

Cybersecurity - VPN project

Larisa [REDACTED], [REDACTED] Voinea

2022-03-19

Contents

Team info	2
1 VPN Project	3
1.1 Point-to-point VPN tunnel	4

Team info

- Larisa [REDACTED]
- Gheorghe [REDACTED] (aka Doru)
- Group: 410

Chapter 1

VPN Project

The objective of this project is to familiarize with the VPN notions by implementing simple VPN tunnels between two computers and checking the connectivity between them.

We picked-up the **OpenVPN** (community edition) and used two virtual machines running **Ubuntu 20.04 LTS** hosted in cloud (Microsoft Azure).

The reasons of using cloud infrastructure vs using our home computers/networks are:

- easy to setup the VMs;
- easy to setup the virtual private network and to create subnets;
- no security risks compared with the home networks where we have to open ports and to protect the internal assets

OpenVPN install

Run the following commands on both computers

```
sudo apt-get update
sudo apt-get install net-tools
sudo apt-get install traceroute
sudo apt-get install openvpn
```

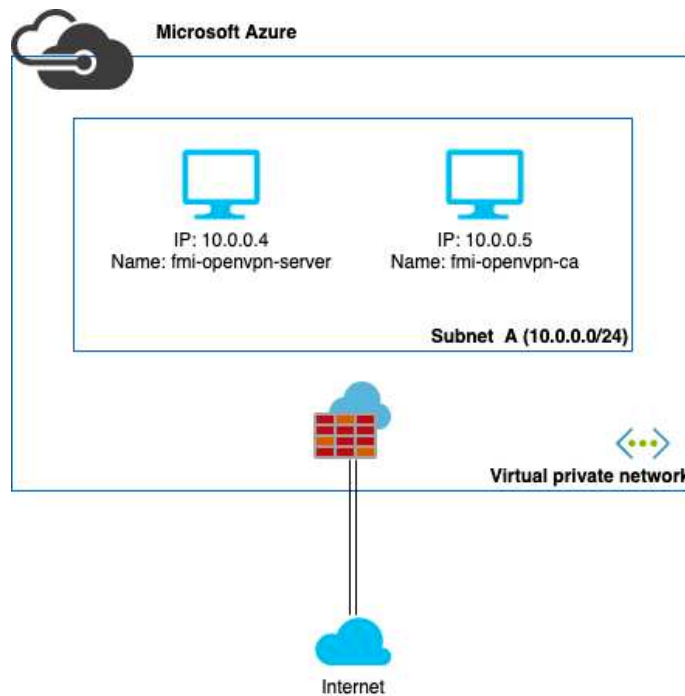


Figure 1.1: Network diagram

1.1 Point-to-point VPN tunnel

For this configuration we are creating a direct VPN tunnel between two machines over an unsecured channel using UDP transport protocol

- Network configuration of first machine before building the VPN tunnel

```
azureuser@fmi-openvpn-server:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.4 netmask 255.255.255.0 broadcast 10.0.0.255
    inet6 fe80::222:48ff:fea6:9151 prefixlen 64 scopeid 0x20<link>
    ether 00:22:48:a6:91:51 txqueuelen 1000 (Ethernet)
    RX packets 160183 bytes 92748614 (92.7 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 120546 bytes 42704670 (42.7 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 642 bytes 73522 (73.5 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 642 bytes 73522 (73.5 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

- Network configuration of second machine before building the VPN tunnel

```

azureuser@fmi-openvpn-ca:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
    inet 10.0.0.5  netmask 255.255.255.0  broadcast 10.0.0.255
    inet6 fe80::20d:3aff:fe73:9711  prefixlen 64  scopeid 0x20<link>
    ether 00:0d:3a:73:97:11  txqueuelen 1000  (Ethernet)
    RX packets 78966  bytes 38864913 (38.8 MB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 73582  bytes 18955116 (18.9 MB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
    inet 127.0.0.1  netmask 255.0.0.0
    inet6 ::1  prefixlen 128  scopeid 0x10<host>
    loop txqueuelen 1000  (Local Loopback)
    RX packets 717  bytes 77134 (77.1 KB)
    RX errors 0  dropped 0  overruns 0  frame 0
    TX packets 717  bytes 77134 (77.1 KB)
    TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

```

- Routes on first machine

```

azureuser@fmi-openvpn-server:~$ route -n
Kernel IP routing table

```

Destination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
0.0.0.0	10.0.0.1	0.0.0.0	UG	100	0	0	eth0
10.0.0.0	0.0.0.0	255.255.255.0	U	0	0	0	eth0
168.63.129.16	10.0.0.1	255.255.255.255	UGH	100	0	0	eth0
169.254.169.254	10.0.0.1	255.255.255.255	UGH	100	0	0	eth0

1.1.1 Building an unsecured VPN tunnel

We are using **TUN**-style network that allows peering IP traffic.

