Basic Principles in Networking
Assignment 7 - IPsec & VPN
Pair 29:
Nguyen Xuan Binh 887799
Nhut Cao 906939

Section 1: Goals of the experiment

VPN (Virtual Private Network) protects user information by masking their device's IP address, encrypting their data packets and routing them through secure networks to the intended servers around the world. In doing so, it hides the users online identity, ensuring that they are able to browse the Internet securely and anonymously.

In this experiment, we will set up a chat server where clients can send messages to the chat server over a protected network by Wireguard. WireGuard is an open-source communication VPN protocol that implements encrypted VPN tunneling.

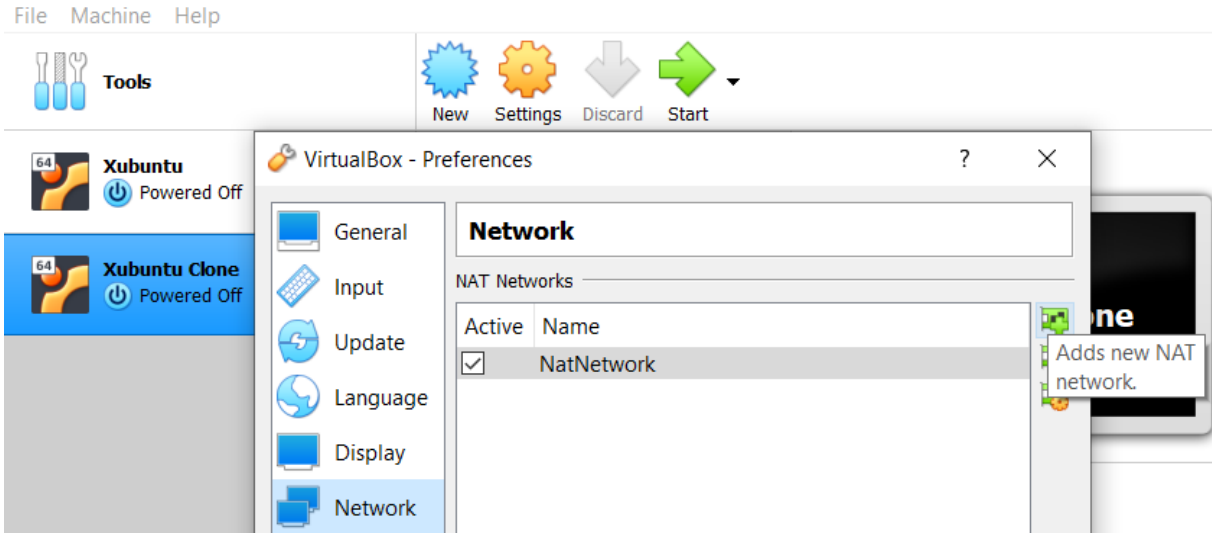
Section 2: Experimental Setup (Details of the experimental setup step by step)

