

# Analyze Network Traffic with TCPDump

## Description

I simulated that someone is trying to open SSH sessions into my workstation and decided to set up a surveillance script to catch any TCP traffic coming through as SSH.

I also did another script to catch any TCP traffic coming from a specific IP address.

## Languages and Utilities Used

- TCPDump

## Environments Used

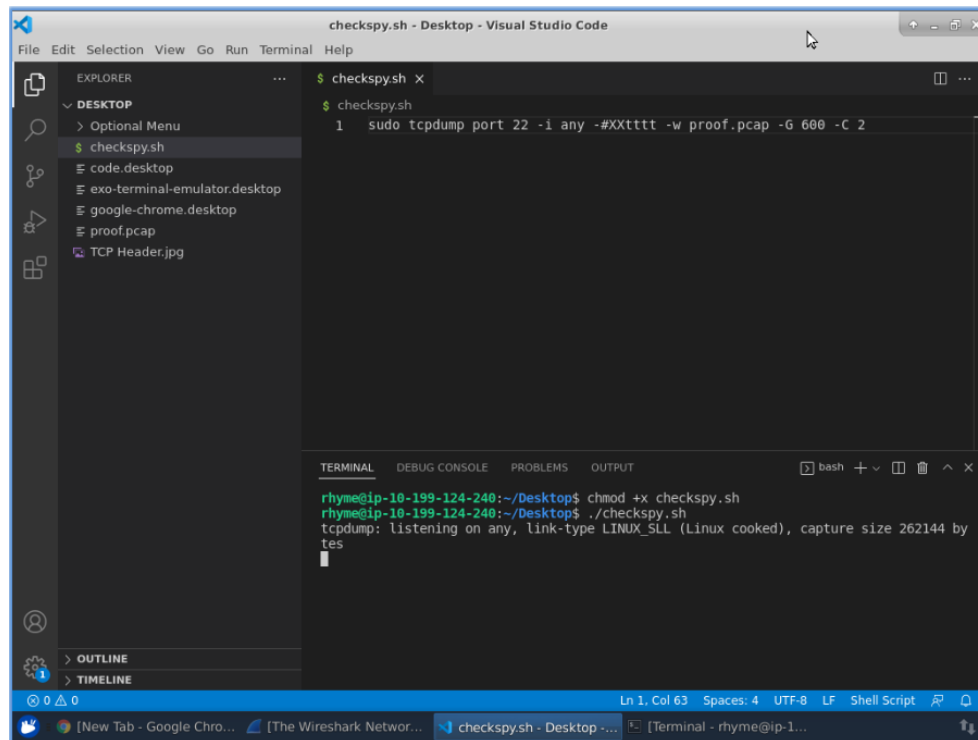
- Linux

## Program walk-through:

### Create a shell script file to capture SSH traffic and give it permissions

First I wanted to create a shell script file to capture SSH traffic with TCPDump through port 22 with the interface option *any*, since I'm testing this locally. I also wanted to print the timestamp of each captured packet in human-readable format and to write the captured packets to a file named *proof.pcap* with no more than 2 megabytes and 10 minutes of capture. To do that I used the command: `sudo tcpdump port 22 -i any -#XXtttt -w proof.pcap -G 600 -C 2`

After that, I gave it executable permissions in the terminal with the command: `chmod +x checkspy.sh`



The screenshot shows the Visual Studio Code interface. The Explorer panel on the left lists files in the 'DESKTOP' directory: 'Optional Menu', 'checkspy.sh', 'code.desktop', 'exo-terminal-emulator.desktop', 'google-chrome.desktop', 'proof.pcap', and 'TCP Header.jpg'. The Editor panel displays the 'checkspy.sh' script with the following content:

