PROJECT REPORT

ON

VPN SERVER USING WIREGUARD

Submitted to

KIIT Deemed to be University

In Partial Fulfilment of the Requirement for the Award

BACHELOR'S DEGREE IN COMPUTER SCIENCE &

SYSTEMS ENGINEERING

By

JYOTIRMOY CHAKRABORTY - 1728010

JYOTI MAURYA - 1728093

UNDER THE GUIDANCE OF

Teacher in-charge: Roshni Pradhan

SCHOOL OF COMPUTER ENGINEERING

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY

BHUBANESWAR, ODISHA – 751024

APRIL 2020

KIIT Deemed to be University

School of Computer Engineering
Bhubaneswar, ODISHA 751024

CERTIFICATE

This is to certify that the project entitled

"**VPN SERVER USING WIREGUARD**"

Submitted by:

JYOTIRMOY CHAKRABORTY - 1728010
JYOTI MAURYA - 1728093

is a record of Bonafede work carried out by them, in the partial fulfillment of the requirement for the award of Degree of Bachelor of Engineering (Computer Science & Systems Engineering) at KIIT Deemed to be University, Bhubaneswar. This work is done during the year 2019-2020, under our guidance.

Date: 25/ 04 / 2020

(**Roshni Pradhan)**
Project Guide

CONTENTS

Topic

ABSTRACT

WireGuard is a comparatively new VPN Technology that utilizes state-of-art technology to provide an easy VPN connecting protocol for the server and the user. The goal of the project was initially to build a simple bash script that could even more simplify the setup process for users wishing to implement the concept on their server and provide an easier client generation interface to include new clients. This was achieved by using a set of bash scripts that run the various commands that are required for the setting up process and to generate newer clients on the interface. This goal was later expanded to simplify the client signing up process with an easy to use webserver that could sign up new users to the platform. Also, further steps needed to be taken to ensure a more secure process so that unverified users do not gain access to the platform to maintain the integrity of the system. To do so, a layer of administrator issued script was prepared that had to be run manually to add the new users after verification. Various technologies already in use has been utilized in order to obtain the required result. Some of them including but not limited to are MySQL, apache, ssh.

INTRODUCTION

A Virtual Private Network is used for creating a private scope of a computer communications or providing a secure extension of a private network through an insecure network. VPN can be built upon OpenVPN, IPsec, WireGuard, etc. For the scope of this project, we chose WireGuard.

According to the documentation of WireGuard,
"WireGuard is an extremely simple yet fast and modern VPN that utilizes state-of-the-art cryptography. It aims to be faster, simpler, leaner, and more useful than IPsec, while avoiding the massive headache. It intends to be considerably more performant than OpenVPN. WireGuard is designed as a general-purpose VPN for running on embedded interfaces and super computers alike, fit for many different circumstances. Initially released for the Linux kernel, it is now cross-platform (Windows, macOS, BSD, iOS, Android) and widely deployable. It is currently under heavy development, but already it might be regarded as the most secure, easiest to use, and simplest VPN solution in the industry."

In this project, we have implemented a WireGuard server on google cloud platform and prepared a bash-script for the creation of the same on future server platforms. Further we also created an easy client creation script that generates a client configuration file that can be used by the client to connect with the server.
This was also extended to a database of users with their e-mail and their approval status. A web interface was also created to simplify the sign-up process.

1.1 FEATURES OF GOOGLE CLOUD PLATFORM

- **Machine types:**
  Compute Engine offers predefined virtual machine configurations for every need from small general-purpose instances to large memory-optimized instances with up to 11.5 TB of RAM or fast compute-optimized instances with up to 60 vCPUs. Further custom machine types can be chosen with specifications depending on needs.
- **Live Migration of VM's:**
  Compute Engine virtual machines can live-migrate between host systems without rebooting, which keeps applications running even when host systems require maintenance.
- **Global load balancing:**
  Global load-balancing technology helps distribute incoming requests across pools of instances across multiple regions, so maximum performance, throughput, and availability can be achieved at a low cost.
- **Linux and Windows support:**

It provides a huge choice of OSs' including Debian, CentOS, CoreOS, SUSE, Ubuntu, Red Hat Enterprise Linux, FreeBSD, or Windows Server 2008 R2, 2012 R2, and 2016. It also provides options for using a shared image from the Google Cloud community or even our own.

- **Convenient billing:**

  Google bills in second-level increments. Bills are only generated for the compute time that you use. Further it also provides committed use savings and sustained use savings for more discounts on regular use systems.