Stage 1: Setting up the Virtual Machines and the NAT network

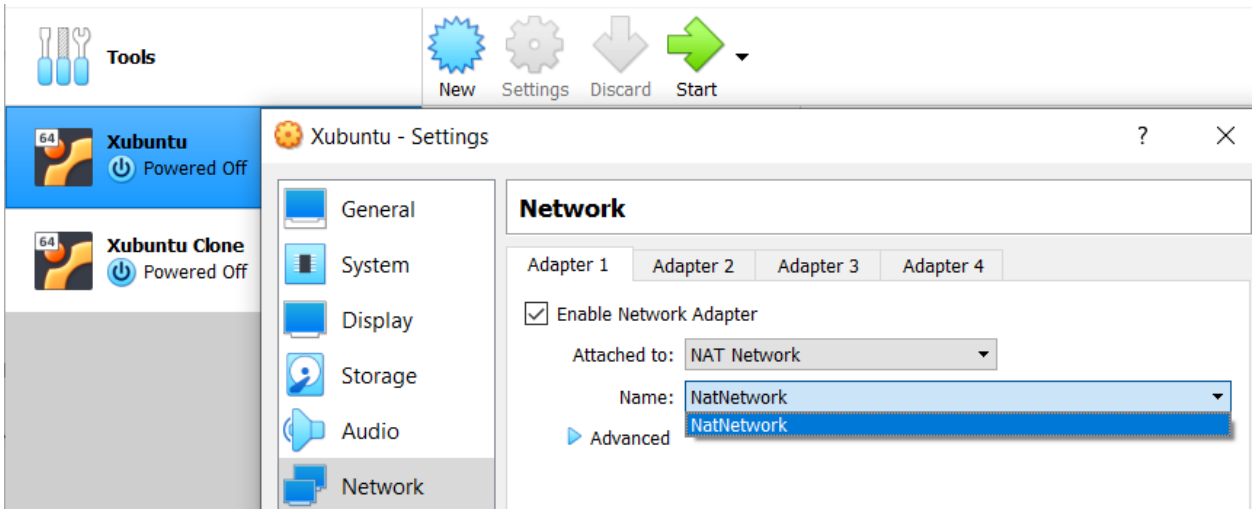
In this experiment, since we do not have 2 laptop devices, we will use 2 virtual machines (VM) to simulate two different computers instead. First of all, we set up two Ubuntu Virtual Box, which are Xubuntu as the server VM and Xubuntu Clone as the client VM.

For the VMs to share the common network, they should be behind the same NAT.

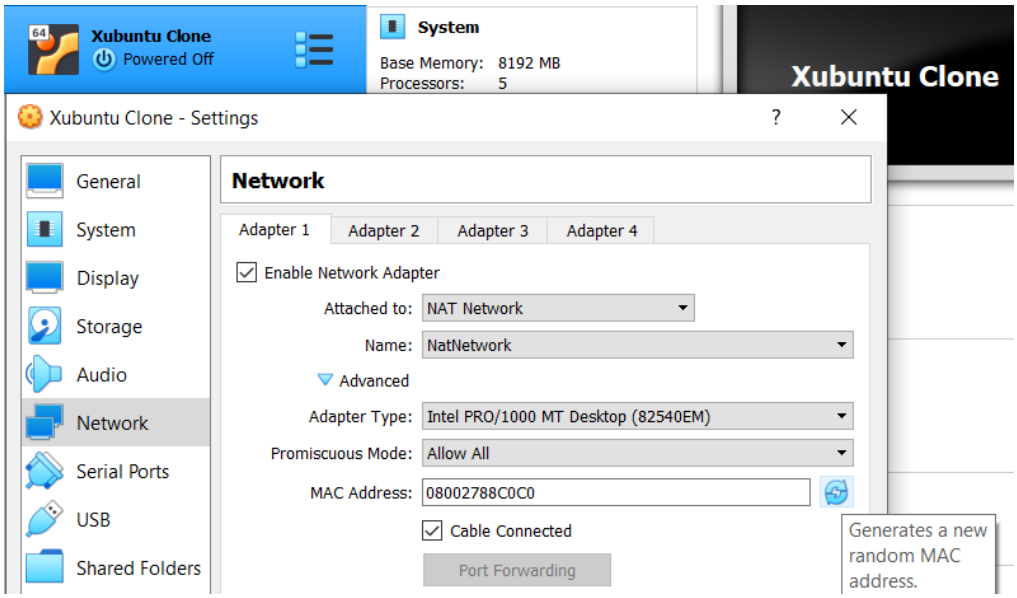
First, go to File > Preferences > Networks > Adds new NAT network. Now, there is a NAT network that can be shared across the VM.



Now we have to put the VM behind the created NAT network. Right click on Xubuntu > Settings > Network. Under Attached to, choose NAT Network. For Name, choose NatNework which was created in the above instructions. Next, do the same thing for Xubuntu Clone as well.



