



Princess Sumaya جامعة
University الأميرة سميرة
for Technology للتكنولوجيا

Bypassing firewalls using VPNs

Name: Yara Al-Mhameed 20220822

Bahauldeen Qabani 20220106

Supervisor: Nour Kattawi

Contents

Introduction:	3
Task 1: VM Setup.....	3
Task 2: Firewall Setup.....	4
Identifying Target Website and checking connectivity	4
Firewall Configuration	4
Verification	5
Task 3: Bypassing Firewall using VPN	5
Setting up a VPN server.....	5
Connecting VPN client.....	6
Routing and NAT configurations.....	6
Telnet connection.....	7
NAT setup	8
Verify connectivity	8
Conclusion.....	9

Table of figures

Figure 1: IP address of each VM.....	3
Figure 2: Identifying IP of Facebook and connectivity	4
Figure 3: Firewall Configurations.....	4
Figure 4: Verifying the firewall rule.....	5
Figure 5: VPN server setup.....	5
Figure 6: Virtual tunnel	5
Figure 7: Client connectivity with VPN server.....	6
Figure 8: Confirming Connectivity.....	6
Figure 9: NAT configuration.....	6
Figure 10: IP forwarding.....	7
Figure 11: pinging Server	7
Figure 12: Telnet.....	7
Figure 13: NAT setup	8
Figure 14: Successful ping to blocked IP	8
Figure 15: Wireshark	8

Introduction:

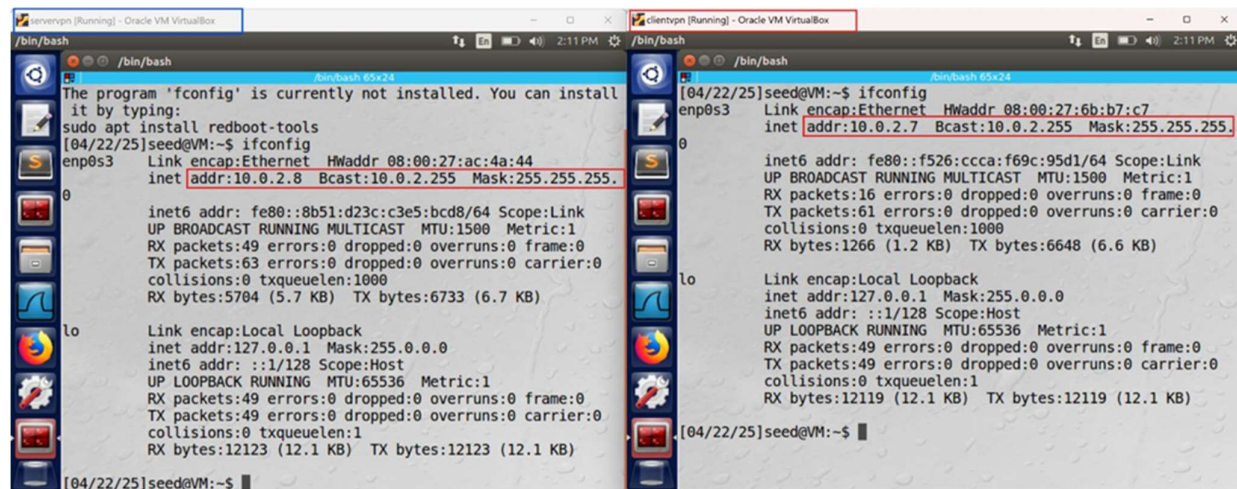
In This assignment we will show how firewalls work in blocking the traffic to specific addresses and how VPN servers bypass the firewall using the method of rerouting traffic through a tunnel. We used a simple example with 2 VMs one acting as a client and the other as a server.

Task 1: VM Setup

First we setup 2 different virtual machines one acting as a Client (**clientvpn**) and the other as a Server (**servervpn**), we then connected both VMs to a NAT network and assigned them IP Addresses.