- **OS patch management:**

  With OS patch management, one can apply OS patches across a set of VMs, receive patch compliance data across your environments, and automate installation of OS patches across VMs - all from a centralized location.

1.2 FEATURES OF WIREGUARD

WireGuard provides the following few important features:

- **Simple and Easy to Use**

  WireGuard aims to be as easy to configure and deploy as SSH. A VPN connection is made simply by exchanging very simple public keys – exactly like exchanging SSH keys – and all the rest is transparently handled by WireGuard.

- **Cryptographically Sound**

  WireGuard uses state-of-the-art cryptography, like the Noise protocol framework, Curve25519, ChaCha20, Poly1305, BLAKE2, SipHash24, HKDF, and secure trusted constructions. It makes conservative and reasonable choices and has been reviewed by cryptographers.

- **Minimal Attack Surface**

  WireGuard has been designed with ease-of-implementation and simplicity in mind. It is meant to be easily implemented in very few lines of code, and easily auditable for security vulnerabilities.

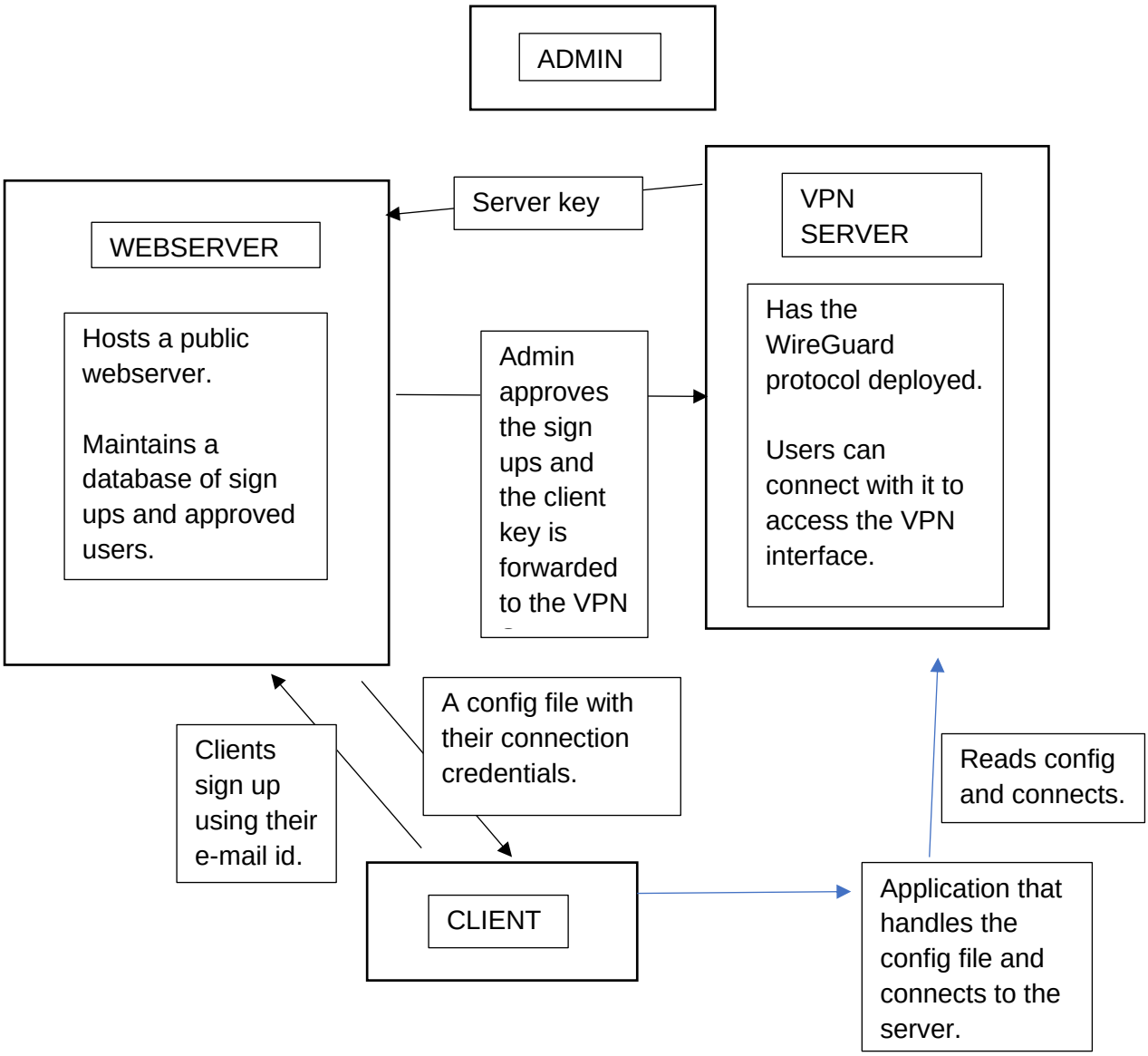
- **High Performance**

  A combination of extremely high-speed cryptographic primitives and the fact that WireGuard lives inside the Linux kernel means that secure networking can be very high-speed. It is suitable for both small embedded devices like smartphones and fully loaded backbone routers.