However, as Xubuntu Clone is the clone of Xubuntu, they will share the same IP address and MAC address, which may not demonstrate the feature of VPN as they are “the same” machine. To change the MAC and IP address of the clone VM, right click on Xubuntu Clone > Settings > Network > Expand Advanced > Generates a new random MAC address. This will change both the MAC and public IPv4 of the Clone VM => The server and client VM are two different stations



Note: to make changes to settings of VM, they must be powered off. Changes can’t be made when VM are running.

Stage 2: Setting up WireGuard VPN tunneling between the server and client VM

- On both the server VM (Xubuntu) and client VM (Xubuntu Clone), we install WireGuard, setting up their key pairs and register them to the WireGuard VPN. Instructions followed from: https://www.youtube.com/watch?v=bVKNSf1p1d0&ab_channel=TheDigitalLife

Commands Purpose	Server VM	Client VM
Installing WireGuard	sudo apt install wireguard	sudo apt install wireguard
Generate the public-private key pair	wg genkey | tee privatekey | wg pubkey > publickey	wg genkey | tee privatekey | wg pubkey > publickey
View the public key	cat publickey	cat publickey
(the public key)	8ANLSQ+Hqv684z+eF6rNk7fi6r24mnTwtqpdkZZ5ExQ=	1fc4OWsfIG2XzBD2HqY92J3xbNLIn4ObBAgEVCx403U=
View the private key	cat privatekey	cat privatekey
(the private key)	eN4zEznryYpnBSp4BMgPBRaHKopcDlf5Jqv8QK3/m20=	iJ2iocUGHmPEJDp3NEktQrofTm/v1A8o7f6Hk2S4omk=
View the interface enp0s3	ip addr show enp0s3 // or ifconfig	ip addr show enp0s3 // or ifconfig
(public IPv4)	inet 10.0.2.15/24	inet 10.0.2.4
Create Wireguard config file in Vim	sudo vim /etc/wireguard/wg0.conf	sudo vim /etc/wireguard/wg0.conf

Use Vim text editor, press I to enter Insert mode, paste the text, then Esc > :wq to save and return to Terminal.

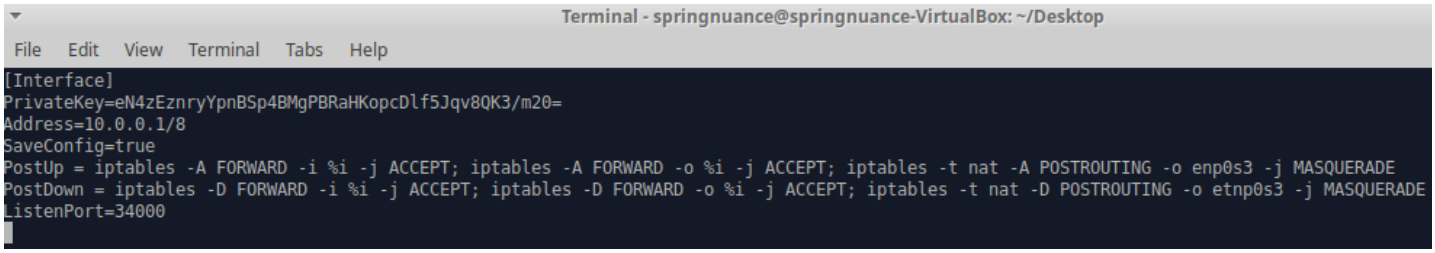
For the Server VM

[Interface]
PrivateKey=eN4zEznryYpnBSp4BMgPBRaHKopcDlf5Jqv8QK3/m20=
Address=10.0.0.1/8
SaveConfig=true
PostUp = iptables -A FORWARD -i %i -j ACCEPT; iptables -A FORWARD -o %i -j ACCEPT; iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE
PostDown = iptables -D FORWARD -i %i -j ACCEPT; iptables -D FORWARD -o %i -j ACCEPT; iptables -t nat -D POSTROUTING -o etnp0s3 -j MASQUERADE
ListenPort=34000
In more details below in the VIM Text editor