```
fmi-openvpn-server:  
$ sudo openvpn --local fmi-openvpn-server --dev tun0 --ifconfig 10.200.0.1 10.200.0.2  
  
fmi-openvpn-ca:  
$ sudo openvpn --remote fmi-openvpn-server --dev tun0 --ifconfig 10.200.0.2 10.200.0.1
```

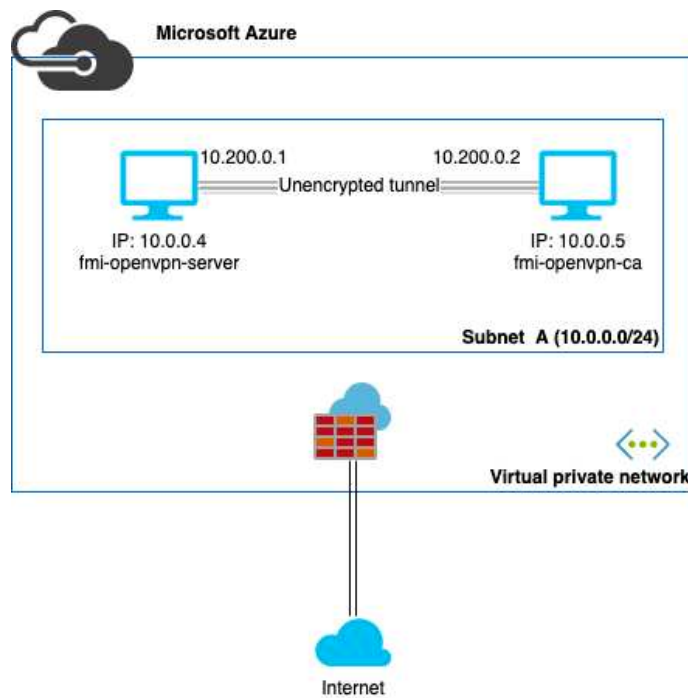


Figure 1.2: Network diagram with VPN tunnel up

Network configuration of first machine after raising the VPN tunnel

```
azureuser@fmi-openvpn-server:~$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.4 netmask 255.255.255.0 broadcast 10.0.0.255
    inet6 fe80::222:48ff:fea6:9151 prefixlen 64 scopeid 0x20<link>
    ether 00:22:48:a6:91:51 txqueuelen 1000 (Ethernet)
    RX packets 172079 bytes 95603700 (95.6 MB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 135212 bytes 46569237 (46.5 MB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (Local Loopback)
    RX packets 662 bytes 75898 (75.8 KB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 662 bytes 75898 (75.8 KB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

tun0: flags=4305<UP,POINTOPOINT,RUNNING,NOARP,MULTICAST> mtu 1500
    inet 10.200.0.1 netmask 255.255.255.255 destination 10.200.0.2
    inet6 fe80::dd94:502:1e08:ded prefixlen 64 scopeid 0x20<link>
    unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00 txqueuelen 100 (UNSPEC)
    RX packets 2 bytes 96 (96.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 2 bytes 96 (96.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

We can see the new interface **tun0** with the IP: 10.200.0.1

```
azureuser@fmi-openvpn-server:~$ route -n
Kernel IP routing table
Destination      Gateway          Genmask          Flags Metric Ref    Use Iface
0.0.0.0          10.0.0.1         0.0.0.0          UG      100    0      0 eth0
10.0.0.0          0.0.0.0          255.255.255.0    U        0      0      0 eth0
10.200.0.0        10.0.0.4         255.255.255.0    UG       0      0      0 eth0
10.200.0.2        0.0.0.0          255.255.255.255 UH       0      0      0 tun0
168.63.129.16    10.0.0.1         255.255.255.255 UGH     100    0      0 eth0
169.254.169.254  10.0.0.1         255.255.255.255 UGH     100    0      0 eth0
```

When we check the routes on first machine we can see there is a new route to **10.200.0.0**

If we use tcpdump to sniff the ICMP traffic on second machine interface **eth0** initiated from first computer, will see traffic over UDP port 1194 as expected

```
azureuser@fmi-openvpn-ca:~$ sudo tcpdump -nvvv port not ssh and \
    host not 168.63.129.16 and host not 169.254.169.254
tcpdump: listening on eth0, link-type EN10MB (Ethernet), capture size 262144 bytes
14:27:12.463235 IP (tos 0x0, ttl 64, id 49274, offset 0, flags [DF], proto UDP (17), length 112)
    10.0.0.4.1194 > 10.0.0.5.1194: [udp sum ok] UDP, length 84
14:27:12.463425 IP (tos 0x0, ttl 64, id 33539, offset 0, flags [DF], proto UDP (17), length 112)
    10.0.0.5.1194 > 10.0.0.4.1194: [bad udp cksum 0x1476 -> 0xe1d9!] UDP, length 84
14:27:13.465269 IP (tos 0x0, ttl 64, id 49332, offset 0, flags [DF], proto UDP (17), length 112)
    10.0.0.4.1194 > 10.0.0.5.1194: [udp sum ok] UDP, length 84
```



```
14:27:13.465390 IP (tos 0x0, ttl 64, id 33667, offset 0, flags [DF], proto UDP (17), length 112)
  10.0.0.5.1194 > 10.0.0.4.1194: [bad udp cksum 0x1476 -> 0xe1d9!] UDP, length 84
```