OBJECTIVE

To deploy a VPN Server on Google Cloud using WireGuard protocol and provide a script for client generation. And also create a webserver to sign up new users and provide an administrator level system to approve them on the platform.

PROPOSED MODEL



SYSTEM SPECIFICATIONS

We have used the following systems for the running of the server and testing of connections,

- Two virtual computer engines in Google Cloud Platform as the VPN Server and webserver
- Test Clients.

The VM in cloud was initialized with the following settings for the VPN Server:

- Region: asia-south-1 (Mumbai)
- System type: N1-type processor with 1vCPU with 675MiB RAM
- System Operating System: ubuntu 20.04 LTS minimal
- System Networking Specifications:
  - ➔ Network Adapter: ens4
  - ➔ IP Forwarding: ON
  - ➔ Custom Open ports:
    51820 (WireGuard igress connection)
    53 (for DNS traffic)
    20/22 (for ssh connections)

The VM in cloud was initialized with the following settings for the webserver:

- Region: asia-south-1 (Mumbai)
- System type: N1-type processor with 1vCPU with 1.7GiB RAM
- System Operating System: ubuntu 20.04 LTS minimal
- System Networking Specifications:
  - ➔ Network Adapter: ens4
  - ➔ IP Forwarding: OFF
  - ➔ Custom Open ports:
    80 (for HTTP connections)
    20/22 (for SSH connections)

Test client systems had the following specifications:

- Mobile device running android 10 and LTE connection.
- Mobile device running android 10 over Wi-Fi connection.
- Mobile device running android 8.0 over LTE connection.
- Laptop PC running Windows 10 Home.
- Laptop PC running Arch Linux.

IMPLEMENTATION AND RESULT

VPNSERVER

The first virtual machine was initialized with a set of scripts which have been now published under open source in https://www.github.com/jyotirmoy1904/vpnserver-setup.sh .
These scripts were written in bash to be implemented in a Debian based Linux system preferably ubuntu 19.10 and above. The entire script was primarily divided into five sub-scripts which was made executable using the driver script.

1. driver.sh
Initializes the environment for installation and executes the scripts in a sequential manner.

2. Installer.sh
It checks the eligibility of the system and installs the WireGuard protocol and necessary components on the systems.

3. serversetup.sh
It generates the various server-keys and initializes the interface. It is also responsible for setting up the IP Forwarding and firewall rules on the server to enable client connections.

4. client_generate.sh
It creates a client profile and generates a configuration file for the client to use to initialize the system. Further it also enables the client connection on the WireGuard interface.

5. dns_settings.sh
An initial plan was also to setup the DNS server for network access on the same server. However, this could not be achieved and the client connections to the internet failed when using the server DNS. For that reason, the DNS server was later changed to Google Public DNS Server.

6. database_setup.sh
This script initializes the MySQL database with a default user, database, and table so that new entries from the client can be incorporated into the database to maintain a record of clients.

For the scope of this project, only the driver.sh runs only the installer.sh, serversetup.sh and the dns_settings.sh scripts as the webserver was used to generate new client configurations and manage the databases to distribute load amongst the systems.

The server_public_key which was generated in the server_setup.sh process was later transferred to the webserver so that it could be used to setup the client configs.

Later, a new script was added, which could be executed from the webserver to sign up new users as and when approved by the admin.

WEBSERVER

The webserver was designed to handle new user connections using user-provided email ID. HTML and PHP was used to handle the user-input and pass it to a bash script which generated the configuration and returned the script to the user. Further, the script also entered each new entry into a SQL database with the client public key, email, date, and approved status (initialized to 0). The HTML form data was processed into the PHP using POST Request method.

The database_setup script which was initially designed for the VPNServer was later used in the webserver and the client_generate.sh script was modified for use in the webserver instead.

