

ELEC-C7420 Basic Principles in Networking Spring 2022

Assignment VII: Ipsec & VPN



PSALTAKIS GEORGIOS

Aalto University
School of Engineering

Goals of the experiment

This experiment is a quite simple implementation of VPN network using socket programming for the traffic we want to sniff. This time the socket is not implemented on a local host but on the ip address for it to be affected by the VPN service. This assignment is to give us an understandting on how vpn tunneling works and get a bigger picture of encrypted/unencrypted VPN service providers.

Experimental Setup

The Experimental Setup is a simple python Server and a client that keeps sending on a loop a simple txt file . This is first done Without VPN and using wireshark to try and sniff the packets. The next case scenario is by deploying the the server and client in the IP adress of the VPN provider. I used both Aalto VPN service as well as my personal one TunnelBear.

ANNEX

SERVER CODE

```
import socket
port = 9999
s = socket.socket()
host = "192.168.241.102"

#host = "130.233.20.72"
s.bind((host, port))
s.listen(5)

print ('Server listening....')

while True:
    conn, addr = s.accept()
```

```
        print ('Got connection from'), addr
        data = conn.recv(1024)
        print('Received', repr(data))

        filename='/Users/georgepsaltakis/Desktop/ASSIGMENT 7/hi.txt'
        f = open(filename,'rb')
        l = f.read(1024)
        while (l):
            conn.send(l)
            print('Sent ',repr(l))
            conn.send('Thank you for receiving file'.encode())
```

ANNEX

CLIENT CODE

```
import socket
import time
s = socket.socket()
host = "192.168.241.102"
#host = "130.233.20.72"
port = 9999

s.connect((host, port))
s.send('Hello server!'.encode())
while True:
    time.sleep(2)
```

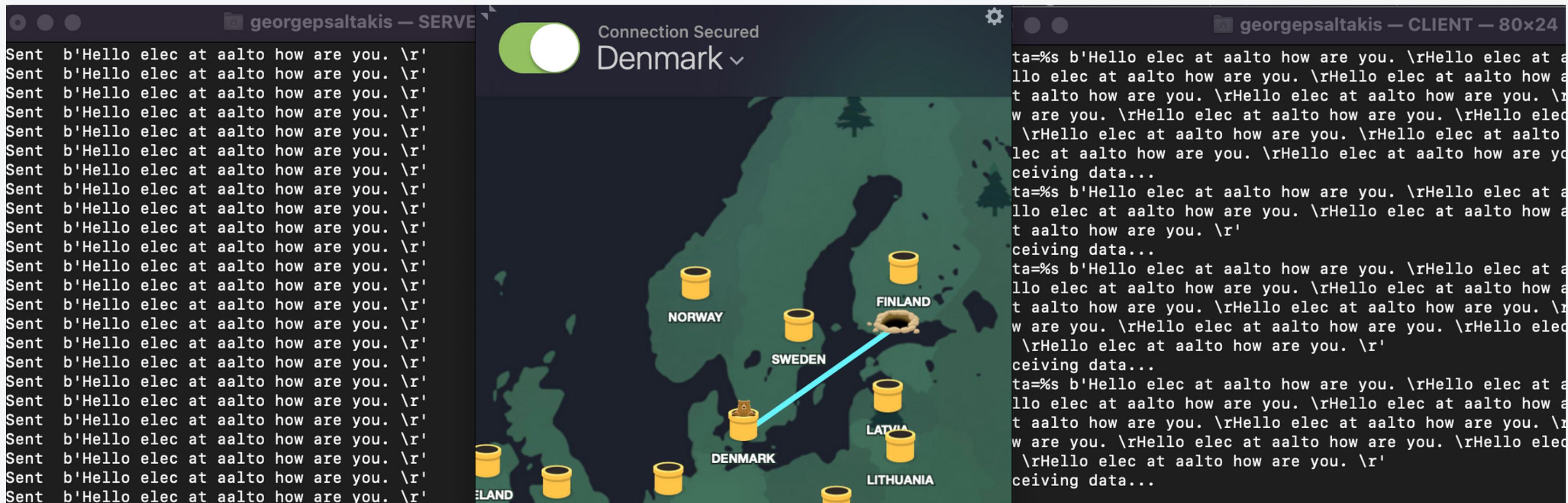
```
with open('received_file', 'wb') as f:
    print ('file opened')
    while True:
        print('receiving data...')
        time.sleep(1)
        data = s.recv(1024)
        print('data=%s', (data))
        if not data:
            break
        f.write(data)
```

