Unit Name: Network Security and Resilience

Unit Code: TNE30009

Title: NSR/AS Lab 3 – VPNs

Name: S M RAGIB REZWAN

ID: 103172423

Contents

Abstract:

This is a lab report based upon TNE30009's week 6's lab3 task **[1]** on OpenVPN. It begins with an "Introduction to VPNs" section which contains brief background on VPN tunnels, its tunneling protocols and its relevance in implementing Organizational Security Policy. After this, it leads to the "OpenVPN behavior" section which explains what OpenVPN is, how to set up the VPN tunnel (along with screenshot of the steps' outcomes for visualization) and how traffic is passed between the interfaces. Lastly, it has the "Conclusion" section where it wraps everything up as a brief summary.

Introduction to VPNs:

VPN is basically a mechanism that is used to create a secure network over an existing insecure communication medium (like Internet) **[2]**. It does this by creating a tunnel that connects two end points (mimicking a point to point connection) and encapsulating the packets passing through it using its own tunneling packet header.

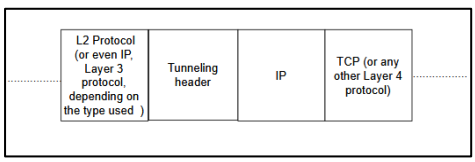

Figure 1: Generic Headers present in packet travelling through VPN tunnel

Here the information present in the Tunnel header varies depending on the type of protocol that has been used. This has been detailed in table1:

Tunneling protocol	Ones who created it	Types of tunneling protocol that are used by it	Types of encryption
IPSec	IETF **[3]**	Authentication Header (AH), Encapsulating security payload (ESP), Internet key Exchange (IKE)	AES, Blowfish, Triple DES, ChaCha, DES-CBC
PPTP	Microsoft, Ascend Communication, 3Com/Primary Access, ECI Telematics, and US Robotics **[4]**	PPP datagrams (to encapsulate IPX, NetBEUI, TCP/IP packets)	Remote access service(RAS) shared-secret, PPP encryption and RSA's RC4 (Rivest Cipher 4)
L2TP	Cisco **[5]**	L2TP protocols (but can be used in conjunction to IPSec to use its protocols too) **[6]**	It doesn't provide any encryption on its own, but when used with IPSec, it can provide IpSec's encryption types **[7]**
SSTP	Microsoft **[8]**	SSL/TLS **[8] [9]**	AES (Advanced Encryption Standard) **[8]**

Table1: VPN tunneling protocols characteristic and capabilities

But basically, no matter what type of tunneling protocol is used, VPN still provides advantages in terms of lower implementation and management costs, better connectivity, improved security, better scalability, higher efficiency, better privacy, etc. **[5]**.

Furthermore, it is also utilized in setting up the security policies in an organization. These are basically clear, comprehensive, and well-defined plans, rules, and practices that regulate access to an organization's system and the information included in it **[10]**. Hence it is quite easy to see how crucial VPN is in that regard as it ensures that only those with proper authentication and authorization can access the resources though the tunnel.

Note: this has been further supported by the creation of template VPN policy by the NCS (National Cybersecurity Society **[11]**) (which has been used in various companies worldwide (AWS, Apple, Dell Technologies, etc.**[12]**) in order to prevent unauthorized access to their resources, ensure data using the tunnel is properly encrypted, setup time limits for access, etc**[11]**).

OpenVPN behaviour:

But how does OpenVPN relate to all of these?

Well, OpenVPN is basically an open–source VPN protocol that has become quite popular in recent years. It is used to not only create the point to point (or site to site) VPN connections, but also ensure proper encryption (hence security) for the packets in the tunnel with the help of OpenSSL library and SSL/TSL encryption header. Furthermore, it can also traverse both NAT and firewall with ease **[13]**.