In addition,
A new *approve.sh* script was written, which can enable the administrator to check new entries and approve them or reject them. Approving then, forwarded the client_key to the VPNServer and rejection deleted their entries from the database.

TEST SCENARIO

The execution of the above methods and the clients generated the following configuration files and the database entries.

APPROVAL PROCESS

```
jyoti@website-vpn:~$ ./approve.sh
Unapproved Client requests:
Enter password:
+-----------+--------------------------+--------------------------------------------+---------+--------------------+------------+----------+
| client_id | client_email             | client_key                                 | last_ip | config_file_name   | date       | Approved |
+-----------+--------------------------+--------------------------------------------+---------+--------------------+------------+----------+
|         2 | ijyotimaurya150@gmail.com | 3+2XkL5WzHNeVFZ41IQ/BYSP6XxNaIlPo/lY/KSswzk= | 0.0.0.0 | 3+2XkL5WzHNeVFZ.conf | 2020-05-25 |        0 |
|         3 | test@xywa.com            | 9Pf/NmnYBh8evgBo7er/68CU6jFNANPM6Y4WY+XgXRU= | 0.0.0.0 | 9PfNmnYBh8evgB.conf  | 2020-05-25 |        0 |
+-----------+--------------------------+--------------------------------------------+---------+--------------------+------------+----------+


Select the Client to approve
2
Enter password:
+-----------+--------------------------+--------------------------------------------+---------+--------------------+------------+----------+
| client_id | client_email             | client_key                                 | last_ip | config_file_name   | date       | Approved |
+-----------+--------------------------+--------------------------------------------+---------+--------------------+------------+----------+
|         2 | ijyotimaurya150@gmail.com | 3+2XkL5WzHNeVFZ41IQ/BYSP6XxNaIlPo/lY/KSswzk= | 0.0.0.0 | 3+2XkL5WzHNeVFZ.conf | 2020-05-25 |        0 |
+-----------+--------------------------+--------------------------------------------+---------+--------------------+------------+----------+
To Approve, enter y
y
APPROVING...
Enter password:
Approved 3+2XkL5WzHNeVFZ41IQ/BYSP6XxNaIlPo/lY/KSswzk=
Enter password:
```

DATABASE AFTER APPROVAL

```
jyoti@website-vpn:~$ mysql -uroot -p -e"SELECT * FROM vpn.clients"
Enter password:
+-----------+--------------------------+--------------------------------------------+---------+--------------------+------------+----------+
| client_id | client_email             | client_key                                 | last_ip | config_file_name   | date       | Approved |
+-----------+--------------------------+--------------------------------------------+---------+--------------------+------------+----------+
|         2 | ijyotimaurya150@gmail.com | 3+2XkL5WzHNeVFZ41IQ/BYSP6XxNaIlPo/lY/KSswzk= | 0.0.0.0 | 3+2XkL5WzHNeVFZ.conf | 2020-05-25 |        1 |
|         3 | test@xywa.com            | 9Pf/NmnYBh8evgBo7er/68CU6jFNANPM6Y4WY+XgXRU= | 0.0.0.0 | 9PfNmnYBh8evgB.conf  | 2020-05-25 |        0 |
|         1 | jyotirmoy1904@gmail.com  | NnaDgDeisBYsYrlNdhdtq6doVr6yaMtJAupcHZcoyjk= | 0.0.0.0 | NnaDgDeisBYsYrl.conf | 2020-05-25 |        1 |
+-----------+--------------------------+--------------------------------------------+---------+--------------------+------------+----------+
jyoti@website-vpn:~$
```

CLIENT CONFIGURATIONS:

Client-3:

```
[Interface]
Address = 10.200.200.4/24
PrivateKey = mB[illegible]
DNS = 8.8.8.8

[Peer]
PublicKey = 6WBgdabZsiDXpabWcyw9KR+jtL+b06T90jo5mpNXRj8=
Endpoint = 35.244.40.221:51820
AllowedIPs = 0.0.0.0/0
PersistentKeepalive = 21
```