Results & Conclusion

Successful implementation of server and client

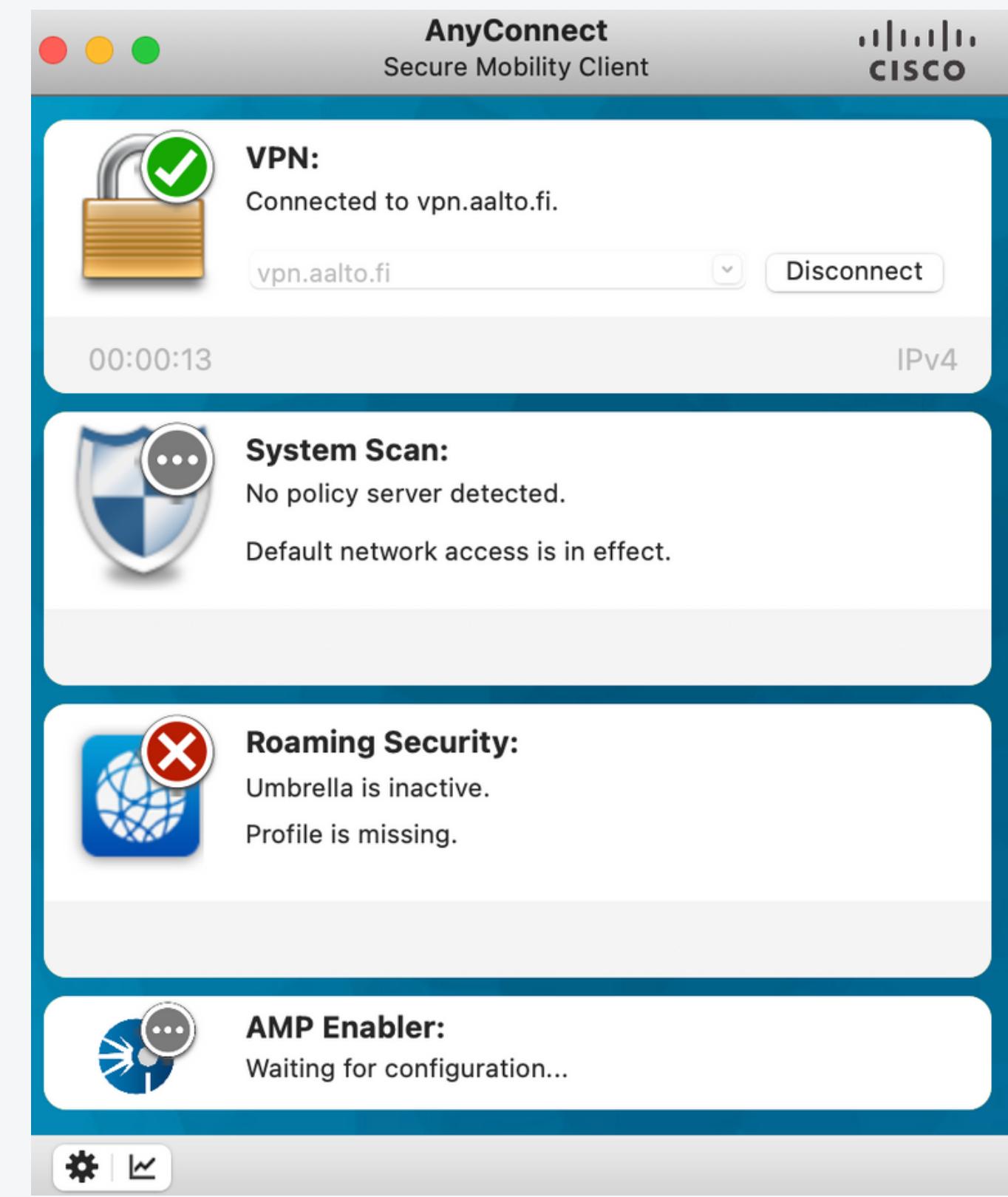
