VPN Implementation and Benchmarking Tool User Guide

This guide details the configuration and running instructions for the VPN, including the options of different encryption standards and performance benchmarking process.

To start, it is required that Docker is installed on your system. To install Docker, navigate to the official Docker documentation webpage which will help you determine the best version to install for your system.

This guide assumes that the VPN project files have been downloaded and extracted to some location on the system.

Environment Information

The project includes a QUIC implementation, an RSA implementation, and an X25519 + AES-256-GCM implementation.

Technology	Directory Location	Client Container Name	Router Container Name
QUIC	`VPN/QUIC`	QUIC-client-10.9.0.5	QUIC-server-router
RSA	`VPN/RSA`	RSA-client-10.9.0.5	RSA-server-router
X25519 + AES-256-GCM	`VPN/X25519`	X25519-client-10.9.0.5	X25519-server-router

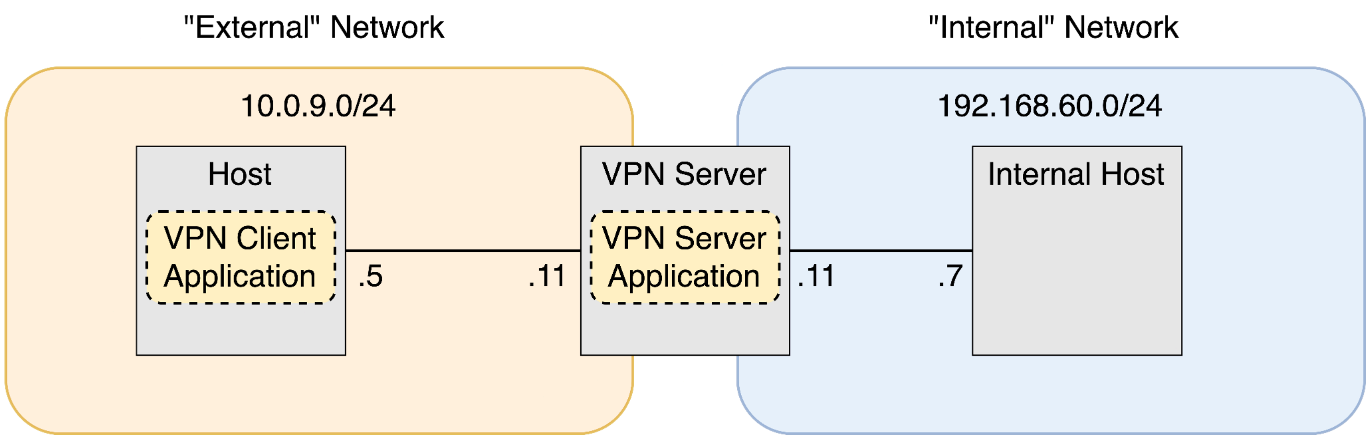

Figure 1: Docker system architecture of VPN containers and networking,

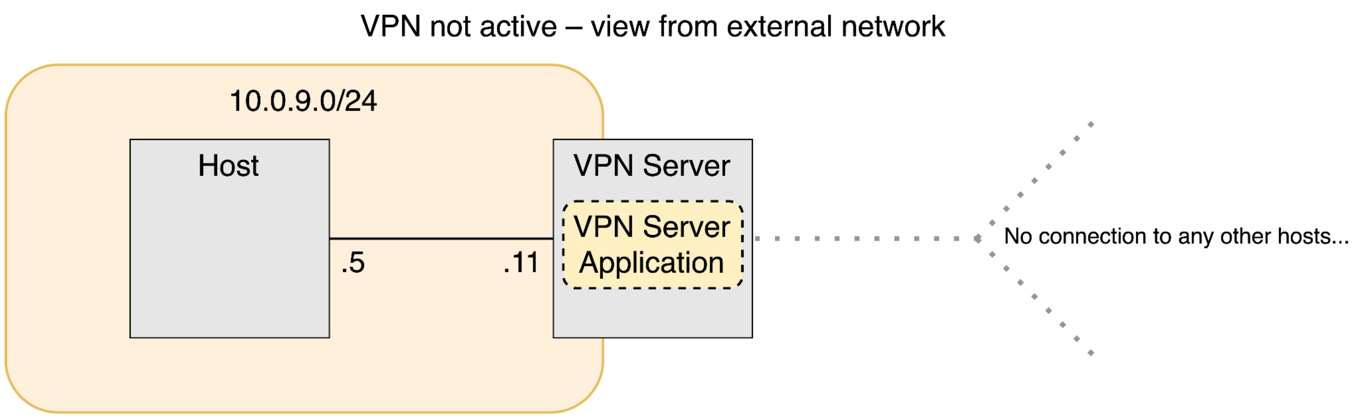
VPN not active – view from external network
10.0.9.0/24
Host
.5
.11
VPN Server
VPN Server Application
No connection to any other hosts...