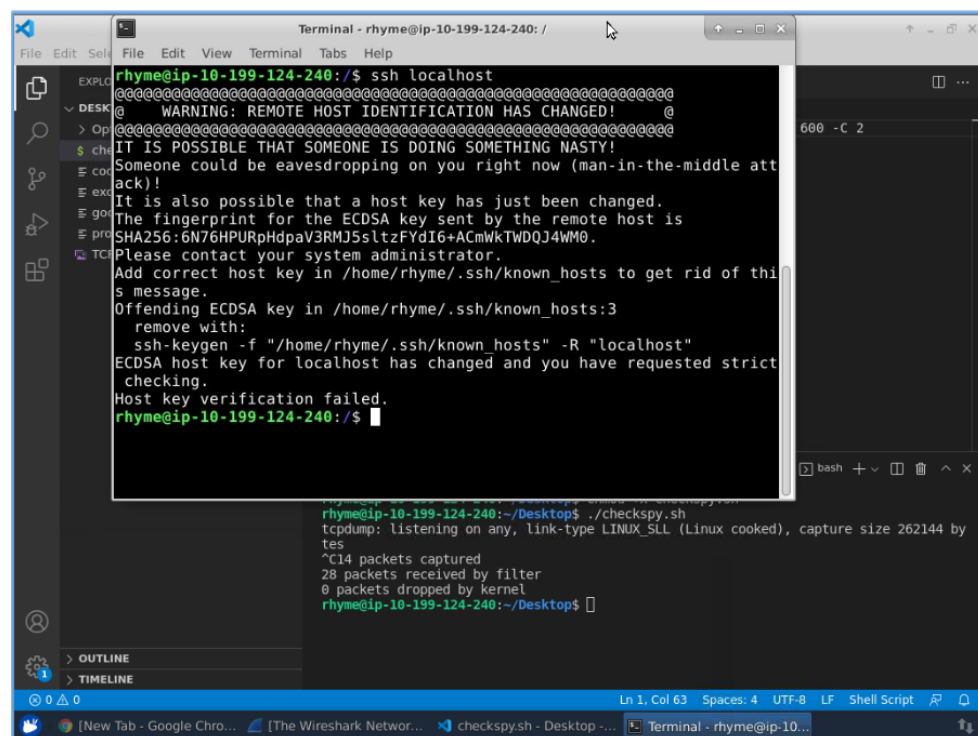
```
$ checkspy.sh
1 sudo tcpdump port 22 -i any -#XXtttt -w proof.pcap -G 600 -C 2
```

The TERMINAL panel at the bottom shows the execution of the script:

```
rhyme@ip-10-199-124-240:~/Desktop$ chmod +x checkspy.sh
rhyme@ip-10-199-124-240:~/Desktop$ ./checkspy.sh
tcpdump: listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 by tes
```

## Test the script

After launching the script to test it, I opened a terminal window and ran SSH localhost (it will not be successful, but it will generate SSH traffic)



The screenshot shows a terminal window titled 'Terminal - rhyme@ip-10-199-124-240: /'. The user runs the command `ssh localhost`, which results in a warning message about a changed remote host identification:

```
rhyme@ip-10-199-124-240:/$ ssh localhost
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@ WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED! @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that a host key has just been changed.
The fingerprint for the ECDSA key sent by the remote host is
SHA256:6N76HPURpHdpav3RMJ5sLtzFYdI6+ACmWKTWDQJ4WM0.
Please contact your system administrator.
Add correct host key in /home/rhyme/.ssh/known_hosts to get rid of this message.
Offending ECDSA key in /home/rhyme/.ssh/known_hosts:3
remove with:
  ssh-keygen -f "/home/rhyme/.ssh/known_hosts" -R "localhost"
ECDSA host key for localhost has changed and you have requested strict checking.
Host key verification failed.
rhyme@ip-10-199-124-240:/$
```