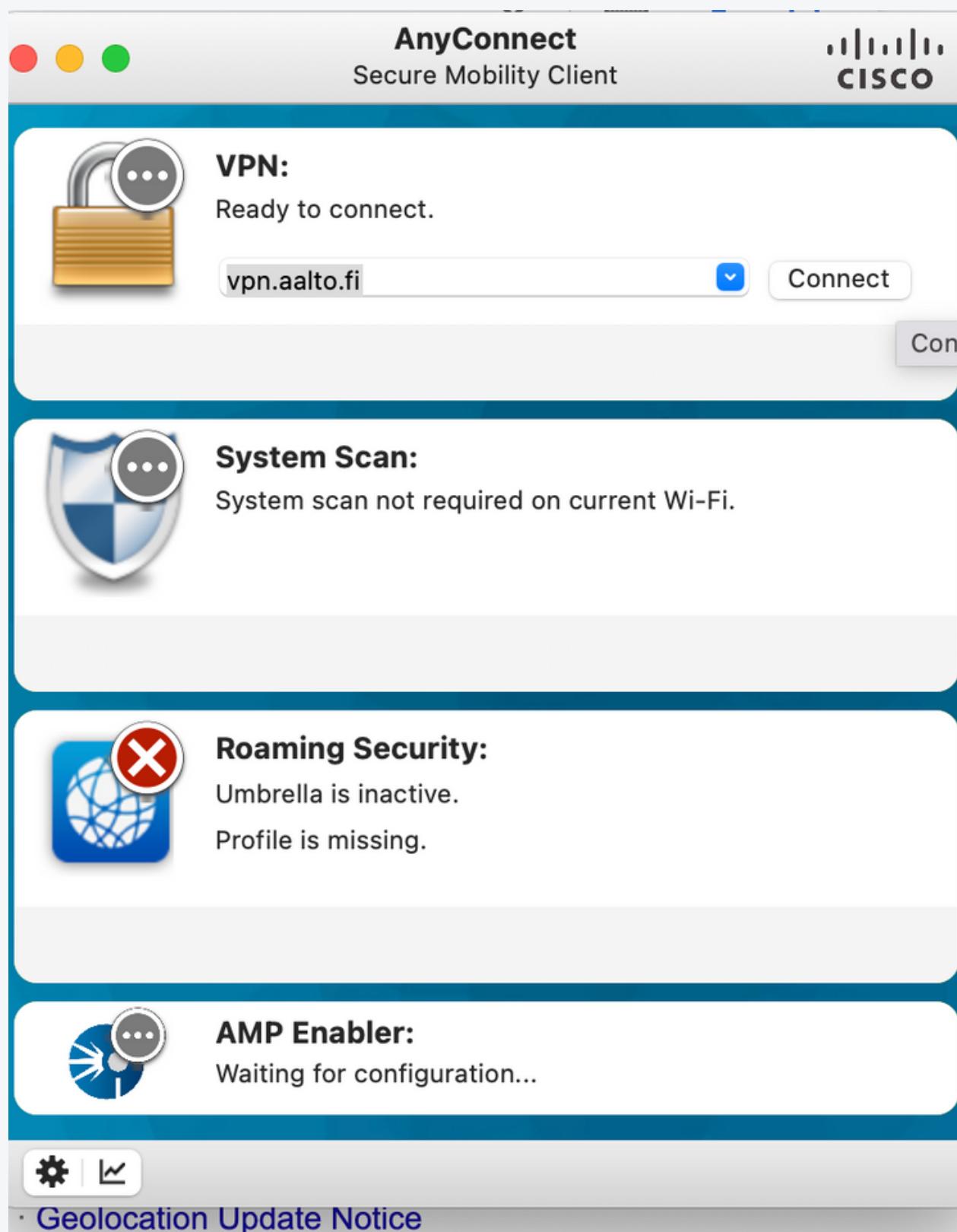
Here we can see the successfull implementation of the server and the Client.