Overall, it provides the benefits of having better security, stronger encryption and reliable connection, with the trade off in terms of speed, setup, and its 3rd party application requirements **[14]**. But even so, on average, it is still better than the other commonly used VPN protocols (including PPTP, L2TP/IPSec, SSTP,etc) in terms of encryption, security, speed, stability and compatibility (**see figure 2**):

	OpenVPN	PPTP	L2TP/IPsec	SoftEther	WireGuard	SSTP	IKEv2/IPSec
Encryption	160-bit, 256-bit	128-bit	256-bit	256-bit	ChaCha20	256-bit	256-bit
Security	Very high	Weak	High security (might be weakened by NSA)	High	High	High	High
Speed	Fast	Speedy, due to low encryption	Medium, due to double encapsulation	Very fast	Fast	Fast	Very fast
Stability	Very stable	Very stable	Stable	Very stable	Not yet stable	Very stable	Very stable
Compatibility	Strong desktop support, but mobile could be improved. Requires third-party software.	Strong Windows desktop support.	Multiple device and platform support.	Multiple desktop and mobile OS support. No native operating system support.	Linux, being built for other platforms and operating systems.	Windows-platform, but works on other Linux distributions.	Limited platform support beyond Windows and Blackberry

Figure 2: Comparison between VPNS in terms of encryption, security, speed, stability and compatibility **[15]**

But, how does OpenVPN provide these benefits?

In order to answer this question, a simple encrypted tunnel has been set up between two virtual machines in the lab. This has mainly been done to understand how OpenVPN works and the details regarding what the steps taken (alongside the rationale behind them, interesting observations, and the ways to verify/ test them) have been noted in points below:

1) Setup of the host configuration: In this step, the IP addresses present on the Ethernet interface (ens33) of the two VMs (virtual machines) have been obtained using the command **"ifconfig"**. After that, ping test has been performed using the command **"ping x.x.x.x"** where x.x.x.x was basically the IP address of VMs. This has been done to ensure that the TCP/IP protocols have been properly set within the VM (for self ping) and also ensure that the VMs can communicate with one another.

```
nsr@nsr-VirtualBox:~/Desktop$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.19.128  netmask 255.255.255.0  broadcast 192.168.19.255
        inet6 fe80::7b4c:4bff:cbc5:2b4f  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:3c:d4:8e  txqueuelen 1000  (Ethernet)
        RX packets 22  bytes 3512 (3.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 76  bytes 8920 (8.9 KB)
        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 45  bytes 5202 (5.2 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 45  bytes 5202 (5.2 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 3: IP address on ens33 for VM1

```
nsr@nsr-VirtualBox:~/Desktop$ ifconfig
ens33: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.19.129  netmask 255.255.255.0  broadcast 192.168.19.255
        inet6 fe80::32cb:dd33:ef4f:640  prefixlen 64  scopeid 0x20<link>
        ether 00:0c:29:22:b8:1c  txqueuelen 1000  (Ethernet)
        RX packets 66  bytes 9599 (9.5 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 75  bytes 8283 (8.2 KB)
        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 41  bytes 4099 (4.0 KB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 41  bytes 4099 (4.0 KB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

Figure 4: IP address on ens33 for VM2

```
nsr@nsr-VirtualBox:~/Desktop$ ping 192.168.19.128
PING 192.168.19.128 (192.168.19.128) 56(84) bytes of data.
64 bytes from 192.168.19.128: icmp_seq=1 ttl=64 time=0.017 ms
64 bytes from 192.168.19.128: icmp_seq=2 ttl=64 time=0.025 ms
64 bytes from 192.168.19.128: icmp_seq=3 ttl=64 time=0.045 ms
64 bytes from 192.168.19.128: icmp_seq=4 ttl=64 time=0.056 ms
^C
--- 192.168.19.128 ping statistics ---
4 packets transmitted, 4 received, 0% packet loss, time 3068ms
rtt min/avg/max/mdev = 0.017/0.035/0.056/0.015 ms
nsr@nsr-VirtualBox:~/Desktop$ ping 192.168.19.129
PING 192.168.19.129 (192.168.19.129) 56(84) bytes of data.
64 bytes from 192.168.19.129: icmp_seq=1 ttl=64 time=0.510 ms
64 bytes from 192.168.19.129: icmp_seq=2 ttl=64 time=0.785 ms
^C
--- 192.168.19.129 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1015ms
rtt min/avg/max/mdev = 0.510/0.647/0.785/0.137 ms
```

