground up to run without the use of a graphical user interface of any kind. This is advantageous in 2 ways. Firstly, it eliminates the need for the connection of additional adapters for a display output and other peripherals. Secondly, it allows the user to install a terminal-only version of the operating system we are going to be using for the project, Raspberry Pi OS, formerly known as Raspbian, which, as it is the platform that will be utilised during this project, will be discussed further later in the thesis. The second optional addition is the previously mentioned GPIO pins. These are pins that can, through appropriate software development, be used to connect any peripheral a given user may desire, ranging from connecting old gaming console controllers to adapting the pins to a SATA slot for adding external storage to Raspberry Pi, which can for instance be used as a platform for creating a basic instance of a NAS(Network-attached storage) device.

Raspberry Pi OS [4] is a Linux distribution based on Debian Linux, which runs on both 32-bit and 64-bit devices based on the ARM architecture. For this project, the 64-bit variant has a multitude of benefits despite the slightly higher resource requirements the choice entails due to many closed-source apps only being available for arm64, and open-source apps not being fully optimised to run on 32-bit armhf. Beyond that, the arm64 instruction set has some inherent performance gains, which are already apparent in benchmarks against amrhf but are expected to be reflected in real-world application performance in the future, although for our purposes the 32-bit variant, which admittedly is far more mature, would be suitable as well.

NB! At the time of authoring this paper, Docker for 32-bit ARM architectures, for example armhf, may experience GPG key verification errors for mirrors in newer Debian-based distros due to libseccomp2 library issues , which are currently still being fixed, thus it is highly recommended to avoid using the 32-bit version of Raspberry Pi OS.



```
pi@raspberrypi:~ $ neofetch
       _,met$$$$$gg.          pi@raspberrypi
    ,g$$$$$$$$$$$$$$$P.       --------------
  ,g$$P"     """Y$$.".        OS: Debian GNU/Linux 11 (bullseye) aarch64
 ,$$P'              `$$$.     Host: Raspberry Pi Zero 2 W Rev 1.0
',$$P       ,ggs.     `$$b:   Kernel: 5.15.32-v8+
`d$$'     ,$P"'   .    $$$    Uptime: 2 days, 3 hours, 56 mins
 $$P      d$'     ,    $$P    Packages: 583 (dpkg)
 $$:      $$.   -    ,d$$'    Shell: bash 5.1.4
 $$;      Y$b._   _,d$P'      Terminal: /dev/pts/0
 Y$$.    `.`"Y$$$$P"'         CPU: BCM2835 (4) @ 1.000GHz
 `$$b      "-.__              Memory: 157MiB / 419MiB
  `Y$$
   `Y$$.
     `$$b.
       `Y$$b.
          `"Y$b._
              `""""

pi@raspberrypi:~ $ docker ps
CONTAINER ID   IMAGE                            COMMAND                  CREATED        STATUS        PORTS
        NAMES
235d2176731a   prometheus:latest                "/bin/prometheus --c…"   45 hours ago   Up 45 hours   0.0.0.0:9090->9090/
tcp     prometheus
32a2c134b9bd   lscr.io/linuxserver/wireguard    "/init"                  45 hours ago   Up 45 hours   0.0.0.0:51820->5182
0/udp   wireguard
14e2a4b72651   grafana/grafana-enterprise       "/run.sh"                45 hours ago   Up 45 hours   0.0.0.0:3000->3000/
tcp     grafana
pi@raspberrypi:~ $
```

Figure 2. Raspberry Pi OS

3. Ansible and the infrastructure

Ansible is a management tool that uses special scripts called playbooks as a basis for automating the setup and deployment of key services [5]. Ansible playbooks are effectively frameworks, or pre-written code that developers can utilise on the fly or as a starting point. Ansible playbooks are used to automate IT infrastructure (such as operating systems or container platforms), networking, security systems, and developer personas. Ansible connects to your nodes and transmits lightweight programs known as "Ansible modules" to them. These programs are intended to serve as resource models for the desired state of the system. Ansible then runs these modules (by default over SSH) and removes them as soon as they have been executed. A significant benefit of using this toolkit that must be mentioned is that it runs in agentless mode, meaning it does not require any additional software to be deployed on the hosts Ansible is serving, which is a crucial benefit of using Ansible for deployment automation. The structure of how an Ansible playbook is executed is quite simple. When executing the playbook, the master host then attempts to establish an SSH connection to the hosts and then executes the requested commands, listed in the host file the user specifies.

4. VPN Protocol Selection

A virtual private network (VPN) is a secure link between a client and a network. All commonly used operating systems can be used as a VPN client.

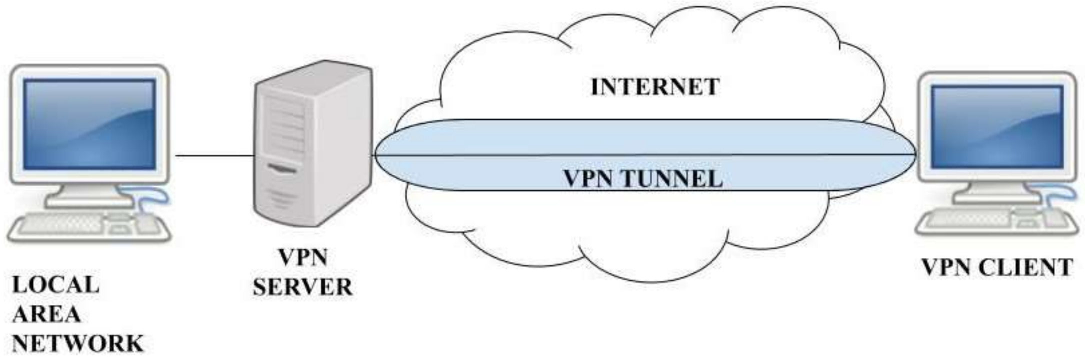


Figure 3. Basic visualisation of how a VPN works[6]

There are multiple VPN protocols in consideration, namely of OpenVPN, IPSec IKEv2 and Wireguard. Fundamentally each of the protocols have specific traits that warrant its usage over other options, depending on the scenario. Based on this fact, the individual traits of the previously mentioned VPN protocols will be discussed and the best one among them will be chosen for usage during the experiment.