Successful communication between server and client using VPN



I used my personal and university provided VPN and it works with both of them correctly

VPN SETUP OF AALTO.



PACKAGE CAPTURE AT WIRESHARK (Without VPN)

Capturing from Loopback: lo0

ip.src == 192.168.241.102

No.	Time	Source	Destination	Protocol	Length	Info
31	21.260934	192.168.241.102	192.168.241.102	TCP	56	60588 → 9999 [ACK] Seq=14 Ack=579 Win=407680 Len=0
32	21.260946	192.168.241.102	192.168.241.102	TCP	192	9999 → 60588 [PSH, ACK] Seq=579 Ack=14 Win=408256 Len=0
33	21.260956	192.168.241.102	192.168.241.102	TCP	56	60588 → 9999 [ACK] Seq=14 Ack=715 Win=407552 Len=0
34	21.260968	192.168.241.102	192.168.241.102	TCP	192	9999 → 60588 [PSH, ACK] Seq=715 Ack=14 Win=408256 Len=0
35	21.260979	192.168.241.102	192.168.241.102	TCP	56	60588 → 9999 [ACK] Seq=14 Ack=851 Win=407424 Len=0
36	21.260990	192.168.241.102	192.168.241.102	TCP	192	9999 → 60588 [PSH, ACK] Seq=851 Ack=14 Win=408256 Len=0
37	21.261000	192.168.241.102	192.168.241.102	TCP	56	60588 → 9999 [ACK] Seq=14 Ack=987 Win=407296 Len=0
38	21.261012	192.168.241.102	192.168.241.102	TCP	192	9999 → 60588 [PSH, ACK] Seq=987 Ack=14 Win=408256 Len=0
39	21.261023	192.168.241.102	192.168.241.102	TCP	56	60588 → 9999 [ACK] Seq=14 Ack=1123 Win=407168 Len=0
40	21.261034	192.168.241.102	192.168.241.102	TCP	192	9999 → 60588 [PSH, ACK] Seq=1123 Ack=14 Win=408256 Len=0
41	21.261044	192.168.241.102	192.168.241.102	TCP	56	60588 → 9999 [ACK] Seq=14 Ack=1259 Win=407040 Len=0
42	21.261056	192.168.241.102	192.168.241.102	TCP	192	9999 → 60588 [PSH, ACK] Seq=1259 Ack=14 Win=408256 Len=0
43	21.261066	192.168.241.102	192.168.241.102	TCP	56	60588 → 9999 [ACK] Seq=14 Ack=1395 Win=406848 Len=0
44	21.261078	192.168.241.102	192.168.241.102	TCP	192	9999 → 60588 [PSH, ACK] Seq=1395 Ack=14 Win=408256 Len=0
45	21.261088	192.168.241.102	192.168.241.102	TCP	56	60588 → 9999 [ACK] Seq=14 Ack=1531 Win=406720 Len=0
46	21.261100	192.168.241.102	192.168.241.102	TCP	192	9999 → 60588 [PSH, ACK] Seq=1531 Ack=14 Win=408256 Len=0
47	21.261111	192.168.241.102	192.168.241.102	TCP	56	60588 → 9999 [ACK] Seq=14 Ack=1667 Win=406592 Len=0
48	21.261123	192.168.241.102	192.168.241.102	TCP	192	9999 → 60588 [PSH, ACK] Seq=1667 Ack=14 Win=408256 Len=0
49	21.261133	192.168.241.102	192.168.241.102	TCP	56	60588 → 9999 [ACK] Seq=14 Ack=1803 Win=406464 Len=0
50	21.261145	192.168.241.102	192.168.241.102	TCP	192	9999 → 60588 [PSH, ACK] Seq=1803 Ack=14 Win=408256 Len=0
51	21.261155	192.168.241.102	192.168.241.102	TCP	56	60588 → 9999 [ACK] Seq=14 Ack=1939 Win=406336 Len=0
52	21.261168	192.168.241.102	192.168.241.102	TCP	192	9999 → 60588 [PSH, ACK] Seq=1939 Ack=14 Win=408256 Len=0