Figure 2: View from external network when the VPN client is not running.

Building and running the environment

Open a terminal window (e.g. PowerShell, Bash, Command Prompt) inside the folder where the project files were downloaded. This guide will use the example locations of `C:\VPN` for Windows systems and `~/VPN` for UNIX–like systems. Adjust to match your own configuration if required. Enter the location of the encryption technology desired.

Windows

```
cd C:\VPN\{tech}
```

Linux and macOS

```
cd ~/VPN/{tech}
```

For QUIC, replace `{tech}` with `QUIC`.

Generating encryption keys (RSA and X25519 only)

If QUIC is chosen, continue to the next section of this guide.

The server and client both need their own public and private keys to facilitate the encryption and decryption of network packets securely. These can be generated in the command line if the `openssl` command is available on your operating system. macOS and most Linux distributions will come with OpenSSL installed, whereas for Windows it should be installed manually. As it comes included with the Git for Windows installer, it is recommended to install this.

Windows, Linux, and macOS

```
mkdir keys
```

RSA only

```
openssl genrsa -out keys/server_private.pem 2048
openssl rsa -in keys/server_private.pem -outform PEM -pubout -out keys/server_public.pem
openssl genrsa -out keys/client_private.pem 2048
openssl rsa -in keys/client_private.pem -outform PEM -pubout -out keys/client_public.pem
```

X25519 only

```
openssl genpkey -algorithm X25519 -out keys/x-server_private.pem
openssl pkey -in keys/x-server_private.pem -pubout -out keys/x-server_public.pem
openssl genpkey -algorithm X25519 -out keys/x-client_private.pem
openssl pkey -in keys/x-client_private.pem -pubout -out keys/x-client_public.pem
```

Building and starting the Docker environment

The following commands will create Docker containers with the tools required to run the VPN preconfigured. The three containers are:

- Client – the client machine.
- Router – the VPN server.
- Host – a client on the router's private network used to show external connections routed through the VPN can communicate with the private network.

Windows, Linux, and macOS

```
docker compose build
docker compose up -d
```

Testing connectivity with VPN off

The following command will execute a command on the client container which will attempt to connect to the host container. Replace `{client}` with the container name outlined in the environment information.

Windows, Linux, and macOS

```
docker exec -it {client} ping 192.168.60.7
```

Connections should fail to be established.

Starting the VPN client and server

Each container is already up and running as the environment has been started. However, the VPN client and server software are not running on each container yet. Run the following commands to start them, replacing the curly braces with names from the environment information.

Windows, Linux, and macOS

```
docker exec -it {client} env PYTHONPATH=/volumes python3 /volumes/client/client.py &
docker exec -it {router} env PYTHONPATH=/volumes python3 /volumes/server/server.py &
```

Testing connectivity with VPN on

As executed with the VPN off, run the same command to test a connection. Replace `{client}` with the container name outlined in the environment information.

Windows, Linux, and macOS

```
docker exec -it {client} ping 192.168.60.7
```

This time, connections should run successfully.

Benchmarking

Basic benchmarking can be conducted to test the performance of each variant available of the VPN implementation.

Prerequisites:

- RSA keys generated as per this guide
- X25519 keys generated as per this guide

It is not required, but however highly recommended, to test each VPN implementation to ensure they have been configured correctly before running the benchmarking.

The following bash script if run in the `/VPN` directory will automatically test each implementation. This can be executed in Git Bash on Windows (this guide recommends installing it to use the `openssl` command) if you are unsure on how to execute it.

Bash script (.sh file)

```
for dir in $(ls -d */ | sed 's/\///g');
do
    pushd .
    cd $dir
    docker-compose build
    docker-compose up -d
    docker exec -itd ${dir}-client-10.9.0.5 env PYTHONPATH=/volumes python3
/volumes/client/client.py
    docker exec -itd ${dir}-server-router env PYTHONPATH=/volumes python3 /volumes/server/server.py
    python3 benchmark/run_tests.py >> benchmark-output.txt
    docker-compose kill
    docker-compose down
    popd
done
```

Alternatively, a PowerShell script may be run. This is natively supported on Windows and can also be executed on Linux and macOS if PowerShell is installed. It performs the same task.

PowerShell script (.ps1 file)

```
$dirs = gci -Directory
foreach ($dir in $dirs) {
    pushd
    cd $dir.FullName
    docker-compose build
    docker-compose up -d
    $dirName=$dir.Name
    docker exec -itd "${dirName}-client-10.9.0.5" env PYTHONPATH=/volumes python3
/volumes/client/client.py
    docker exec -itd "${dirName}-server-router" env PYTHONPATH=/volumes python3
/volumes/server/server.py
    python3 benchmark/run_tests.py | Out-File -Append ./benchmark-output.txt
    docker-compose kill
    docker-compose down
    popd
}
```