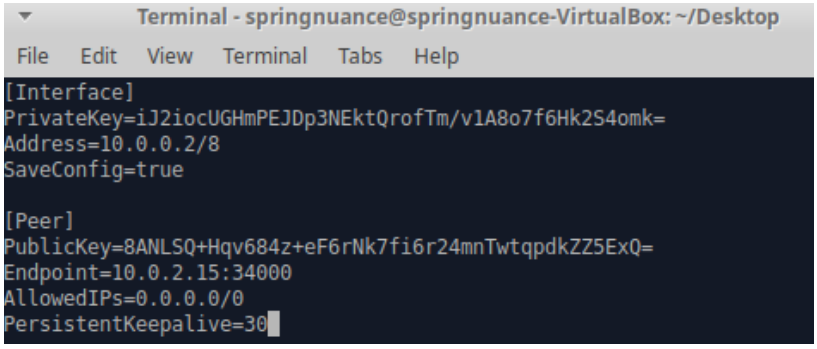
For the Client VM

[Interface]
PrivateKey=iJ2iocUGHmPEJDp3NEktQrofTm/v1A8o7f6Hk2S4omk=
Address=10.0.0.2/8
SaveConfig=true
[Peer]
PublicKey=8ANLSQ+Hqv684z+eF6rNk7fi6r24mnTwtqpdkZZ5ExQ=
Endpoint=10.0.2.15:34000
AllowedIPs=0.0.0.0/0
PersistentKeepalive=30

In Server VM:
PrivateKey is the private key of the server. Address is the default address for wg0, which is 10.0.0.1/8 for the first endpoint. ListenPort can be any port



In Client VM:
Private key is the private key of the client. Address is the default address for wg0, which is 10.0.0.2/8 for the second endpoint. For [Peer] part, PublicKey is the public key of the server VM and endpoint is the public IPv4 of the server VM. AllowedIPs is 0.0.0.0/0 for simply routing all connections via the server.



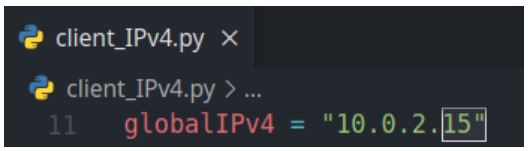
***** Connecting the Client and Server into the VPN *****
In Server VM, we add the client VM to the server VM via WireGuard with command:
sudo wg set wg0 peer 1fc4OWsfIG2XzBD2HqY92J3xbNLln4ObBAgEVCx403U= allowed-ips 10.0.0.2/32
where 1fc4….403U is the public key and allowed-ips is the public IPv4 of the client VM.
Everything is set up as clients can connect to the server via VPN. We can start using WireGuard

Section 3: Results & Conclusion

➢ Implementation of server and client

For the sake of simplicity, we use the previous implementation of the Chat Application, where clients can chat with each other via a chat server.
We will connect the chat server to the IPv4 of the server VM