Server = 10.0.2.8

Client = 10.0.2.7



The image shows two terminal windows from an Oracle VM VirtualBox. The left window is titled 'servervpn [Running] - Oracle VM VirtualBox' and shows the output of the 'ifconfig' command for the 'seed@VM' user. It displays the configuration for the 'enp0s3' interface, including the IP address 10.0.2.8, broadcast address 10.0.2.255, and mask 255.255.255. The right window is titled 'clientvpn [Running] - Oracle VM VirtualBox' and shows the output of the 'ifconfig' command for the 'seed@VM' user. It displays the configuration for the 'enp0s3' interface, including the IP address 10.0.2.7, broadcast address 10.0.2.255, and mask 255.255.255. Both windows also show the configuration for the 'lo' (loopback) interface.

```
/bin/bash
The program 'fconfig' is currently not installed. You can install
it by typing:
sudo apt install redboot-tools
[04/22/25]seed@VM:~$ ifconfig
enp0s3  Link encap:Ethernet  HWaddr 08:00:27:ac:4a:44
        inet addr:10.0.2.8  Bcast:10.0.2.255  Mask:255.255.255
        inet6 addr: fe80::8b51:d23c:c3e5:bcd8/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:49 errors:0 dropped:0 overruns:0 frame:0
        TX packets:63 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:5704 (5.7 KB)  TX bytes:6733 (6.7 KB)

lo      Link encap:Local Loopback
        inet addr:127.0.0.1  Mask:255.0.0.0
        inet6 addr: ::1/128 Scope:Host
        UP LOOPBACK RUNNING  MTU:65536  Metric:1
        RX packets:49 errors:0 dropped:0 overruns:0 frame:0
        TX packets:49 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1
        RX bytes:12123 (12.1 KB)  TX bytes:12123 (12.1 KB)

[04/22/25]seed@VM:~$
```

```
/bin/bash
[04/22/25]seed@VM:~$ ifconfig
enp0s3  Link encap:Ethernet  HWaddr 08:00:27:6b:b7:c7
        inet addr:10.0.2.7  Bcast:10.0.2.255  Mask:255.255.255
        inet6 addr: fe80::f526:ccca:f69c:95d1/64 Scope:Link
        UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
        RX packets:16 errors:0 dropped:0 overruns:0 frame:0
        TX packets:61 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1000
        RX bytes:1266 (1.2 KB)  TX bytes:6648 (6.6 KB)

lo      Link encap:Local Loopback
        inet addr:127.0.0.1  Mask:255.0.0.0
        inet6 addr: ::1/128 Scope:Host
        UP LOOPBACK RUNNING  MTU:65536  Metric:1
        RX packets:49 errors:0 dropped:0 overruns:0 frame:0
        TX packets:49 errors:0 dropped:0 overruns:0 carrier:0
        collisions:0 txqueuelen:1
        RX bytes:12119 (12.1 KB)  TX bytes:12119 (12.1 KB)

[04/22/25]seed@VM:~$
```

Figure 1: IP address of each VM

Task 2: Firewall Setup

Identifying Target Website and checking connectivity

In this assignment the target website is www.facebook.com, we first tested if we had connectivity and additionally got its IP address which we will use later on.

```
[04/22/25]seed@VM:~$ ping wwwfacebook.com
PING wwwfacebook.com (102.132.103.8) 56(84) bytes of data.
64 bytes from edge-star-shv-02-hbe1.facebook.com (102.132.103.8):
  icmp_seq=1 ttl=53 time=26.0 ms
64 bytes from edge-star-shv-02-hbe1.facebook.com (102.132.103.8):
  icmp_seq=2 ttl=53 time=30.9 ms
64 bytes from edge-star-shv-02-hbe1.facebook.com (102.132.103.8):
  icmp_seq=3 ttl=53 time=21.6 ms
64 bytes from edge-star-shv-02-hbe1.facebook.com (102.132.103.8):
  icmp_seq=4 ttl=53 time=22.1 ms
64 bytes from edge-star-shv-02-hbe1.facebook.com (102.132.103.8):
  icmp_seq=5 ttl=53 time=18.9 ms
64 bytes from edge-star-shv-02-hbe1.facebook.com (102.132.103.8):
  icmp_seq=6 ttl=53 time=22.6 ms
64 bytes from edge-star-shv-02-hbe1.facebook.com (102.132.103.8):
  icmp_seq=7 ttl=53 time=21.9 ms
```

Figure 2: Identifying IP of Facebook and connectivity

Here we confirmed the connectivity and got the IP address which corresponds to [102.132.103.8](https://www.facebook.com)

Firewall Configuration

Next we enabled the firewall using the command `sudo ufw enable` then used the command `sudo ufw deny out on enp0s3 to 102.132.103.8` which blocks the traffic going from the client to this IP address ([Facebook](https://www.facebook.com)), and lastly used command `sudo ufw status` to ensure it is blocking the traffic.

```
8 packets transmitted, 8 received, 0% packet loss, time 7014ms
rtt min/avg/max/mdev = 18.985/22.959/30.990/3.663 ms
[04/22/25]seed@VM:~$ sudo ufw enable
Firewall is active and enabled on system startup
[04/22/25]seed@VM:~$ sudo ufw deny out on enp0s3 to 102.132.103.8
Rule added
[04/22/25]seed@VM:~$ sudo ufw status
Status: active

To Action From
--
102.132.103.8 DENY OUT Anywhere on enp0s3
[04/22/25]seed@VM:~$
```