4.1. OpenVPN

OpenVPN is free and open source [7], and it makes use of the OpenSSL library as well as additional security features. OpenSSL is a cryptography and SSL/TLS toolkit, which is required for secure end-to-end communications. All the essential encryption and authentication components are provided by this toolkit. As a result, OpenVPN may make use of the OpenSSL library's multiple ciphers. Out of those ciphers, the Blowfish and AES ciphers are preferred by most VPN services, Blowfish being the default option for OpenVPN, for encrypting data using data channel encryption and the connection via control channel encryption. Most of the time, OpenVPN uses the greatest level of encryption possible, meaning a cipher that uses 256-bit encryption, an RSA-4096 handshake, and SHA-512 hash authentication would be utilised for this purpose. It may include HMAC authentication and Perfect Forward Secrecy on occasion. To establish a connection, the protocol requires the authentication of previously generated certificates of both the server and client, each signed by a master CA (certificate authority). After authentication, the devices exchange traffic via previously generated keys. OpenVPN also takes advantage of hardware acceleration to boost performance and additionally can run on a single TCP/UDP port 443. OpenVPN unfortunately shares a port with HTTPS, which is the building block of secure web traffic. This fact severely complicates securing the port against OpenVPN connections. The UDP port for the service can be set to run on another port number if needed.

4.2. IPSec IKEv2

IKEv2 is a tunnelling protocol [8] , which is used to securely exchange keys. It requires another protocol for encryption and asymmetric authentication to be utilised as a VPN protocol. Therefore, it is used in conjunction with IPSec, which provides a secure connection, supports a variety of encryption schemes, and supports up to 256-bit encryption. The tunnelling protocol was originally developed as result of a collaboration between Microsoft and Cisco to be used as a “request-and-response encryption protocol” and released officially in 2005. IKEv2 uses the Diffie-Hellman key exchange to initiate and complete a secure handshake and permits using Perfect Forward Secrecy to safeguard its data due to IKEv2 characteristics as a key exchange protocol. A mobility and multihoming protocol is used to facilitate conventional mobility (MOBIKE). As a result, users can switch networks while still maintaining a secure connection, which could be used to provide connection redundancy with almost no connection loss due to the VPN link becoming unreachable. IKEv2 operates on the UDP 500 port.

4.3. Wireguard

The final VPN protocol which will be compared is Wireguard [9]. It is the newest standard of the three which are being discussed, officially publicly available as of September 2019. For encryption algorithms and ciphers, ChaCha20 and Poly1305 represent 2 frequently mentioned options for use inside the Wireguard protocol. Encryption is done with ChaCha20, while authentication of the connection is done with Poly1305.Wireguard also has the ability to use other cryptography techniques such as the Noise protocol framework, Curve25519, BLAKE2, SipHash24, and HKDF. The concept of Cryptokey Routing, which works by associating public keys with a list of tunnel IP addresses that are allowed inside the tunnel, is the basis of Wireguard. Each network interface has its own private key as well as a list of peers, with each peer having a unique public key._When sending packets, the list of allowed IPs simulates a routing table, and when receiving packets, it acts as an ACL (Access control list). Encapsulated traffic travels through the virtual tunnel through UDP after a handshake and key exchange between the peer and server has been established. The port number for Wireguard is configured by the user, usually being port numbers such as 51900 or 51820.

4.4. Choosing the appropriate VPN

Due to all of the previously mentioned protocols having their own applications, it is necessary to examine the advantages of each in relation to the objective of this thesis, that being a user-friendly solution to remotely access a user's home network over the open internet, with a reliable connection, which can be deployed on Raspberry Pi Zero 2 W and handle demanding workloads, and which can be accessed from any device.

Beginning with OpenVPN, the advantages of the protocol are quite clear. Due to its modularity, configuration of features such as password authentication, load-balancing or direct access to a Samba share can be configured using community resources, like those on the official OpenVPN website [10]. That extra configuration however is quite complex, which clashes with the goal of this thesis in creating an accessible solution. Moreover, the protocol, has causes about 5-10% bandwidth reduction when used due to every packet being encrypted and decrypted while being transported through the tunnel.

IKEv2 is also a formidable protocol with excellent compatibility with all existing operating systems, especially mobile devices. It delivers robust security for data-sensitive applications, while offering an excellent automatic reconnection feature, which quickly establishes a new connection to the VPN server if the link were to be dropped. Instant network switching is also an excellent point when considering IKEv2 as an option for a protocol. Unfortunately, some drawbacks concerning IPsec IKEv2 hinder its applicability. The first would be that the protocol is closed source. This actively hinders its adaptability and security, as a hidden codebase can be considered far more vulnerable to exploits than an open-source solution. Additionally, it is vital to mention that the port which IKEv2 uses, UDP 500, is frequently blocked on many networks, rendering the protocol completely useless, which is a factor that cannot always be avoided when connecting to public networks.

Wireguard is the final candidate for the protocol this project will use, and which integrates a crucial feature for mobile devices called roaming. Due to how the client configuration contains only an initial endpoint for the server, if the endpoint is changed, the client discovers the new endpoint of the tunnel without much delay in the connection. Clients do not have fixed endpoints for Wireguard, pinpointing their location to the server by sending authenticated data to it. This fact allows for uninterrupted usage of the tunnel while on cellular internet, which frequently changes the IP address of the devices using it. The first message sent by the client is authenticated to avoid the possibility for unauthorized access. Wireguard also has a dynamic port number, which can be adapted to whatever network the user installs their server node on. Finally, Wireguard is the most efficient protocol out of the 3 being compared, idle CPU load of which can be considered negligible. The only significant downside, which should be mentioned, is that the protocol does support cryptographic agility [11], which means that replacing cryptographic algorithms related to the protocol is not possible, as it only supports a small number of established primitives.