Figure 5: Pinging on VM1

```
nsr@nsr-VirtualBox:~/Desktop$ ping 192.168.19.129
PING 192.168.19.129 (192.168.19.129) 56(84) bytes of data.
64 bytes from 192.168.19.129: icmp_seq=1 ttl=64 time=0.032 ms
64 bytes from 192.168.19.129: icmp_seq=2 ttl=64 time=0.027 ms
^C
--- 192.168.19.129 ping statistics ---
2 packets transmitted, 2 received, 0% packet loss, time 1024ms
rtt min/avg/max/mdev = 0.027/0.029/0.032/0.002 ms
nsr@nsr-VirtualBox:~/Desktop$ ping 192.168.19.128
PING 192.168.19.128 (192.168.19.128) 56(84) bytes of data.
64 bytes from 192.168.19.128: icmp_seq=1 ttl=64 time=0.271 ms
64 bytes from 192.168.19.128: icmp_seq=2 ttl=64 time=0.769 ms
64 bytes from 192.168.19.128: icmp_seq=3 ttl=64 time=0.712 ms
64 bytes from 192.168.19.128: icmp_seq=4 ttl=64 time=0.346 ms
64 bytes from 192.168.19.128: icmp_seq=5 ttl=64 time=0.298 ms
64 bytes from 192.168.19.128: icmp_seq=6 ttl=64 time=0.484 ms
^C
--- 192.168.19.128 ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5114ms
rtt min/avg/max/mdev = 0.271/0.480/0.769/0.196 ms
```

Figure 6: Pinging on VM2

2) Generate shared password: In this step, we create the shared password (of 256 byte or 2048 bit size) that would be used later on as a key to encrypt the traffic passing through the tunnel. This has been done by using the command “openvpn –genkey –secret mykey” which is basically asking OpenVPN to generate a secret key and save it in a file called mykey.

Once, this has been done, the process can be verified by using the command “cat mykey” which basically tells Linux to list out the contents in the file.

```
# 2048 bit OpenVPN static key
#
-----BEGIN OpenVPN Static key V1-----
f424db6250d44138b7dfc3ef6213e0bf
8852c9b2b08a21d9f54b20161834c79d
6761f44794c54f801f8ad8f066187d10
636d2ffd0a48cd34aff3a63f24c24754
7daf6bd001afd560955908662e3d1d11
7bcec6c8b510a698cbdc5ea2c19a8cbe
abc2a07089794b233a6b160b1b1fd17a
7820a9a94ce749c861ca561d167487e8
6e71ff3da02e42cef0e5d6d6d9819e33
33bbeadfdbc831e062f068c34ec35795
38131c5f20f0a195b5f883a225b99296
85d455a5fd93e59e0a3e8867ea419c6a
054e592ac8caa7625c10007115435927
fe38c02455d3ceeecc57171a4754e201
df68c09155a7e322f7e8e5260c634db5
deacde78effd6d8bcd44d60ec6088563
-----END OpenVPN Static key V1-----
```

Figure 7: The Openvpn key that have been generated and stored in the mykey file

3) Transfer shared key to other machine: In this step, we use SCP (Secure Copy) to copy it from our current VM (VM1) to the other VM (VM2) that it can communicate with. This has been done as without the key, it would not be possible to set up the tunnel from the other end and thus would make the tunnel nonfunctional. This has been setup using the command “scp mykey nsr@192.168.19.129: Desktop/mykey” to ensure that the system sends the mykey file to the other machine’s nsr user’s desktop and stores it as mykey file there.

Once this has been entered (alongside the password of “user”), it will take a short time before stating it to be 100% complete (**as seen in the figure below**)

```
nsr@192.168.19.129's password:
mykey                                         100%  636   548.2KB/s   00:00
```