Figure 3: Firewall Configurations

Verification

We then pinged the IP address again to check if the Firewall is actually blocking the traffic or not

```
[04/22/25]seed@VM:~$ ping 102.132.103.8
PING 102.132.103.8 (102.132.103.8) 56(84) bytes of data.
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
ping: sendmsg: Operation not permitted
^C
--- 102.132.103.8 ping statistics ---
4 packets transmitted, 0 received, 100% packet loss, time 3078ms
```

Figure 4: Verifying the firewall rule

Which turned out to be Successful

Task 3: Bypassing Firewall using VPN

Setting up a VPN server

First we started a VPN server using a C code which we got from [SeedLabs](#) and then we compiled the code and executed it

```
[04/22/25]seed@VM:~/.../vpn$ gcc vpnserver.c -o vpnserver
[04/22/25]seed@VM:~/.../vpn$ sudo ./vpnserver
```

Figure 5: VPN server setup

We then configured a virtual tunnel

```
/bin/bash 65x24
[04/22/25]seed@VM:~$ sudo ifconfig tun0 192.168.53.1/24 up
[04/22/25]seed@VM:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOW
WN group default qlen 1
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo
_fast state UP group default qlen 1000
    link/ether 08:00:27:ac:4a:44 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.8/24 brd 10.0.2.255 scope global enp0s3
        valid_lft forever preferred_lft forever
    inet6 fe80::8b51:d23c:c3e5:bcd8/64 scope link
        valid_lft forever preferred_lft forever
3: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc
pfifo fast state UNKNOWN group default qlen 500
    link/none
    inet 192.168.53.1/24 scope global tun0
        valid_lft forever preferred_lft forever
    inet6 fe80::9005:a77d:943b:87e6/64 scope link flags 800
        valid_lft forever preferred_lft forever
[04/22/25]seed@VM:~$
```

Figure 6: Virtual tunnel

For the **server** we gave it the tunnel IP address of **192.168.53.1/24** and for the **client** we gave the tunnel IP address of **192.168.53.5/24**

Connecting VPN client

We used a C code which we got from **SeedLabs** and then we compiled the code and executed it, this code helped us in connecting with the VPN server

```
[04/22/25]seed@VM:~/.../vpn$ gcc vpnclient.c -o vpnclient
[04/22/25]seed@VM:~/.../vpn$ sudo ./vpnclient 10.0.2.8
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
```

Figure 7: Client connectivity with VPN server

We then returned to the VPN server to check if the connection was successful

```
[04/22/25]seed@VM:~/.../vpn$ sudo ./vpnsrv
Connected with the client: Hello
Got a packet from TUN
Got a packet from TUN
Got a packet from TUN
Got a packet from the tunnel
Got a packet from the tunnel
Got a packet from the tunnel
```

Figure 8: Confirming Connectivity

Which was confirmed to be a success with the message **Connected with the client: Hello**

Routing and NAT configurations

First we configured a route on the client which allowed traffic to move through the VPN

```
[04/22/25]seed@VM:~$ sudo route add -net 192.168.53.0/24 dev tun0

[04/22/25]seed@VM:~$ route
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref
Use Iface
default 10.0.2.1 0.0.0.0 UG 100 0
0 enp0s3
10.0.2.0 * 255.255.255.0 U 100 0
0 enp0s3
link-local * 255.255.0.0 U 1000 0
0 enp0s3
192.168.53.0 * 255.255.255.0 U 0 0
0 tun0
192.168.53.0 ← * 255.255.255.0 U 0 0
0 tun0
[04/22/25]seed@VM:~$
```

Figure 9: NAT configuration

Then we did the same to the server and added a command that enabled **IP forwarding**

```
[04/22/25]seed@VM:~$ sudo route add -net 192.168.53.0/24 dev tun0
[04/22/25]seed@VM:~$ route
Kernel IP routing table
Destination        Gateway            Genmask           Flags Metric Ref
Use Iface
default            10.0.2.1          0.0.0.0           UG    100    0
0 enp0s3
10.0.2.0           *                 255.255.255.0     U     100    0
0 enp0s3
link-local         *                 255.255.0.0       U     1000   0
0 enp0s3
192.168.53.0       *                 255.255.255.0     U     0      0
0 tun0
192.168.53.0 ← *   255.255.255.0     U     0      0
0 tun0
[04/22/25]seed@VM:~$ sudo sysctl net.ipv4.ip_forward=1
net.ipv4.ip_forward = 1
[04/22/25]seed@VM:~$
```