Based on this comparison, Wireguard is the best candidate for the application discussed in this thesis, due to the simplicity of configuration, the low overhead and ability to dynamically adjust the endpoints to maintain connection with roaming.

5. Building the script

5.1. Required hardware

Now that the theory of this work has been successfully covered and the type of VPN has been chosen, the breakdown of the practical portion of this thesis can begin. The setup will begin with the hardware components as well as the networking capabilities involved, which admittedly are not extensive. A person wishing to deploy this solution will need 3 pieces of hardware: a Raspberry Pi Zero 2 W, a microSD card and a computer which can be accessed to both flash the memory card with our operating system, Raspberry Pi OS, to use a secure connection like SSH and in order to deploy the ansible script which will be examined thoroughly later during this analysis.

5.2. Required networking capabilities

For the networking components, it is required to know which technology does your internet service provider uses for distributing IP addresses to their clients. If ISP is using CGNAT, then the ability to use a proposed solution depends on whether the provider network has disabled all their interface ports or not and on whether the ISP has adopted IPv6 into their topology as well [12]. CGNAT is a layer between the user and the global Internet that assigns the same public IPv4 address to several private connections at once, routing each point (user) through distinct ports. The difference between CGNAT and regular NAT networks is that instead of all private networks being dedicated their own WAN IP address, CGNAT aggregates multiple host private gateways to a single external gateway. This solution was originally meant to conserve the dwindling supply of public IPv4 addresses, which has since been depleted, until a suitable solution, namely IPv6, could be used to replace it. Since then, it has enabled internet service providers the ability to distribute one IP address to many of their clients, maximising the amount of potential total clients. If a client requires a dedicated IP, whether an ISP would be so inclined to provide the service varies dramatically. If the client has a dedicated IP for his private network by default or have acquired such an address from your provider, then DDNS and, subsequently, Wireguard, should be an attainable solution.

Operating system image can be accessed by visiting the official Raspberry Pi Foundation website and acquiring the Raspberry Pi OS Imager [13] for the selected platform.

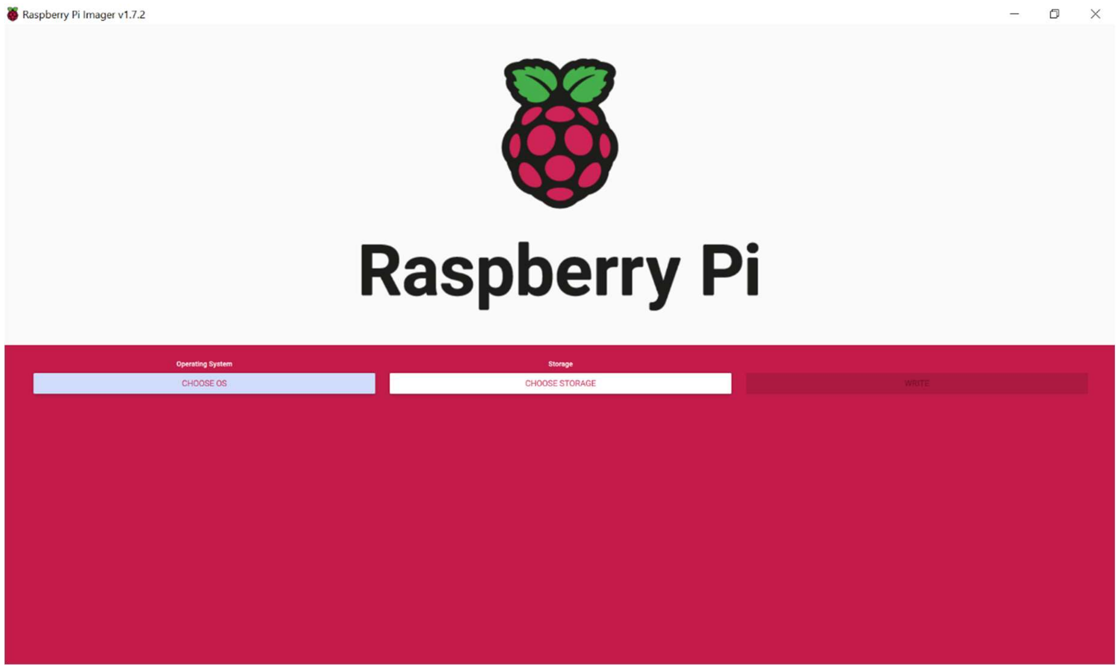


Figure 4. Raspberry Pi Imager Menu

After downloading and running the program the user will be greeted with a simple selection screen, where the first setting screen named ‘Operating System’ enables viewing and selecting a suitable OS image. Since the 64-bit version will be used for this thesis, that option can be selected via ‘Raspberry Pi OS – Other’. Here the user is presented with a choice of whether they want a desktop environment or not. Considering that the VPN node will be operated via a remote shell, as a headless node, and to save on system resources, the version without a desktop environment, the Lite Version, will be used. After selecting the image version, a cog icon appears in the imager, where the user can configure the device hostname, Wi-Fi connection, region settings and Linux user credentials. After suitable configuration, it is only required to select the appropriate volume to flash the operating system and initiate writing the data. After a short amount of time, the SD card is ready for deployment. Inserting the memory card and plugging in a USB power adapter should successfully boot the single-board computer. When the device is powered on, it should have a small yellow LED rapidly flashing whenever the SBC is trying to execute and action. If the boot of the machine was a success, then the user should notice the board status LED having stopped its blinking process. The next step is to cover the infrastructure that is going to be deployed.

The following diagram is presented as the structure of the infrastructure, which is being covered in this thesis.