Window: 6379

Hex	Dec
0000	02 00 00 00 45 00 00 bc
0010	c0 a8 f1 66 c0 a8 f1 66
0020	27 0f ec ac 5a 04 07 f0
0030	... E .. .@. @..
0040	... f... f ' ... Z...
0050	J\U..... d.....
0060	..t./' C Hello el
0070	ec at aa lto how
0080	are you. .Hello
0090	elec at aalto ho

Explanation of the above image

This is without any VPN at all. Thats the reason the IP address is the networks address as normally. The IP address is 192.168.241.102

We can see all the traffic since this is an open wifi and there is no encrypted VPN in use. This is also seen since the sender and reciever is the same and we use TCP protocol. So we could easily isolate the package in sent and received by the python script i wrote.

PACKAGE CAPTURE AT WIRESHARK (With Aalto VPN)

Loopback: lo0

ip.src == 130.233.20.72 && ip.dst == 130.233.20.72

No.	Time	Source	Destination	Protocol	Length	Info
123	45.567496	130.233.20.72	130.233.20.72	TCP	44	60393 → 9999 [ACK] Seq=1 Ack=1 Win=65535 Len=0
124	45.567644	130.233.20.72	130.233.20.72	TCP	57	60393 → 9999 [PSH, ACK] Seq=1 Ack=1 Win=65535 Len=0
125	45.567676	130.233.20.72	130.233.20.72	TCP	44	9999 → 60393 [ACK] Seq=1 Ack=14 Win=65535 Len=0
126	45.569689	130.233.20.72	130.233.20.72	TCP	78	9999 → 60393 [PSH, ACK] Seq=1 Ack=14 Win=65535
127	45.569710	130.233.20.72	130.233.20.72	TCP	44	60393 → 9999 [ACK] Seq=14 Ack=35 Win=65535
128	45.569719	130.233.20.72	130.233.20.72	TCP	78	9999 → 60393 [PSH, ACK] Seq=35 Ack=14 Win=65535
129	45.569725	130.233.20.72	130.233.20.72	TCP	44	60393 → 9999 [ACK] Seq=14 Ack=69 Win=65535
130	45.569731	130.233.20.72	130.233.20.72	TCP	112	9999 → 60393 [PSH, ACK] Seq=69 Ack=14 Win=65535
131	45.569736	130.233.20.72	130.233.20.72	TCP	44	60393 → 9999 [ACK] Seq=14 Ack=137 Win=65535
132	45.569740	130.233.20.72	130.233.20.72	TCP	78	9999 → 60393 [PSH, ACK] Seq=137 Ack=14 Win=6553
133	45.569745	130.233.20.72	130.233.20.72	TCP	44	60393 → 9999 [ACK] Seq=14 Ack=171 Win=65535
134	45.569750	130.233.20.72	130.233.20.72	TCP	78	9999 → 60393 [PSH, ACK] Seq=171 Ack=14 Win=6553
135	45.569755	130.233.20.72	130.233.20.72	TCP	44	60393 → 9999 [ACK] Seq=14 Ack=205 Win=65535
136	45.569760	130.233.20.72	130.233.20.72	TCP	78	9999 → 60393 [PSH, ACK] Seq=205 Ack=14 Win=6553
137	45.569765	130.233.20.72	130.233.20.72	TCP	44	60393 → 9999 [ACK] Seq=14 Ack=239 Win=65535
138	45.569770	130.233.20.72	130.233.20.72	TCP	112	9999 → 60393 [PSH, ACK] Seq=239 Ack=14 Win=6553
139	45.569775	130.233.20.72	130.233.20.72	TCP	44	60393 → 9999 [ACK] Seq=14 Ack=307 Win=65535
140	45.569780	130.233.20.72	130.233.20.72	TCP	78	9999 → 60393 [PSH, ACK] Seq=307 Ack=14 Win=6553
141	45.569785	130.233.20.72	130.233.20.72	TCP	44	60393 → 9999 [ACK] Seq=14 Ack=341 Win=65535
142	45.569790	130.233.20.72	130.233.20.72	TCP	78	9999 → 60393 [PSH, ACK] Seq=341 Ack=14 Win=6553
143	45.569795	130.233.20.72	130.233.20.72	TCP	44	60393 → 9999 [ACK] Seq=14 Ack=375 Win=65535
144	45.569800	130.233.20.72	130.233.20.72	TCP	112	9999 → 60393 [PSH, ACK] Seq=375 Ack=14 Win=65535