Figure 10: IP forwarding

Then we checked connectivity from client to server

```
[04/22/25]seed@VM:~$ ping 192.168.53.1
PING 192.168.53.1 (192.168.53.1) 56(84) bytes of data:
64 bytes from 192.168.53.1: icmp_seq=1 ttl=64 time=0.717 ms
64 bytes from 192.168.53.1: icmp_seq=2 ttl=64 time=2.53 ms
64 bytes from 192.168.53.1: icmp_seq=3 ttl=64 time=2.22 ms
64 bytes from 192.168.53.1: icmp_seq=4 ttl=64 time=3.44 ms
64 bytes from 192.168.53.1: icmp_seq=5 ttl=64 time=0.730 ms
64 bytes from 192.168.53.1: icmp_seq=6 ttl=64 time=1.08 ms
^C
--- 192.168.53.1 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5052ms
rtt min/avg/max/mdev = 0.717/1.787/3.441/1.021 ms
[04/22/25]seed@VM:~$
```

Figure 11: pinging Server

Which was a success

Telnet connection

Next we established a telnet connection which helped with the tunnel connection using the command **telnet**

```
[04/22/25]seed@VM:~$ telnet 192.168.53.1
Trying 192.168.53.1...
Connected to 192.168.53.1.
Escape character is '^]'.
Ubuntu 16.04.2 LTS
VM login: seed
Password:
Welcome to Ubuntu 16.04.2 LTS (GNU/Linux 4.8.0-36-generic i686)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

1 package can be updated.
0 updates are security updates.
```

Figure 12: Telnet

NAT setup

To accomplish NAT connection on the VPN server we do the following commands

```
[04/22/25]seed@VM:~$ sudo iptables -F  
[04/22/25]seed@VM:~$ sudo iptables -t nat -F  
[04/22/25]seed@VM:~$ sudo iptables -t nat -A POSTROUTING -j MASQUERADE -o enp0s3  
[04/22/25]seed@VM:~$
```

Figure 13: NAT setup

Sudo iptables -F: Cleared all firewall ruled

sudo iptables -t nat -F: Clear NAT table rules

sudo iptables -t nat -A POSTROUTING -j MASQUERADE -o enp0s3: Masks client traffics as if the traffic is forwarded from the server

Verify connectivity

Last we returned to the client to check if the connectivity to the blocked IP address is successful

```
[04/22/25]seed@VM:~$ ping 102.132.103.8  
PING 102.132.103.8 (102.132.103.8) 56(84) bytes of data.  
64 bytes from 102.132.103.8: icmp_seq=1 ttl=53 time=21.2 ms  
64 bytes from 102.132.103.8: icmp_seq=2 ttl=53 time=21.0 ms  
64 bytes from 102.132.103.8: icmp_seq=3 ttl=53 time=23.2 ms  
64 bytes from 102.132.103.8: icmp_seq=4 ttl=53 time=31.8 ms  
64 bytes from 102.132.103.8: icmp_seq=5 ttl=53 time=24.7 ms  
64 bytes from 102.132.103.8: icmp_seq=6 ttl=53 time=19.6 ms  
^C  
--- 102.132.103.8 ping statistics ---  
6 packets transmitted, 6 received, 0% packet loss, time 5031ms  
rtt min/avg/max/mdev = 19.613/23.635/31.846/4.029 ms
```

Figure 14: Successful ping to blocked IP

Which was successful

To fully check that the traffic is actually bring forwarded we used **Wireshark** to confirm

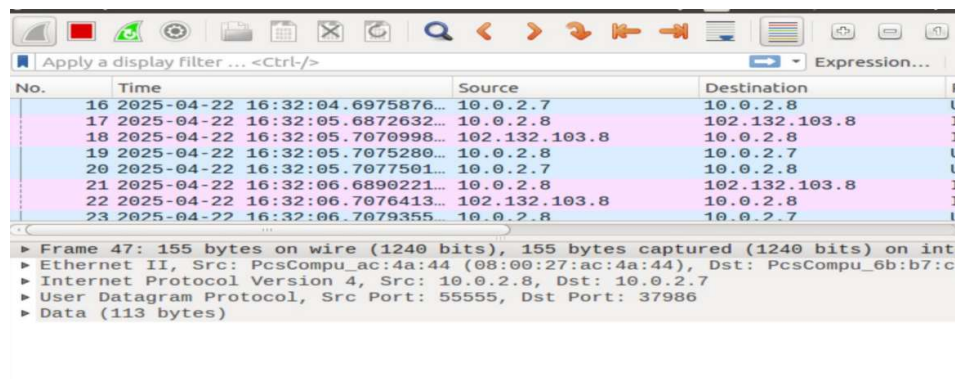


Figure 15: Wireshark

Conclusion

Using a virtual private network (VPN) to get around firewalls entails sending your internet traffic to a distant server via an encrypted tunnel, which conceals your IP address and your online behaviour. This enables users to bypass restrictions and access content that is blocked. Although efficient, it can be against local laws or terms of service, and not all VPNs can get past sophisticated firewalls like those that use deep packet inspection.