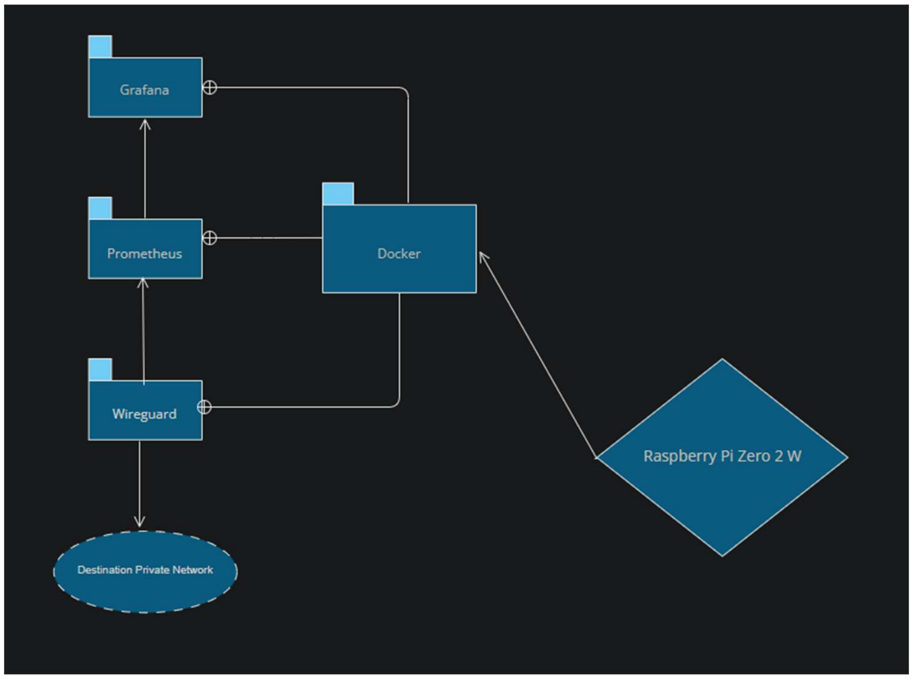


Figure 5. Infrastructure model

This mock-up depicts how the entire infrastructure would function. Docker will be the center of the application suite, hosting Wireguard, Grafana and Prometheus, using containers from both community-proven and official Docker images. Prometheus Node Exporter will be used to forward required diagnostic data to the Grafana container which would have to be setup manually. The installation script is separated into 4 parts: Wireguard installation and configuration, Prometheus installation, Grafana installation and Node Exporter installation.

Wireguard installation includes installing Wireguard itself, the configuration files that determine how the tunnel would operate and setting up the Docker container. For container installation, a community Wireguard image by LinuxServer [14] will be used as the base. The configurations feature a private network, to which, after a successful handshake, a client can connect to using a client app. The following configuration files show how the Wireguard VPN is configured [14].

Server Interface Configuration:

```
[Interface]
Address = xxx.xxx.xxx.1/24
ListenPort = xxx
PrivateKey = --redacted--

PostUp   = iptables -A FORWARD -i %i -j ACCEPT; iptables -A FORWARD
-o %i -j ACCEPT; iptables -t nat -A POSTROUTING -o wlan0 -j
MASQUERADE
PostDown = iptables -D FORWARD -i %i -j ACCEPT; iptables -D FORWARD
-o %i -j ACCEPT; iptables -t nat -D POSTROUTING -o wlan0 -j
MASQUERADE
[Peer]
## Client
AllowedIPs = xxx.xxx.xxx.2/32
PublicKey = --redacted--
```

Client Interface Configuration:

```
[Interface]
## Client
Address = xxx.xxx.xxx.3/24
PrivateKey = --redacted--
[Peer]
PublicKey = --redacted--
AllowedIPs = xxx.xxx.xxx.1/32
Endpoint = --redacted--:xxx
```

As it is depicted in the configuration, the bridge is created from a private network, for example 10.155.2.1, to an endpoint accessible outside of this network. This endpoint can be both an IP address, like a VPS, or in the case of this thesis, a DDNS domain. DDNS can be acquired via both paid and free sources. DuckDNS can be recommended as a free service, but if the user is in search of his own domain, then purchasing a budget domain from Namecheap and using their DDNS service is recommended. The previously mentioned Asus DDNS service [15] is also an option if the user owns the supported hardware to support it, namely an Asus wireless access point. Due to the existence of both options, it is then paramount to examine both as well.

For the purposes of this thesis, the Namecheap solution will be utilised, due to having less services involved with the VPN solution. If a user owns an ASUS wireless network router, the manufacturer also offers such a service, which can be activated from the ASUS Router administration panel. The interface in the PostUp and PostDown configuration depends on the interface on the host device used to access the internet. In this case, the wlan0 interface oversees routing all IP addresses (hence the 0.0.0.0/0 marked in the allowed IP address section) and should handle all the traffic passing through your server. By changing this value to any subnet, corresponding to your desired network topology, the user can restrict a peer's access to networks through the VPN, for example by locking traffic to only move within the private network.

5.2. Prometheus

Prometheus [16] is a free and open-source program for gathering and processing metrics. It comprises of a timeseries database and a query language for retrieving and processing the metrics it stores. Metric exposure is handled by separate services, which the Prometheus server can access. It comes with an extremely basic web interface out of the box. Third-party technologies like Grafana can be utilized to create an useful dashboard system. The default configuration [17] keeps track of the prometheus process, but not much else. You can use prometheus-node-exporter to perform system monitoring by pulling metrics from the local system. The prometheus-node-exporter service can be started and enabled, which will launch it by default on port 9100 of the host server. Once the service is up and operating, prometheus must be configured to scrape the exporter service on a regular basis in order to collect data.