If will tie the tcpdump to **tun0** interface will see something more familiar

```
azureuser@fmi-openvpn-ca:~$ sudo tcpdump -nvvv -i tun0
tcpdump: listening on tun0, link-type RAW (Raw IP), capture size 262144 bytes
14:28:02.990859 IP (tos 0x0, ttl 64, id 26037, offset 0, flags [DF], proto ICMP (1), length 84)
  10.200.0.1 > 10.200.0.2: ICMP echo request, id 42, seq 1, length 64
14:28:02.990896 IP (tos 0x0, ttl 64, id 33900, offset 0, flags [none], proto ICMP (1), length 84)
  10.200.0.2 > 10.200.0.1: ICMP echo reply, id 42, seq 1, length 64
14:28:04.010783 IP (tos 0x0, ttl 64, id 26184, offset 0, flags [DF], proto ICMP (1), length 84)
  10.200.0.1 > 10.200.0.2: ICMP echo request, id 42, seq 2, length 64
14:28:04.010815 IP (tos 0x0, ttl 64, id 33936, offset 0, flags [none], proto ICMP (1), length 84)
  10.200.0.2 > 10.200.0.1: ICMP echo reply, id 42, seq 2, length 64
```

To wrap-up the above steps, we saved the configurations within config files:

```
# server.conf
# openvpn plain tunnel
dev tun
ifconfig 10.200.0.1 10.200.0.2
local fmi-openvpn-server
```

```
# client.conf
# openvpn plain tunnel
dev tun
ifconfig 10.200.0.2 10.200.0.1
remote fmi-openvpn-server
```

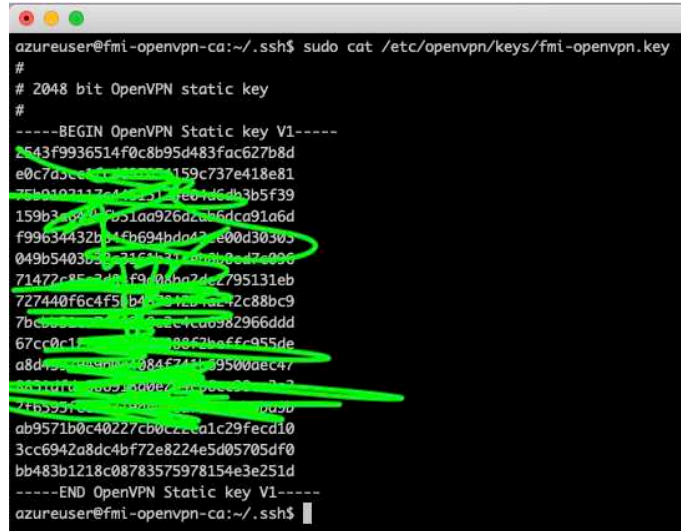
1.1.2 Secured VPN tunnel with a shared static key

In this section we will create a secret key on **fmi-openvpn-ca** and will place it on the other machines as well.

This is not the best solution, as it will not assure the **forward secrecy**, so if a rouge person will intercept the encrypted traffic and later one found the symmetric key, they will be able to decrypt the messages.

- Generating the secret key

```
sudo openvpn --genkey --secret fmi-openvpn.key
```



```

azureuser@fmi-openvpn-ca:~/.ssh$ sudo cat /etc/openvpn/keys/fmi-openvpn.key
#
# 2048 bit OpenVPN static key
#
-----BEGIN OpenVPN Static key V1-----
2543f9936514f0c8b95d483fac627b8d
e0c7a3ee5ac188881159c737e418e81
75b0102117c442311e04d64b3b5f39
159b31690f051aa926d2ab6dca91a6d
f99634432b14fb694bda42e00d30393
049b5403030c316511f00d08ed7c090
71477c85340f9f08ba24c2795131eb
727440f6c4f51b41204204ac72c88bc9
7bc000000000000000000000000000
67cc0c1f0000000000000000000000
a8d400000000000000000000000000
000000000000000000000000000000
4fb55f1c0000000000000000000000
ab9571b0c40227cb0c22ca1c29fecdl0
3cc6942a8dc4bf72e8224e5d05705df0
bb483b1218c08783575978154e3e251d
-----END OpenVPN Static key V1-----
azureuser@fmi-openvpn-ca:~/.ssh$

```

Figure 1.3: Static key

Will copy the key on both server and client, and will change the configuration files to refer to the key

```

# server.conf
# openvpn static key
dev tun
ifconfig 10.200.0.1 10.200.0.2
local fmi-openvpn-server
secret /etc/openvpn/keys/fmi-openvpn.key
cipher AES-256-CBC

```