Figure 8: SCP has been successful in sending the mykey file

This can be further verified by checking the files on the desktop on VM2 (using the command “ls - al” which lists all the files in a long list format) and “cat’-ing the mykey file (**as seen in the figure below**).

```
nsr@nsr-VirtualBox:~/Desktop$ ls -al
total 12
drwxr-xr-x  2 nsr nsr 4096 Apr  4 22:17 .
drwxr-xr-x 17 nsr nsr 4096 Apr  4 21:29 ..
-rw-------  1 nsr nsr  636 Apr  4 22:17 mykey
nsr@nsr-VirtualBox:~/Desktop$ cat mykey
#
# 2048 bit OpenVPN static key
#
-----BEGIN OpenVPN Static key V1-----
f424db6250d44138b7dfc3ef6213e0bf
8852c9b2b08a21d9f54b20161834c79d
6761f44794c54f801f8ad8f066187d10
636d2ffd0a48cd34aff3a63f24c24754
7daf6bd001afd560955908662e3d1d11
7bcec6c8b510a698cbdc5ea2c19a8cbe
abc2a07089794b233a6b160b1b1fd17a
7820a9a94ce749c861ca561d167487e8
6e71ff3da02e42cef0e5d6d6d9819e33
33bbeadfdbc831e062f068c34ec35795
38131c5f20f0a195b5f883a225b99296
85d455a5fd93e59e0a3e8867ea419c6a
054e592ac8caa7625c10007115435927
fe38c02455d3ceeecc57171a4754e201
df68c09155a7e322f7e8e5260c634db5
deacde78effd6d8bcd44d60ec6088563
-----END OpenVPN Static key V1-----
nsr@nsr-VirtualBox:~/Desktop$
```

Figure 9: Verification of the mykey file on VM2

4) Setup of the encrypted tunnel: In this step, the OpenVPN encrypted tunnel has been set up on both VMs to ensure that the packets going both ways (VM1 ⇔ VM2) will be able to make use of the tunnel to encrypt themselves and thus prevent any form of Man-in-the-middle type attacks. This has been done by using the command “sudo openvpn –remote 192.168.19.129 –dev tun1—ifconfig 10.4.0.1 10.4.0.2 –verb 5 –secret mykey” on VM1 (which basically tells VM1 to set up a tunnel with VM2 where the end points will be 10.4.0.1 for VM1 and 10.4.0.2 for VM2, encryption will be set using mykey and reporting of messages passed will be set at a medium level using verb5) and “sudo openvpn –remote 192.168.19.128 –dev tun1—ifconfig 10.4.0.2 10.4.0.1 –verb 5 –secret mykey” on VM2 end(which basically tells VM2 to set up a tunnel with VM1 where the end points will be 10.4.0.2 for VM2 and 10.4.0.1 for VM1, encryption will be set using mykey and reporting of messages passed will be set at a medium level using verb5).

When this is entered on both ends, the VMs will take some time to process the information and setup the tunnel. Once that has been completed a new interface called “tun1” would be created on both interfaces and the following information would be displayed in the terminal:

```
rWrWWrWRTue Apr  4 22:23:37 2023 us=288662 Peer Connection Initiated with [AF_
INET]192.168.19.129:1194
Tue Apr  4 22:23:37 2023 us=288694 WARNING: this configuration may cache passwo
rds in memory -- use the auth-nocache option to prevent this
Tue Apr  4 22:23:37 2023 us=288701 Initialization Sequence Completed
wRwWRrWRWRwRwrWRwRwrWRwrWRwrWRwrWRwrWRwrWRwrWRwrWRwrWRwrWRwrWRwrWRwrWRw
```

Figure 10: How it looks in terminal once tunnel has been set

5) Using the VPN tunnel: In this step, we will use the VPN tunnel to ping and telnet between the VMs. This would be done using the commands “ping x.x.x.x” and “telnet x.x.x.x” followed by username and password. In here, the x.x.x.x value would be 10.4.0.1 and 10.4.0.2 respectively as these are the endpoint IP addresses of the tunnel and we want to ensure that traffic can use the tunnel to pass the information between the VMs.

```
nsr@nsr-VirtualBox:~/Desktop$ ping 10.4.0.1
PING 10.4.0.1 (10.4.0.1) 56(84) bytes of data.
64 bytes from 10.4.0.1: icmp_seq=1 ttl=64 time=0.560 ms
64 bytes from 10.4.0.1: icmp_seq=2 ttl=64 time=1.57 ms
64 bytes from 10.4.0.1: icmp_seq=3 ttl=64 time=1.54 ms
64 bytes from 10.4.0.1: icmp_seq=4 ttl=64 time=1.52 ms
64 bytes from 10.4.0.1: icmp_seq=5 ttl=64 time=1.51 ms
64 bytes from 10.4.0.1: icmp_seq=6 ttl=64 time=0.434 ms
64 bytes from 10.4.0.1: icmp_seq=7 ttl=64 time=2.46 ms
64 bytes from 10.4.0.1: icmp_seq=8 ttl=64 time=2.51 ms
64 bytes from 10.4.0.1: icmp_seq=9 ttl=64 time=0.437 ms
64 bytes from 10.4.0.1: icmp_seq=10 ttl=64 time=0.443 ms
64 bytes from 10.4.0.1: icmp_seq=11 ttl=64 time=1.88 ms
64 bytes from 10.4.0.1: icmp_seq=12 ttl=64 time=0.581 ms
64 bytes from 10.4.0.1: icmp_seq=13 ttl=64 time=1.56 ms
```