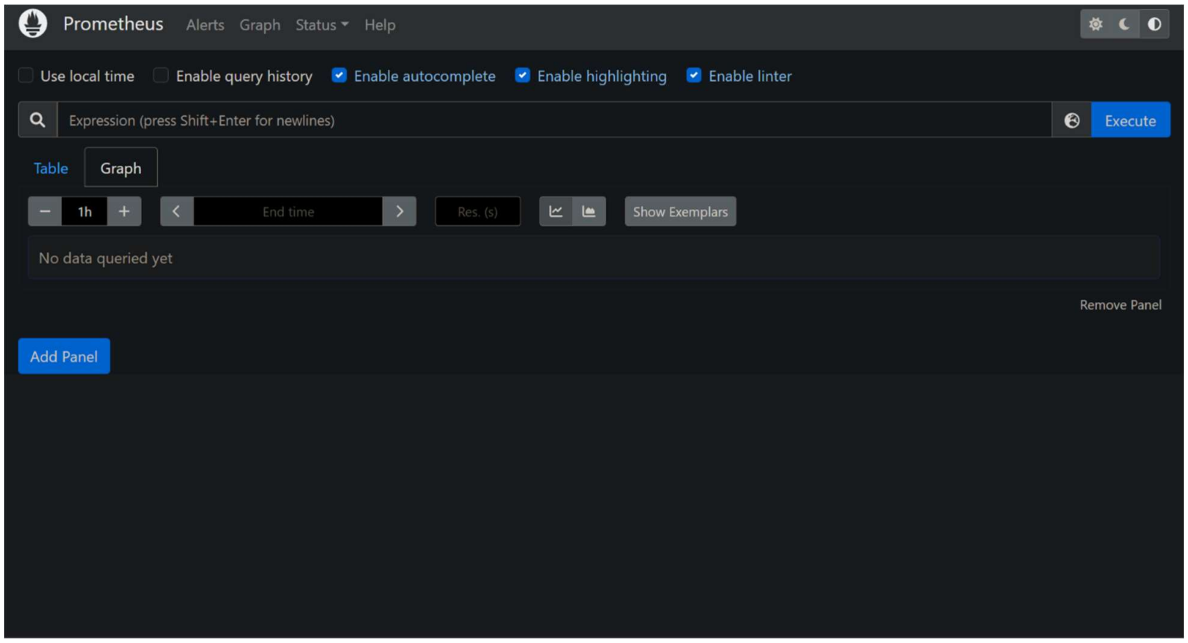


Figure 6. Prometheus GUI

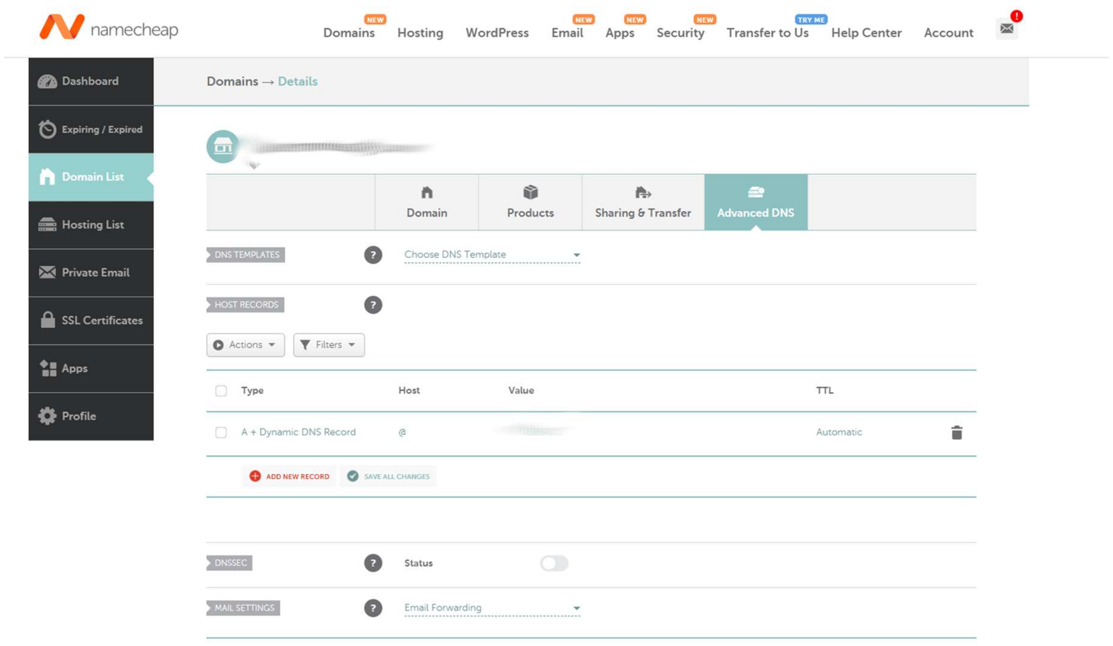
5.3. Grafana

Grafana is a web-based analytics and interactive visualization program that is open-source. It enables the user to consume data from a wide range of sources, query it, and visualize it in beautiful, customisable charts for quick study. It is also possible to configure alerts so that the user can be notified of unusual activity in a significantly quicker manner than the user would otherwise. Grafana, in simple terms, allows the user to visualize the mountains of performance metric data generated from user's apps. This will enable the user to draw educated conclusions and make suitable changes in order to maintain the health of your application stack. The data gets extracted from exporters like Prometheus and is displayed using panels on a dashboard.

Grafana configuration, in the case of the official Docker container, is done via environment variables when deploying the container itself. If installing on the actual hardware, outside of a container, then the configuration is controlled via a default.ini or grafana.ini configuration file, depending on the operating system used [18].

5.4. DDNS Updater

Since this thesis covers connecting remotely to your own home network from outside of such network, it would be logical to assume that the device you are hosting the VPN through is also located in your home and/or private network. Using a VPN client, the user gains access to that network by pointing the connection to the external IP address of your network, which, using port forwarding, transmits the VPN server port over the Internet. However, this regular solution has some shortcomings, namely the fact that your external IP can be configured to frequently change, in which case your Wireguard instance will fail to complete the required handshake for the VPN connection to be established. To avoid encountering such an issue, DDNS must be used, in order to bind an A record to your external network IP address. To achieve successful DDNS routing, a provider for the service must be chosen. For the purposes of this thesis a solution using Namecheap as the DDNS provider will be used. A solution involving editing the configuration of an Asus-branded gateway router of the private network will be given a demonstration as well. To send DNS Updates, in the case of Namecheap, a Python program [19] will be used which creates an API call request to Namecheap, using variables such as the domain, IP address to which needs to be matched and the DDNS password, which was previously generated and received when creating the appropriate A Record in the ‘Advanced DNS’ tab of Namecheap’s client domain management. If all the data matches, then Namecheap can point the listed domain to the requested IP address, which allows for more intuitive establishing of a VPN connection.