On the client side, we also connect to the IPv4 of the server VM

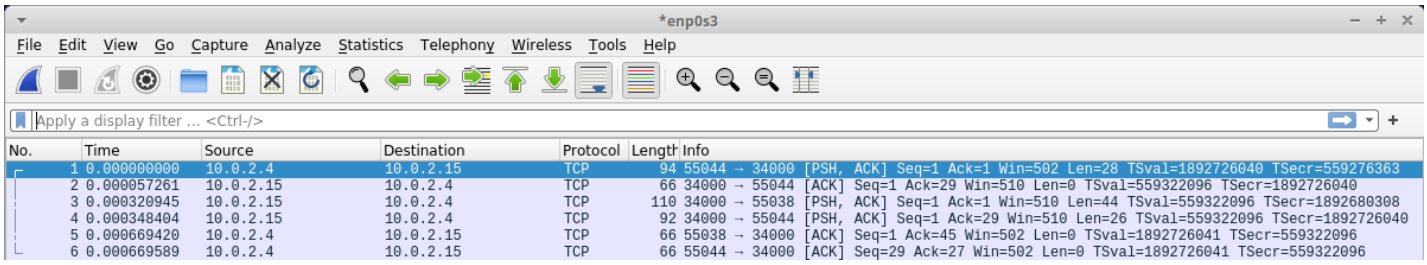


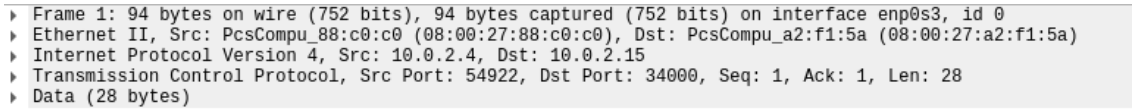
We now run the chat server on Xubuntu and connects to it 2 clients on Xubuntu Clone

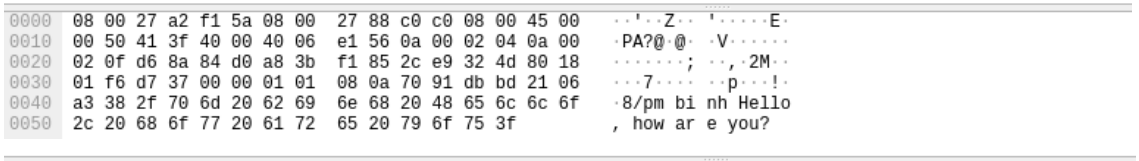
```
springnuance@springnuance-VirtualBox:~/Desktop/Instant Message App Termina
l Command version$ /bin/python3 "/hom
e/springnuance/Desktop/Instant Message App Terminal Command version/server.py"
Waiting for the clients to connect...
```

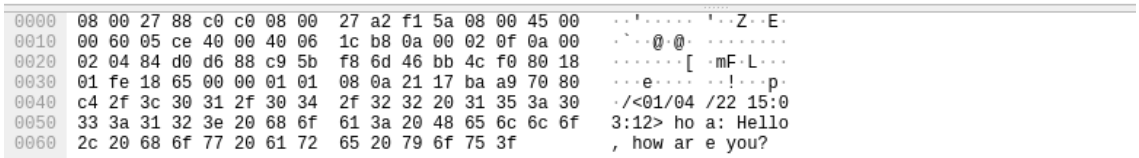
```
springnuance@springnuance-VirtualBox:~/Desktop/Instant Message App Terminal Command version$ python3 client_IPv4.py
>>> Welcome to the chat server!

First, please enter your name to register/login
hoa
>>>
Send '/command' to see instructions on how to use the chat application
hoa has joined the server
/pm binh Hello, how are you?
>>> binh has seen your message
```

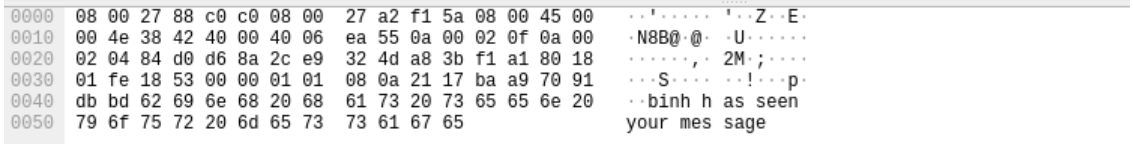
There are now 2 clients named Binh and Hoa. Hoa has sent a message to Binh and we want to observe how the message is sent over the network. First we can turn off wg0 VPN by: **wg-quick down wg0** for both server and client VM**,** if they are already turned on. Then we run Wireshark inside the server VM to capture the packets between the server and the clients with: **sudo wireshark** (wireshark will not capture network interfaces without superuser privilege). Wireshark has captured several TCP packets which are the transmission protocol of the chat application









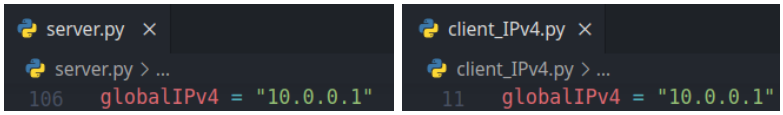


As we can observed, the plaintext data sent over the network are not encrypted,

=> Without WireGuard VPN, the data packets are not encrypted and authenticated at the endpoints at all, which can be susceptible to be misused by unintended third parties

➢ **Communication between server and client using VPN**

Because the IP address of the server WireGuard VPN is 10.0.0.1, we will connect the chat server and clients to “10.0.0.1”, or connect via the VPN.