Figure 11: Pinging the endpoints of the tunnel

```
nsr@nsr-VirtualBox:~/Desktop$ telnet 10.4.0.1
Trying 10.4.0.1...
Connected to 10.4.0.1.
Escape character is '^]'.
Ubuntu 20.04.1 LTS
nsr-VirtualBox login: user
Password:

Login incorrect
nsr-VirtualBox login: nsr
Password:
Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.15.0-69-generic x86_64)

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


463 updates can be installed immediately.
228 of these updates are security updates.
To see these additional updates run: apt list --upgradable

New release '22.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Your Hardware Enablement Stack (HWE) is supported until April 2025.
Last login: Tue Apr  4 21:59:21 AEST 2023 from 10.4.0.2 on pts/1
nsr@nsr-VirtualBox:~$ ls -al
total 92
drwxr-xr-x 17 nsr  nsr  4096 Apr  4 21:31 .
drwxr-xr-x  4 root root 4096 Feb  4  2021 ..
-rw-------  1 nsr  nsr  1736 Apr  4 22:06 .bash_history
-rw-r--r--  1 nsr  nsr   220 Feb  4  2021 .bash_logout
-rw-r--r--  1 nsr  nsr  3771 Feb  4  2021 .bashrc
drwx------ 13 nsr  nsr  4096 Feb  9  2021 .cache
drwx------ 12 nsr  nsr  4096 Feb  9  2021 .config
drwxr-xr-x  2 nsr  nsr  4096 Apr  4 22:14 Desktop
drwxr-xr-x  2 nsr  nsr  4096 Feb  4  2021 Documents
drwxr-xr-x  2 nsr  nsr  4096 Feb  4  2021 Downloads
drwx------  3 nsr  nsr  4096 Apr  4 21:29 .gnupg
drwx------  3 nsr  nsr  4096 Feb  4  2021 .local
drwx------  5 nsr  nsr  4096 Feb  9  2021 .mozilla
drwxr-xr-x  2 nsr  nsr  4096 Feb  4  2021 Music
-rw-------  1 nsr  nsr   636 Apr  4 21:31 mykey
drwxr-xr-x  2 nsr  nsr  4096 Feb  4  2021 Pictures
-rw-r--r--  1 nsr  nsr   807 Feb  4  2021 .profile
drwxr-xr-x  2 nsr  nsr  4096 Feb  4  2021 Public
```

Figure 12: Telnetting the endpoints of the tunnel

During this time, Wireshark has been run on the side for both VMs observing both the ens33 interface (the actual interface between the VMs that packets are travelling through) and also the tun1 interface (the virtual tunnel interface set on the ens33 which the packets are using to encrypt themselves). This had resulted in the following interesting observations:

1) For both Ping and Telnet case, on both machines, although the wireshark on ens33 realized it was following OpenVPN protocol, it was not able to recognize what type of packet it was, nor the data that was being sent through it. Furthermore, it had considered those packets as Malformed and was also able to identify where those data were going (i.e the final source and destination IP).

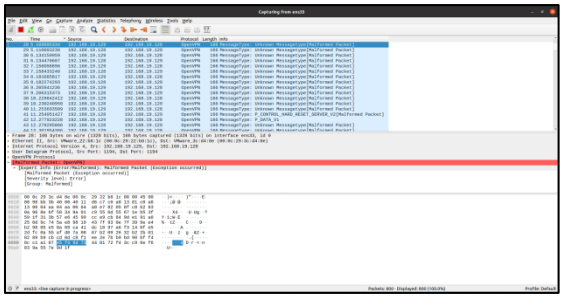

Figure 13: ens33 not being able to properly detect the traffic

2) On the other hand, the wireshark on tun1 was able to recognize the type of protocol that was being passed and see (and follow) all the data passing through the tunnel, but had identified the endpoint IP address of the tunnel as the source and destination addresses of the packets.