DNS TEMPLATES: Choose DNS Template

HOST RECORDS

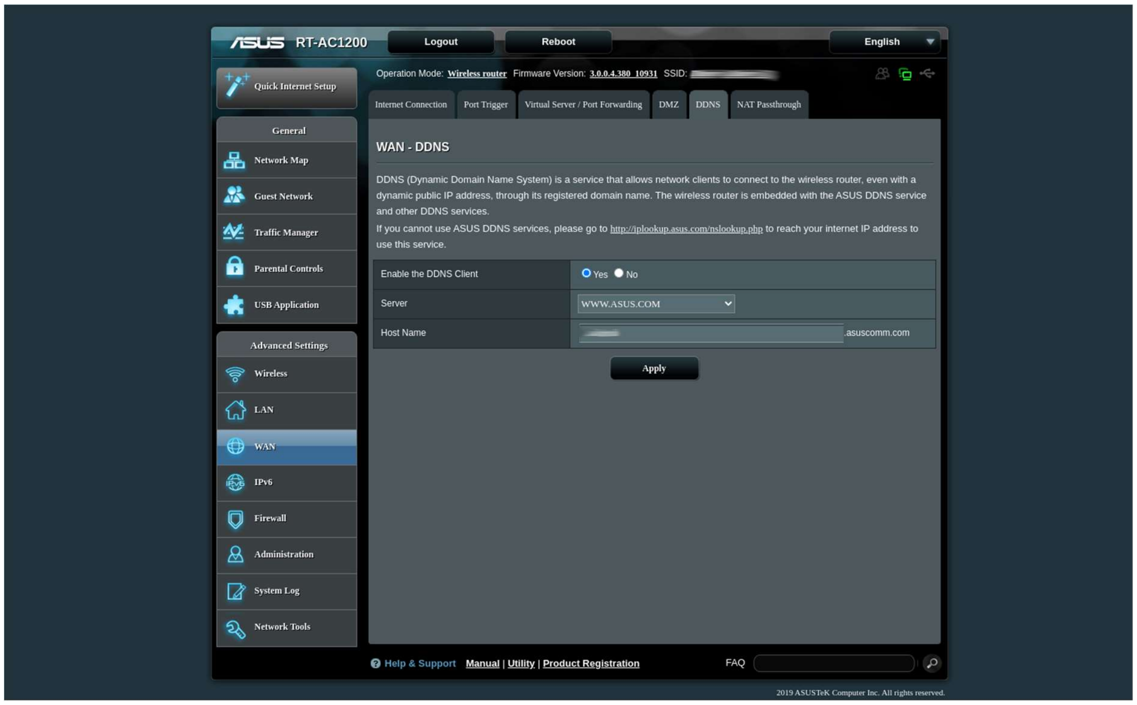
Type	Host	Value	TTL
A + Dynamic DNS Record	@		Automatic

DNSSEC: Status

MAIL SETTINGS: Email Forwarding

Figure 7. Namecheap Advanced DNS Configuration

In the case of the Asus router configuration, it becomes mildly easier. By accessing the router control panel, which is by default 192.168.1.1, the person is greeted with an administration sign-in and subsequently the admin panel. From this screen navigating to ‘WAN’ and ‘DDNS’ the user are presented with a menu to select your DDNS provider and mark a URL via which the router will be accessible outside the private network. After applying the settings, the URL should now be paired to the WAN IP of your router.



Operation Mode: Wireless router Firmware Version: 3.0.0.4.380_10931 SSID: [redacted]

WAN - DDNS

DDNS (Dynamic Domain Name System) is a service that allows network clients to connect to the wireless router, even with a dynamic public IP address, through its registered domain name. The wireless router is embedded with the ASUS DDNS service and other DDNS services.

If you cannot use ASUS DDNS services, please go to http://iplookup.asus.com/nslookup.php to reach your internet IP address to use this service.

Enable the DDNS Client	[✓ Yes] No
Server	WWW.ASUS.COM
Host Name	[redacted].asuscomm.com

Apply

Figure 8. DDNS Configuration menu in Asus Router Administration panel

5.5. Security level of infrastructure

Before the infrastructure can be installed, an analysis from a security perspective must be performed to determine whether the infrastructure meets all of the necessary requirements for data security. According to FIPS 199 [20], security can be divided into 3 main pillars: availability, integrity and confidentiality. Availability stipulates that a service or source of information must be available in a timely and dependable manner. Integrity points to protection against information forgery and the destruction of information by using methods to prove data authenticity. The final main security objective is confidentiality, which concerns protecting and maintaining restricted access to sensitive information. If any of these 3 security categories are compromised, an adverse effect of varied importance would be seen. These security objective breach categories can be classified as either of low, moderate or high severity. In the case of Wireguard, all of the 3 categories could have a moderate or high adverse effect on the service. The first security to be examined to determine the severity of adverse effects is confidentiality. If a third party gains access to sensitive information like the private key of the server node used in Wireguard, then the third party would have access to all traffic being transported through the secure tunnel, which in most use cases for this device, would mean the entire home private network would be compromised, but only if safeguards have not been put in place to protect individual nodes on the private network. Due to the server private key being held in a Docker container, which has SSH connections to it completely disabled and which can only be accessed through a secure connection to the VPN node itself. Taking this assessment into account, loss of confidentiality could prove catastrophic to the security of the service, although secured sufficiently. Integrity of the connection is ensured via handshakes performed between the VPN host and client by exchanging public keys, as well as the encryption algorithms used in forming the tunnel. Failure in key exchange yields a failed connection. Integrity can only be feasibly compromised if a third party gains access to the necessary cryptographic keys. Finally, availability is the least catastrophic point of failure. If availability is compromised, the service will not be accessible to the client, which fortunately does not result in data loss due to the connection not being formed in the first place. A possible vector of attack could be the login credentials used for a dynamic DNS service, which the user could have set up. In conclusion, the most severe data breach for this infrastructure from a cyber security point of view would be neglectful storage of associated server and client private keys, which would allow anyone full access to a given private network. Security vectors for prometheus and Grafana are negligible as the services only supply metrics to graphical interface. Although this risk needs to be considered, due to the keys being generated inside a Docker Container, it is unlikely that such a breach would occur if other security practices were followed.