Next, we turn on WireGuard wp0 VPN by the command: **wg-quick up wg0.** Now WireGuard VPN tunnel will start receiving data from the clients and route it to the server and vice versa through a protected network.

On the server VM

```
springnuance@springnuance-VirtualBox:~/Desktop$ wg-quick up wg0
[#] ip link add wg0 type wireguard
[#] wg setconf wg0 /dev/fd/63
[#] ip -4 address add 10.0.0.1/8 dev wg0
[#] ip link set mtu 1420 up dev wg0
[#] iptables -A FORWARD -i wg0 -j ACCEPT; iptables -A FORWARD -o wg0 -j ACCEPT;
iptables -t nat -A POSTROUTING -o enp0s3 -j MASQUERADE
```

On the client VM

```
springnuance@springnuance-VirtualBox:~/Desktop$ wg-quick up wg0
wg-quick must be run as root. Please enter the password for springnuance to continue:
[#] ip link add wg0 type wireguard
[#] wg setconf wg0 /dev/fd/63
[#] ip -4 address add 10.0.0.2/8 dev wg0
[#] ip link set mtu 1420 up dev wg0
[#] wg set wg0 fwmark 51820
[#] ip -4 route add 0.0.0.0/0 dev wg0 table 51820
[#] ip -4 rule add not fwmark 51820 table 51820
[#] ip -4 rule add table main suppress_prefixlength 0
[#] sysctl -q net.ipv4.conf.all.src_valid_mark=1
[#] iptables-restore -n
```

Now to check status of wg0 interface on both VM, we can use command: **sudo wg**

On the server VM

```
springnuance@springnuance-VirtualBox:~/Desktop$ sudo wg
interface: wg0
  public key: 8ANLSQ+Hqv684z+eF6rNk7fi6r24mnTwtqpdkZZ5ExQ=
  private key: (hidden)
  listening port: 34000

peer: 1fc4OWsfIG2XzBD2HqY92J3xbNLIn4ObBAgEVCx403U=
  endpoint: 10.0.2.4:47951
  allowed ips: 10.0.0.2/32
  latest handshake: 1 minute, 1 second ago
  transfer: 2.10 KiB received, 2.34 KiB sent
```

On the client VM

```
springnuance@springnuance-VirtualBox:~/Desktop$ sudo wg
interface: wg0
  public key: 1fc4OWsfIG2XzBD2HqY92J3xbNLIn4ObBAgEVCx403U=
  private key: (hidden)
  listening port: 47951
  fwmark: 0xca6c

peer: 8ANLSQ+Hqv684z+eF6rNk7fi6r24mnTwtqpdkZZ5ExQ=
  endpoint: 10.0.2.15:34000
  allowed ips: 0.0.0.0/0
  latest handshake: 1 minute, 3 seconds ago
  transfer: 2.52 KiB received, 2.70 KiB sent
  persistent keepalive: every 30 seconds
```