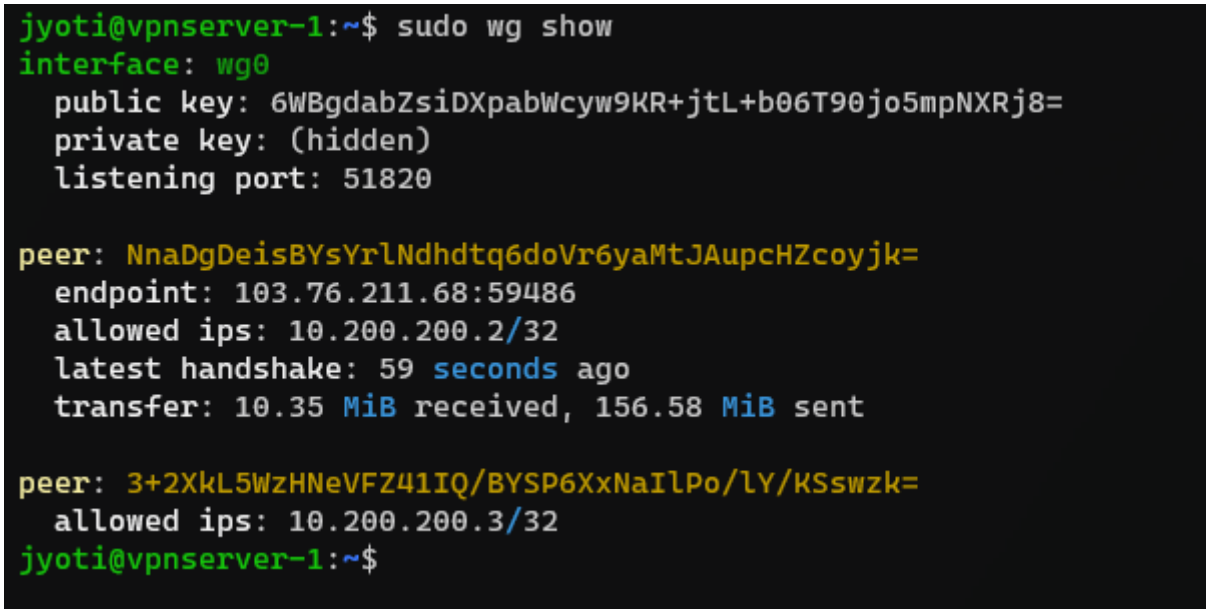
Client-2:

```
[Interface]
Address = 10.200.200.3/24
PrivateKey = WHCm[illegible]
DNS = 8.8.8.8

[Peer]
PublicKey = 6WBgdabZsiDXpabWcyw9KR+jtL+b06T90jo5mpNXRj8=
Endpoint = 35.244.40.221:51820
AllowedIPs = 0.0.0.0/0
PersistentKeepalive = 21
```

Client-1:

```
[Interface]
Address = 10.200.200.2/24
PrivateKey = wNgy[illegible]
DNS = 8.8.8.8

[Peer]
PublicKey = 6WBgdabZsiDXpabWcyw9KR+jtL+b06T90jo5mpNXRj8=
Endpoint = 35.244.40.221:51820
AllowedIPs = 0.0.0.0/0
PersistentKeepalive = 21
```

VPNSERVER WG STATUS:



endpoint shows the last IP address the client connected from, latest handshake shows when was the last connection made and transfer shows the total amount of data transferred between the peers.
There are two peer entries as, the third one was not approved and as the second peer has not yet initialized a connection with the server, there are no entries for it.

RESULT AND DISCUSSION

The initial goal of the project was to develop the script for the server setup and client configuration generation for WireGuard protocol. However, the applications of this project include an entire VPN Provider service management. This project can also be further modified in future to include more protocols, ensure more security and also develop a virtual environment for a user with different systems and maintain data easily and securely between them.

FUTURE SCOPE

The project was initially intended with the basic objective as described above. However, it is planned to further increase the scope of the project to provide a better DNS server setup, cleaner scripts, more protocol choices to the user and an interactive graphical based client interface to the administrator for managing clients. Our future advancements to the project include deploying the database on a different server to ensure even more security and a cleaner interface etc.

CONCLUSION

During the development of this project, we came across the wide scope of applications of various network tools and the development of software technologies. We used a basic breakdown of steps used to deploy a server to develop a script for future implementations and a webserver to manage a product sign-up. While doing so we learned of practical applications of the various phases of the software engineering life development process namely: planning, analysis, design, development, and implementation, testing and maintenance. We hand on experimented with database management concepts, ssh connections, remote management of servers, and executing scripts on a different system from one system. By further using GitHub to manage and co-develop the source code, we understood of various uses of git for source code control and monitoring co-development progress. Further, we understood the importance of teamwork to help us further our own knowledge and learn new things.

CONTRIBUTIONS OF GROUP MEMBERS:

1. Jyotirmoy Chakraborty, 1728010@kiit.ac.in:
   Was responsible for designing the WireGuard setup and initialization script and debugging the servers, manage the intercommunication between the servers and writing the bash scripts for the project. Was also responsible for the project paper and contributed to the testing of the final result.