6.Installation

6.1. Script Deployment

Running the setup script requires to clone the repository that has been created by the author for this specific purpose (namely https://github.com/AAdmiral56/wireguard-docker-pi). The user also is required to install Ansible on their computer, using which they have elected to run the installation script. The installation itself is easiest on a Linux-based device, as most package managers, like apt or rpm or others, have Ansible as an easily installable package without any tweaks necessary. In addition to that, both MacOS and Linux distributions support installing Ansible through the Python-based pip package manager. As Ansible requires a Unix-based system to run, installation to Windows involves the usage of the

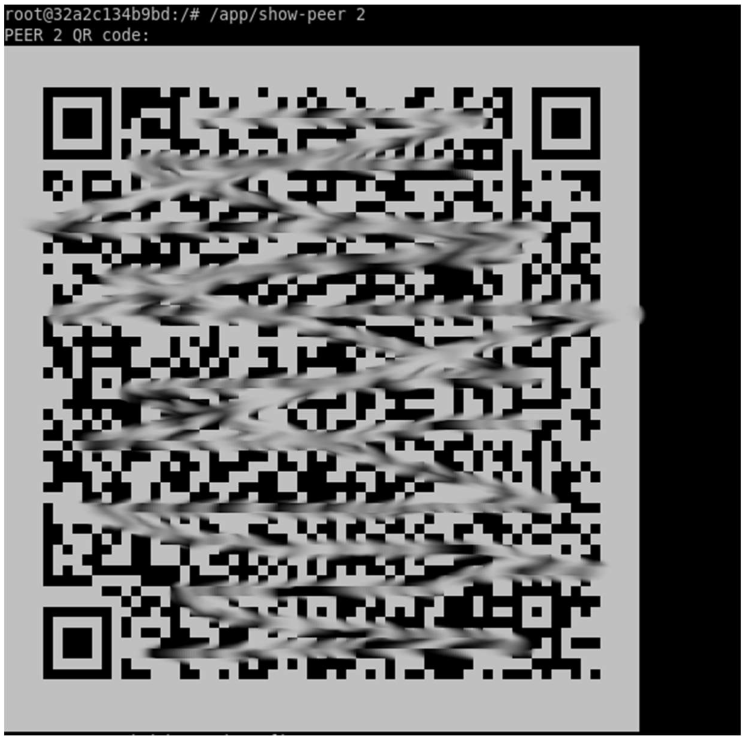
WSL(Windows Subsystem For Linux) [21], the installation of which requires enabling a specific feature in a menu named ‘Turn Windows Features on and off.’ After installing and rebooting, the user is required to install a Linux container using an image from the Microsoft Store, after which the installation is the exact same as it were to be on a Linux-based system.

Before executing the script, variables for /group_vars/all.yaml must be edited with the values that the user desires and the IP Address of the locally placed Raspberry Pi has to match the one allocated on the users’ network. After this step, the playbook can be executed. After 20 minutes or so, the infrastructure should be running on your Raspberry Pi Zero 2 W.

6.2. Mobile Installation

To connect your devices, the user would need special client configuration. For instance, for Android devices, it is paramount that the Wireguard app be downloaded to the phone using either the Google Play Store or the open-source F-Droid App Store. It is important to mention that the second method requires enabling installation of apps through third-party sources, or in other words from sources excluding the Google Play Store, which opens a gateway for malicious applications to a user’s device if not managed properly.

After installation, the user is required to open the application, click the add button and activate the camera for QR Code-based login. To acquire the appropriate configuration code, one must either enter the command ‘qrencode ansiutf8 << app/show-peer/ <number of peer>’ from inside of the Docker Container CLI after entering ‘docker exec –it wireguard bash’ or passthrough the previously listed command by appending it to the docker exec line mentioned here. By acquiring the QR code and scanning it using the built-in scanner, a handshake between the remote server and the mobile device is established. If the previously mentioned process was successful, then the mobile device should have access to the previously defined parts of the network, which in this case would be the entire Internet including the target private network.



```
root@32a2c134b9bd:/# /app/show-peer 2
PEER 2 QR code:
```

Figure 9. QR Code Generation

6.3. Desktop Installation

Compared to the mobile clients, which have a quite intuitive installation procedure, the desktop version is not as seamless. The potential peer must start by installing the desktop version of Wireguard from the project website wireguard.com/install or, in the case of most Unix-based operating systems, installing through the appropriate package manager.

In the case of Windows, once installed, Wireguard opens a graphical menu where one might import their peer configuration to the device. Since this project uses a terminal-based version of Raspberry Pi OS, downloading the configuration from the server itself may not be the most elegant solution. The solution that can be recommended however is using SSH from the device you want to pair to the VPN server to output the contents of the peer configuration file into the console window, after which the user can copy it over to an empty tunnel, which you can create by pressing Ctrl + N on your keyboard while Wireguard is the app in focus. After installing the appropriate config, the tunnel can be established by clicking the ‘Activate’ button that has now appeared in the program. After a couple of seconds, provided the configuration was copied correctly, your desktop device should now be using the VPN tunnel.