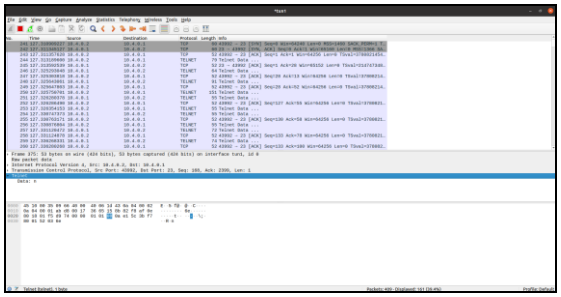

Figure 14: tun1 being able to properly detect the traffic

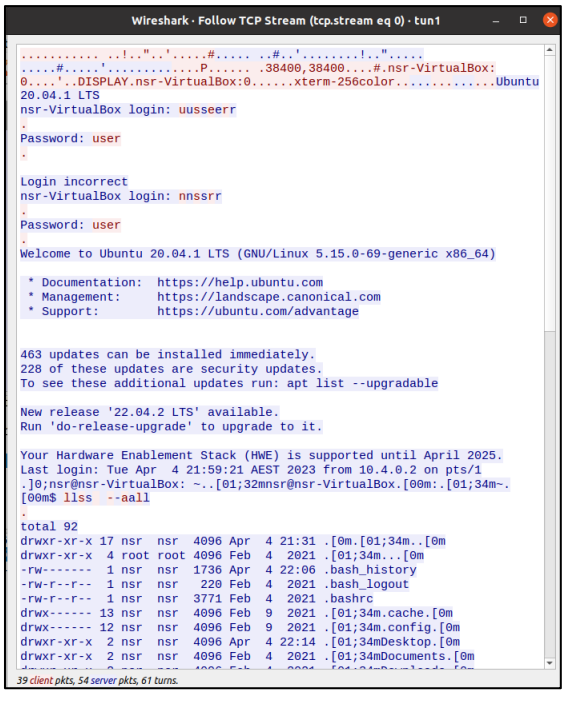

Figure 15: tun1 being able to follow the traffic

3) Moreover, in the case of telnet, ens33 had wrongly identified the Layer 4 header as UDP, whilst the tun1 had correctly identified the header as TCP.

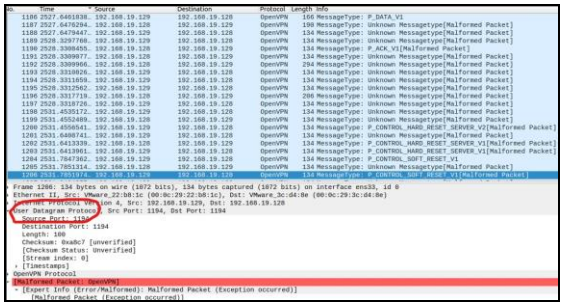

Figure 16: end33 mistakenly thinks it is UDP

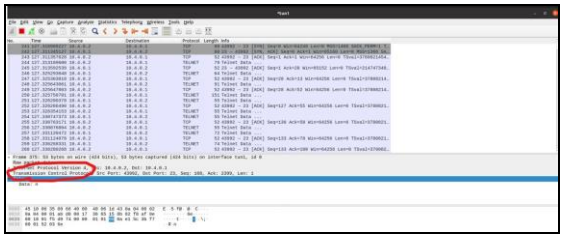

Figure 17: tun1 correctly identifies it as TCP

In order to understand why all these observations were found, one must first understand how the OpenVPN packet encapsulation works. This has been shown in using the general packet encryption (noted by T. Novickis **[16]** and L-A. Daniel, J.de Ruiter, E. Poll **[17]** which has been **shown in the figure below**:

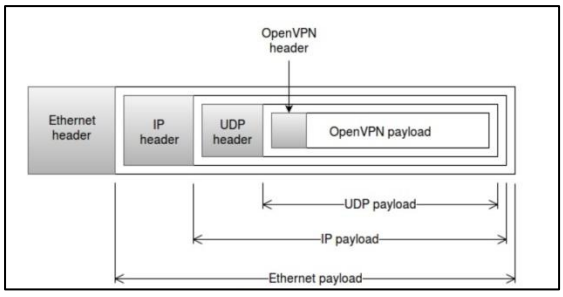

Figure 18: OpenVPN packet encapsulation **[16]**

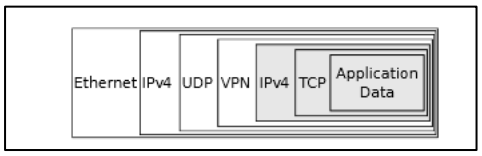

Figure 19: Detailed OpenVPN packet encapsulation **[17]**

By combining these two images we can get the following information:

At first, there is application data, which is encapsulated by TCP, which in turn is encapsulated by an IPv4 packet. This is in turn encapsulated by a VPN header and this entire section from **figure 19** is represented as OpenVPN header and OpenVPN payload in **figure 18**.

After this, the rest part is similar where the previously mentioned headers have been encapsulated by a UDP header, which in turn has been encapsulated by an IP header. After this, all of them are finally encapsulated by an Ethernet layer.

This shows that the observations made were actually sound as:

1) Ens33 only saw the packet from outside the tunnel and wasn't able to break the encryption used to protect the payload inside the OpenVPN. Hence, it was only able to see where the packet was going and not what information or packet type was being passed.

2) Tun1 saw the packet from inside the tunnel and thus did not need to break any encryption. Hence it was able to see the information in the packet and the type of packets that was actually being passed inside the OpenVPN payload. Furthermore, it was saying 10.4.0.1 and 10.4.0.2 as the source and destination addresses as these were the endpoints assigned to the VMs on the tun1 end.

c) Ens33 thought it was a UDP packet. That's because traffic going through the OpenVPN tunnel prefers to use Layer 4 header of UDP (on the protocol outside the OpenVPN header) for the packets **[9] [16]** to avoid "TCP meltdown issue" (as each TCP layer has its own transmission and congestion issue and interaction between the multiple TCP layers

can lead to huge increase in congestion and delay) whilst hiding the fact that it's actually using the TCP as layer 4 (on the protocol inside the OpenVPN header) for the packet. Thus, by not being able to see the information stored inside, ens33 had mistakenly assumed the entire packet to be a UDP one.

Conclusion

Overall, in this report, a brief introduction to VPN tunnel, its tunneling protocols and its relevance to organizational security policy has been noted. After that, a brief explanation regarding OpenVPN has also been provided, along with the steps (including rational behind the steps and their verification processes) that are needed to set up a simple VPN tunnel.

Once, the VPN was fully set and functional, some interesting observations had been noticed on the interfaces associated with the setup. These had been later analyzed and explained with the help of OpenVPN's packet's encapsulation process.

But does this mean, OpenVPN has been completely explored though this process?

Not necessarily.

That's because, here we only tested for transmission of "Ping" and "Telnet". But we have not tested for other variations like IMAP, SSH, etc. Furthermore, we have also only tested with small traffic between two VMs, and not with huge traffic between several VMs (or between VMs that have several devices and machines connected in between them).

Hence, if the purpose of the lab was to only gain a basic understanding of how it works (and what it can and can't do), then it has been fully successful. But if its aim was to ensure a complete and absolute understanding of everything OpenVPN can and can't do, then it would be better to further extend the lab considering the factors that has been mentioned in the paragraph before.

References

[1] A/Prof. P.Branch (2023). TNE30009/TNE80009 - Laboratory Session 3 [Portable document format (pdf)]. Available: https://swinburne.instructure.com/courses/49751/pages/laboratory-week-6?module_item_id=3185485 . Accessed 20 Apr. 2023.

[2] Wikipedia Contributors. "Virtual Private Network." *Wikipedia*, Wikimedia Foundation, 25 Apr. 2019, https://en.wikipedia.org/wiki/Virtual_private_network . Accessed 20 Apr. 2023.