Window: 65535

Hex	Dec	ASCII
0000	02 00 00 00 45 00 00 4a 00 00 40 00 40 06 00 00	...E J @ @ . . .
0010	82 e9 14 48 82 e9 14 48 27 0f eb e9 31 9e 36 20	...H H ' 1 6
0020	ff 6a 65 71 50 18 ff ff 2e 9f 00 00 48 65 6c 6c	.jeqP . Hell
0030	6f 20 65 6c 65 63 20 61 74 20 61 61 6c 74 6f 20	o elec a t aalto
0040	68 6f 77 20 61 72 65 20 79 6f 75 2e 20 0d	how are you. .

Explanation of the above image

This is with the Aalto VPN. I can see the appended IP address of the Server as 130.233.20.72 as well as the receiver of the package. I can also understand due to the fact of the TCP protocol in use as well as the sniffed package. This is of Course is happening due to the fact that not all the VPNs are using encryption as is with this case to just connect to the intranet of the University. That is also the reason why we can see the package normally.

PACKAGE CAPTURE AT WIRESHARK (With Encrypted VPN)

The screenshot shows a Wireshark interface with the following details:

- Panels:** Top panel: Standard Wireshark toolbar with icons for file, edit, search, and zoom. Middle panel: "utun3" interface selection. Bottom panels: "TCP segment data (1220 bytes)" and "Hex dump" panes.
- Packet List:** Shows 1085 total packets. The 1083rd packet is highlighted in red and selected. Other packets are in light blue or white.
- Selected Packet Details:** The selected packet (No. 1083) is shown in the details pane:
 - Source: 172.18.13.138
 - Destination: 17.248.254.17
 - Protocol: TCP
 - Length: 44
 - Info: 60802 → 443 [RST] Seq=1 Win=0 Len=0
- Selected Packet Bytes:** The bytes pane shows the raw hex and ASCII data for the selected RST packet.

Explanation of the above image

Using a VPN that provides encryption we can get the package encrypted so that it cannot be captured and the contents of it to be seen. This is easily seen in the image above where the captured package is gibberish and we cant see anything since the contents of it have been passed by a Cipher.

Conclusions

This experiment gave me an inside understanding of how VPNs work in comparison to not using them at all. There could be an unencrypted VPN that just pass through the data through a server somewhere else so at the end of the day by using a different IP adress for the traffic (Passthrough). There could also be VPNs using encryption embedded so the data not only gets passthrough a different adress but as well as encrypt the packages making them not understandble by the sniffer . The final use is usually the best use of a VPN.