When installing Wireguard as a client on a Unix-based system, the configuration resembles what was done during the initial setup of the VPN server. Once the package has been successfully installed, the user must navigate to the standard Wireguard directory /etc/wireguard. There the peer configuration file contents should be inserted as a Wireguard interface, for example wg0.conf. Once this step has been completed, the user must use 'wg-quick up *interface name*' to activate the configuration, which should complete the required handshake and grant the device access to your desired private network

7. Testing

Once the connection has been established, the user can begin to use their device. Testing the impact of the VPN connection on latency of the connection should be an important starting point, as sudden jumps in network access latency can cause services like SSH to become almost unusable due to the very slow response time. Testing for this thesis was conducted via a 100 time ping request to IP address 1.1.1.1. Pinging from Elisa LTE to this IP address yielded a 46.714 ms average latency, with a maximum latency of 96.228 ms. Now with the Wireguard VPN connection being active, again using mobile internet and running the test again, it yields a 58.807 ms average latency with a peak of 94.217. This result proves that, due to the extra processing power and time required to forward and receive data through the VPN tunnel, latency suffers about a 25.89% increase. What is however fascinating is that the same cannot be said about the maximum latency, as it remains the same in both runs, attributing the jumps in latency to other factors in the network. Answering the question of resource usage, watching a 1080p video on YouTube, while also running htop resource monitoring on the Pi in an open terminal window, yielded spikes on individual cores to around 20% usage, up from the usual 3% idle CPU usage. Stressing the VPN node via watching a 4K HDR(High Dynamic Range) YouTube video yields constant processor usage of about 15% on all cores with spikes up to 60% usage when the video was buffering, and thus sending large chunks of data through the tunnel. Using a remote desktop service called Reemo[22], 100% usage on one of the CPU Cores was achieved for a moment when initially launching the remote session, while basic usage while navigating through the UI produced a processor load of around 15%. Considering streaming video is technically about as much as a regular person would do on their mobile device, this could be considered a success. Memory usage after installing all 3 containers is approximately 150 MB out of 419 MB available to the system when using the Lite image of Raspberry Pi OS. This means that the system has sufficient extra memory capacity for deploying other minor services if required.

Workload	Resource Usage
1080p video streaming on YouTube	3% idle usage, 20% when buffering
4K HDR video streaming on YouTube	15% idle usage, 60% when buffering
Remote Desktop using Reemo.io	15% normal usage, 100% on one CPU core during initial rendering
Infrastructure average memory usage	150 MB

Table 1. Resource usage

8. Conclusions

As can be deduced from how the project deployment performed, provided that the network requirements for deploying such a solution are met, the usage of Wireguard in this manner is quite advantageous. For a very minimal investment, the person gets access to his/her entire internal network, while not being present in the network itself. This tunnel could provide access to a network-attached storage unit to view files you store online, perform software-based maintenance on remote connection. It is also crucial to highlight how portable this solution is, due to the sheer size of the hardware that is being utilised and the added flexibility in rebuilding the infrastructure when required. But this solution is of course not ideal for numerous reasons. One being that the specifications of the system-on-a-chip included with the Raspberry Pi Zero 2 W are, while admittedly far more capable than the previous iteration of the device, quite rudimentary. If a user is ever so inclined as to deploy extra solutions on the same device, he would have to use a Raspberry Pi 4 instead of this SBC, which has a much more powerful CPU than the Zero 2. Considering all these facts, this solution is still viable if the user is willing to pursue a budget-conscious and highly configurable VPN tunnel.

9. References

1. https://www.arm.com/architecture/cpu (visited 01/05/2022)

2. https://www.raspberrypi.com/products/raspberry-pi-zero-2-w/ (visited 01/05/2022)

3. https://www.raspberrypi.com/documentation/computers/configuration.html#setting-up-a-headless-raspberry-pi (visited 01/05/2022)

4. https://www.raspberrypi.com/documentation/computers/os.html#introduction (visited 01/05/2022)

5. https://www.ansible.com/overview/how-ansible-works?hsLang=en-us (visited 01/05/2022)

6. https://afteracademy.com/blog/what-is-a-vpn-explain-its-working (visited 01/05/2022)

7. https://surfshark.com/blog/what-is-openvpn (visited 01/05/2022)

8. https://www.cactusvpn.com/beginners-guide-to-vpn/what-is-ikev2/ (visited 01/05/2022)

9. https://www.wireguard.com/protocol/#primitives (visited 01/05/2022)

10. https://openvpn.net/community-resources/how-to/#connecting-to-a-samba-share-over-openvpn (visited 01/05/2022)

11. https://nordvpn.com/blog/wireguard-simplicity-efficiency/ (visited 01/05/2022)

12. https://www.fortinet.com/solutions/mobile-carrier/4g-5g/carrier-grade-nat (visited 01/05/2022)

13. https://www.raspberrypi.com/software/ (visited 01/05/2022)

14. https://github.com/linuxserver/docker-wireguard (visited 01/05/2022)

15. https://www.asus.com/support/faq/1011725/ (visited 01/05/2022)

16. https://prometheus.io/docs/introduction/overview/ (visited 01/05/2022)

17. https://prometheus.io/docs/introduction/first_steps/ (visited 01/05/2022)

18. https://grafana.com/docs/grafana/latest/administration/configuration/ (visited 01/05/2022)

19. https://github.com/Gaunsessa/namecheap-ddns (visited 01/05/2022)

20. https://csrc.nist.gov/csrc/media/publications/fips/199/final/documents/fips-pub-199-final.pdf (visited 01/05/2022)

21. https://docs.microsoft.com/en-us/windows/wsl/install (visited 01/05/2022)

22. https://reemo.io (visited 01/05/2022)

Appendix 1 – Non-exclusive licence for reproduction and publication of a graduation thesis[1]

1 The non-exclusive licence is not valid during the validity of access restriction indicated in the student's application for restriction on access to the graduation thesis
that has been signed by the school's dean, except in case of the university's right to reproduce the thesis for preservation purposes only. If a graduation thesis is based
on the joint creative activity of two or more persons and the co-author(s) has/have not granted, by the set deadline, the student defending his/her graduation thesis
consent to reproduce and publish the graduation thesis in compliance with clauses 1.1 and 1.2 of the non-exclusive licence, the non-exclusive license shall not be
valid for the period.