2. Jyoti Maurya, 1728093@kiit.ac.in
   Was responsible for designing the webserver and deploying and managing the database. Had important contributions in the writing of the bash scripts and to the research paper editing and discussing the future scopes of the project. She also played an important role in further testing of the result codes and stress operation of the PHP handing within different constraints.

ACKNOWLEGEMENTS AND REFERENCES

This project is a group work and would not have been possible without the contributions of both the members. In addition, the following sites were referred to during various steps of the project execution which includes various other people without whom this project would not have been possible.

Reference Journals:

Static link: wireguard.com/papers/wireguard.pdf
Date: June 30, 2018. A version of this paper appears in *Proceedings of the Network and Distributed System Security Symposium, NDSS 2017*. Copyright © 2015–2017 Jason A. Donenfeld. All Rights Reserved.

Other sites referred to during the setup process:

- https://httpd.apache.org/
- https://dev.mysql.com/doc/refman/8.0/en/
- https://github.com/iamckn/wireguard_ansible
- https://freedif.org/unbound-your-own-dns-server
- https://cloud.google.com/docs/overview
- https://stackoverflow.com/questions/tagged/bas

SOURCE CODE

The source code for the VPNServer is available at:
htttps://www.github.com/jyotirmoy1904/vpnserver-setup.git

The other codes for the webserver are:

1. index.html

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width, initial-scale=1">
        <title>Main Page</title>
        <link rel="stylesheet" href="style.css">
    </head>
      <body>
        <table cellpadding="10">
            <tr>
                <th><b>inryo </b></th>
            </tr>
        </table>
         <br>
         <br>
         <br>
         <h1><b>Wireguard VPN Service</b></h1>
         <br>
          <p>
            <b>
                Project github <a href="https://github.com/jyotirmoy1904/vpnserver-setup" target="_blank">link </a>
                <br>
            </b>
            <br>
            <b>
                Sign-
up to our VPN Service <a href="page_2.html"><u>here</u></a>
            <br></b>
            <br>
            <b>
                Contact us
            <br></b>
          </p>
      </body>
</html>j
```

2. page_2.html

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title></title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <table cellpadding="10">
      <tr>
          <th><b>inryo </b></th>
      </tr>
  </table>

     <form action="generate.php" method="post" >
       Enter E-
mail id : <input type="text" name="email" placeholder="Write your email addres
s">
       <br>
       <br>

      <button type="submit" name = "submit">Generate config</button>
    </form>
    <p>
      You will receive an email when the downloaded config file is ready to us
e.<br>
    </p>
  </body>
```

3. generate.php

```
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title></title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <table cellpadding="10">
      <tr>
          <th><b>inryo </b></th>
      </tr>
    </table>
    <br>
    <br>
        <?php
                $email = $_POST['email'];
                $command = 'bash /var/www/html/client_generate.sh '.$email.'';
                $output = shell_exec($command);?>
                <a href="<?php echo $output;?>">Download config</a>
                <br>
                Right click on "Download Config" and select SaveAs to download
 the configuration file.
                For further instructions <a href="wiki.html"> Click here </a>
</body>
</html>
```

4. client_generate.sh

```
date=$(date +%F)
##Generating client keys
email=$1
workdir="vpnserver-wireguard"
check2=$(ls | grep "flag")
while  [[ ! -z $check2  ]]
  do
    sleep 0.5
  done
touch flag
client=$(cat $workdir/client_qty)
peer=$((client+1))
echo $peer > $workdir/client_qty
rm -f flag
client_private_key=$(wg genkey)
client_public_key=$(echo $client_private_key | wg pubkey)
peer=$((client+1))
server_public_key=$(cat $workdir/server_public_key)
file_name=$(echo $client_public_key | cut -c1-15 | tr -d /)
##Generating client configs
touch $file_name.conf
echo "[Interface]
Address = 10.200.200.$peer/24
PrivateKey = $client_private_key
DNS = 8.8.8.8" > $file_name.conf
echo "
[Peer]
PublicKey = $server_public_key
Endpoint = <server_ip>:51820
AllowedIPs = 0.0.0.0/0
PersistentKeepalive = 21" >> $file_name.conf
mysql --defaults-extra-file=.my.cnf -D vpn -
   e "INSERT INTO clients VALUES($client, '$email', '$client_public_key',
   '0.0.0.0', '$file_name.conf', '$date', 0);"
echo "$file_name.conf"
```