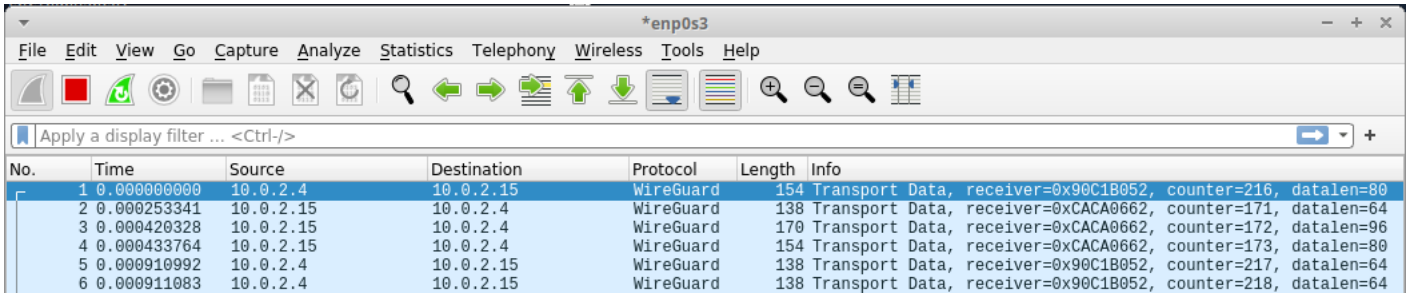
As we can see, the chat server can function without failure when we turn on wg0. The connection has now been protected by VPN tunneling

```
/pm binh Hello, how are you?
>>> binh has seen your message
```

```
<01/04/22 15:28:15> hoa: Hello, how are you?
```

➢ **VPN traffic capture using Wireshark**

Now we open the server VM and run **sudo wireshark** to capture the packets again. This time, the packets have been successfully protected by the WireGuard Protocol

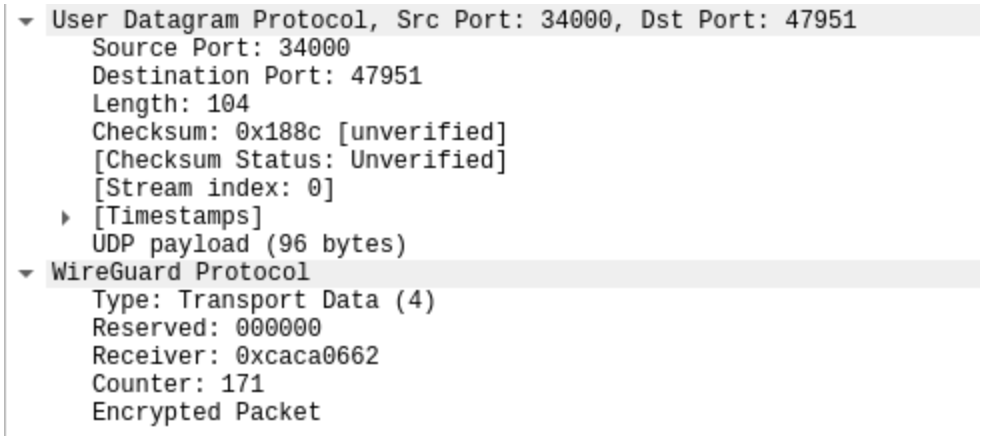



```
▸ Frame 2: 138 bytes on wire (1104 bits), 138 bytes captured (1104 bits) on interface enp0s3, id 0
0000  08 00 27 88 c0 c0 08 00  27 a2 f1 5a 08 00 45 00   ··'····· '··Z··E·
0010  00 7c da 2c 00 00 40 11  88 32 0a 00 02 0f 0a 00   ·|·,··@· ·2·····
0020  02 04 84 d0 bb 4f 00 68  18 8c 04 00 00 00 62 06   ·····O·h ······b·
0030  ca ca ab 00 00 00 00 00  00 00 31 fb c2 bc 29 20   ········ ··1···)
0040  43 34 2b c1 84 21 4c 7c  e4 91 f8 ed 9c d8 f1 59   C4+··!L| ·······Y
0050  81 0f a8 da 01 28 47 dc  15 1f 0b d2 64 99 07 f3   ·····(G· ····d···
0060  cc 71 78 9f ef ac 18 42  5e f3 1e cc c3 cf fd 44   ·qx····B ^·····D
0070  e5 64 63 4c 58 b9 c5 40  5c fa 79 18 fa bb 65 3c   ·dcLX··@ \·y···e<
0080  ea 94 13 9b 71 a4 cb dc  1f 90                     ····q··· ··
```

```
▸ Frame 4: 154 bytes on wire (1232 bits), 154 bytes captured (1232 bits) on interface enp0s3, id 0
0000  08 00 27 88 c0 c0 08 00  27 a2 f1 5a 08 00 45 00   ··'····· '··Z··E·
0010  00 8c da 2e 00 00 40 11  88 20 0a 00 02 0f 0a 00   ····..·@· · ·····
0020  02 04 84 d0 bb 4f 00 78  18 9c 04 00 00 00 62 06   ·····O·x ······b·
0030  ca ca ad 00 00 00 00 00  00 00 ad 2f 24 23 24 46   ········ ···/$#$F
0040  ed 16 01 64 3b c9 27 1e  7c bc 5c 4d 8b 24 04 9e   ···d;·'· |·\M·$··
0050  36 3e 5b 3c bb af a8 22  99 21 da e9 4a f9 b0 91   6>[<···" ·!··J···
0060  f7 20 81 c9 fb fd 1d 24  3e 85 53 89 65 a6 63 e9   ····· ···$ >·S·e·c·
0070  c9 10 a2 59 33 f3 24 a3  55 21 be 36 25 f4 03 36   ···Y3·$· U!·6%··6
0080  c7 eb 91 88 d4 a6 84 1f  6b 97 13 5a 08 ad 65 72   ········ k··Z··er
0090  b6 5c c1 18 23 b7 03 65  46 b0                     ·\··#··e F·
```

```
▸ Frame 5: 138 bytes on wire (1104 bits), 138 bytes captured (1104 bits) on interface enp0s3, id 0