[3] A/Prof. P.Branch (2023). Network Security and Resilience - VPNs - lecture fourteen [Portable document format (pdf)]. Available: https://swinburne.instructure.com/courses/49751/pages/lectures-week-5?module_item_id=3185481 . Accessed 20 Apr. 2023.

[4] Microsoft Corporation. "Understanding Point-To-Point Tunneling Protocol (PPTP)." *Unipd.it*, 2022, wwwdisc.chimica.unipd.it/luigino.feltre/pubblica/unix/winnt_doc/pppt/understanding_pptp.html. Accessed 20 Apr. 2023.

[5] A/Prof. P.Branch (2023). Network Security and Resilience – Virtual Private Networks - lecture thirteen [Portable document format (pdf)]. Available: https://swinburne.instructure.com/courses/49751/pages/lectures-week-5?module_item_id=3185481 . Accessed 20 Apr. 2023.

[6] IBM. "Layer 2 Tunnel Protocol." *Www.ibm.com*, updated 11 Apr. 2023, www.ibm.com/docs/en/i/7.4?topic=concepts-layer-2-tunnel-protocol. Accessed 20 Apr. 2023.

[7] utunnel.io. "OpenVPN vs IKEV2 vs L2TP- VPN Protocols Compared." *Www.utunnel.io*, 11 June 2020, www.utunnel.io/blog/vpn-technology/openvpn-vs-ikev2-vs-l2tp-vpn-protocols-compared. Accessed 20 Apr. 2023.

[8] proofpoint. “What Is SSTP? - SSTP VPN Protocol & Security | Proofpoint AU.” *Proofpoint*, 16 Mar. 2022, www.proofpoint.com/au/threat-reference/sstp. Accessed 20 Apr. 2023.

[9] A/Prof. P.Branch (2023). Network Security and Resilience – VPNs - lecture fifteen [Portable document format (pdf)]. Available: https://swinburne.instructure.com/courses/49751/pages/lectures-week-5?module_item_id=3185481 . Accessed 20 Apr. 2023.

[10] NCES. “Chapter 3-Security Policy: Development and Implementation, from Safeguarding Your Technology, NCES Publication 98-297 (National Center for Education Statistics).” *Ed.gov*, 2019, https://nces.ed.gov/pubs98/safetech/chapter3.asp. . Accessed 20 Apr. 2023.

[11] National Cybersecurity Society. *Virtual Private Network Policy Template*. 10 May 2019. Available: https://nationalcybersecuritysociety.org/wp-content/uploads/2019/10/VPN-Policy-Template-FINAL.pdf . Accessed 20 Apr. 2023.

[12] National Cybersecurity Society. “NCS Partner Portal | Meet Our Platinum Partners | NCS AU.” *Www.ncs.co*, 2021, www.ncs.co/en-au/partners/. Accessed 20 Apr. 2023.

[13] wikipedia. “OpenVPN.” *Wikipedia*, 26 Feb. 2021, https://en.wikipedia.org/wiki/OpenVPN. Accessed 21 Apr. 2023.

[14] Rimeikis, Antanas. “What Is OpenVPN and What Does It Have to Do with Your VPN?” *Surfshark*, 14 Nov. 2022, https://surfshark.com/blog/what-is-openvpn. Accessed 21 Apr. 2023.

[15] Mardisalu, Rob. “Best VPN Protocols: OpenVPN vs PPTP vs L2TP vs Others (Comparison).” *TheBestVPN.com*, 9 June 2018, https://thebestvpn.com/pptp-l2tp-openvpn-sstp-ikev2-protocols/. Accessed 21 Apr. 2023.

[16] T. Novickis. “Protocol state fuzzing of an OpenVPN”. Master Thesis, Faculty of Sciecne, Radboud University, Nijmegen in Netherlands, 2016 [Portable document format (pdf)]. Available: www.ru.nl/publish/pages/769526/tomas_novickis.pdf Accessed 21 Apr. 2023.

[17] L. -A. Daniel, E. Poll and J. de Ruiter, "Inferring OpenVPN State Machines Using Protocol State Fuzzing," *2018 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW)*, London, UK, 2018, pp. 11-19, doi: 10.1109/EuroSPW.2018.00009. Accessed 21 Apr. 2023.