0000  08 00 27 a2 f1 5a 08 00  27 88 c0 c0 08 00 45 00   ··'··Z·· '·····E·
0010  00 7c e5 dc 00 00 40 11  7c 82 0a 00 02 04 0a 00   ·|····@· |·······
0020  02 0f bb 4f 84 d0 00 68  58 7a 04 00 00 00 52 b0   ···O···h Xz····R·
0030  c1 90 d9 00 00 00 00 00  00 00 50 52 f5 e0 71 86   ········ ··PR··q·
0040  bc 7e ec e8 8a 0b 9f b0  6c 28 c1 6c cf 30 fc 39   ·~····· l(·l·0·9
0050  c6 83 5f f1 58 a4 a5 53  22 39 e9 9a b2 fd 28 cf   ··_·X··S "9····(·
0060  0a 5c 82 9b da 41 31 0d  ff 6e e1 22 80 8d 3c 51   ·\···A1· ·n·"··<Q
0070  bd 6c 11 08 ec 55 3f 12  c4 22 7a 09 b6 04 a6 09   ·l···U?· ·"z·····
0080  f8 2e 09 1e 72 09 5c bc  33 62                     ····r·\· 3b
```

Observation: the data packets have been encrypted into indecipherable text, which cannot be read by unintended third parties. The WireGuard Protocol serves to encrypt and authenticate the data packets at both endpoints



Source port 34000 is from the server VM and destination port 47951 is from the client VM, suggesting that this packet is indeed sent via the chat application. Under WireGuard protocol, Encrypted Packet means that the data has been protected over the network by encryption.

WireGuard Protocol explanation [1]:

- The WireGuard protocol passes traffic over UDP, which uses no handshake protocols.
- For encryption and authentication, WireGuard uses asymmetric cryptography that uses a public and private key pair. How the encryption and authentication works have been illustrated in the previous report.
- Besides creating an encrypted tunnel between two devices, WireGuard also works when the client device's IP address changes. For example, user can switch from mobile data to Wi-Fi without waiting thirty seconds for the VPN to reconnect.
- WireGuard is designed to have high speed performance and low attack surface. In addition, WireGuard is secure because its code is simple and therefore it is less prone to errors and misconfigurations.

[1] Source: https://cybernews.com/what-is-vpn/wireguard-protocol/

Section 4: Annexed Code

The code used for the chat application is contained in the attached zip file in the submission box. It is not attached in this report since the application code is significantly long