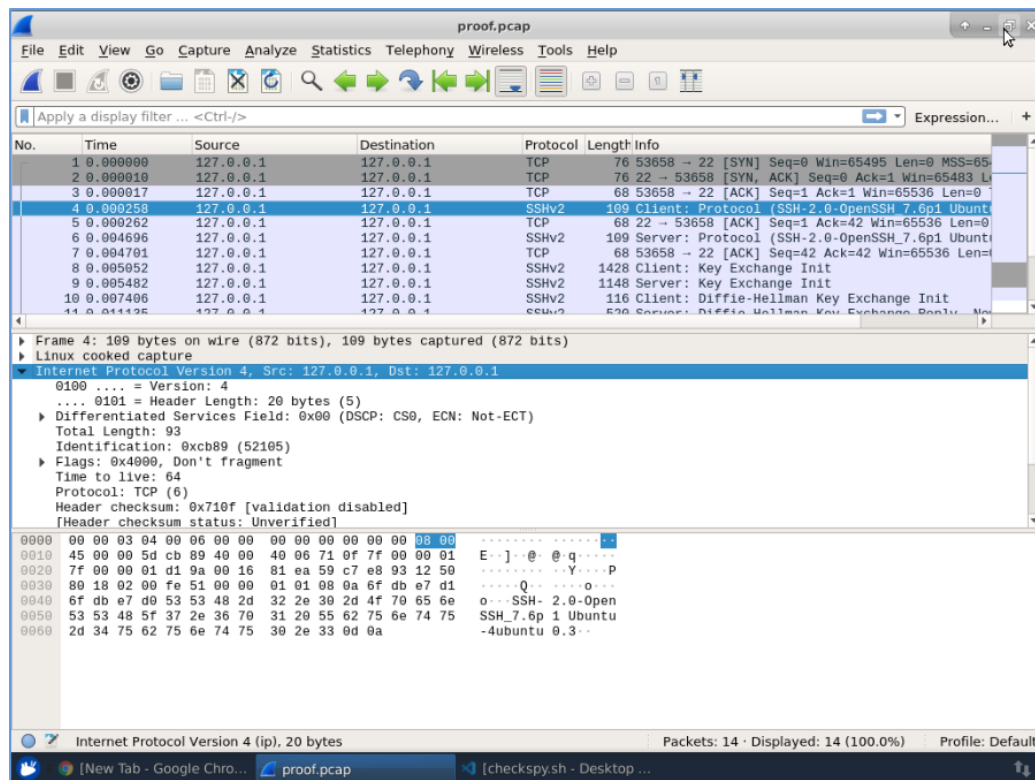
The background terminal window shows the output of the `checkspy.sh` script:

```
rhyme@ip-10-199-124-240:~/Desktop$ ./checkspy.sh
tcpdump: listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 by tes
^C14 packets captured
28 packets received by filter
0 packets dropped by kernel
rhyme@ip-10-199-124-240:~/Desktop$
```

As expected SSH localhost failed but generated some packets as shown in the terminal

## Analyze the dump file with Wireshark

The SSH traffic is easily noticeable on Wireshark. Though the source IP is from our host machine, it would appear vastly different in real-world scenarios.

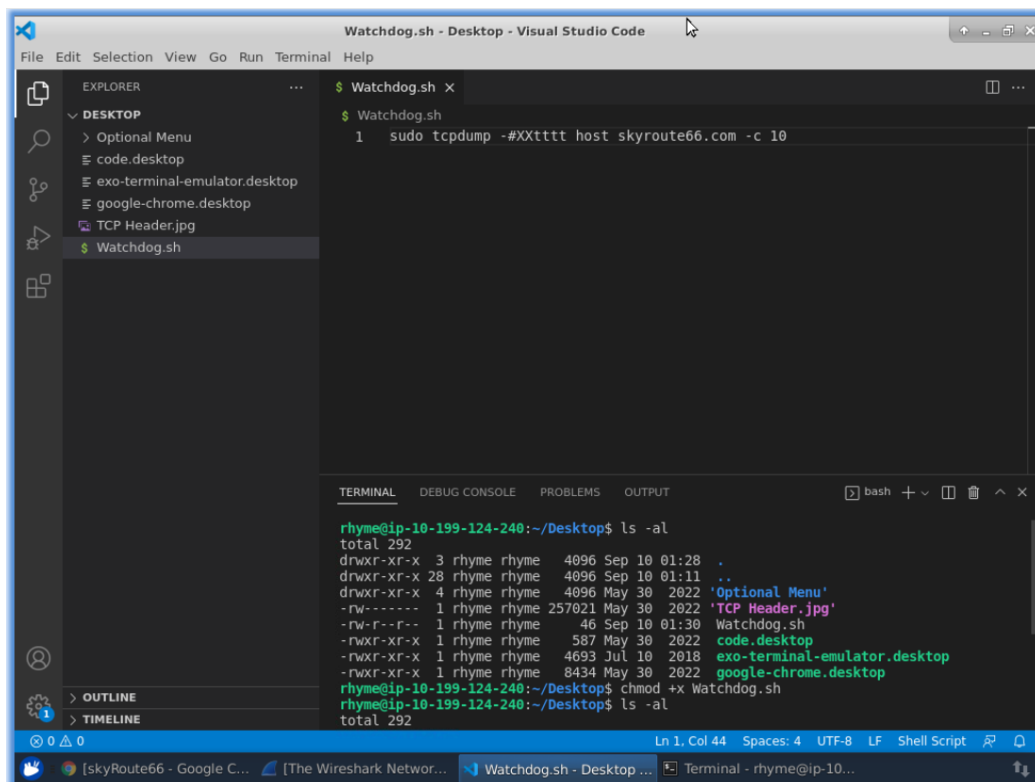


If there were numerous packets involved, it would be easier to locate them quickly by using display filters, in this case the Wireshark filter named `ssh`.

## Create a shell script file to capture traffic coming from a specific IP address and give it permissions

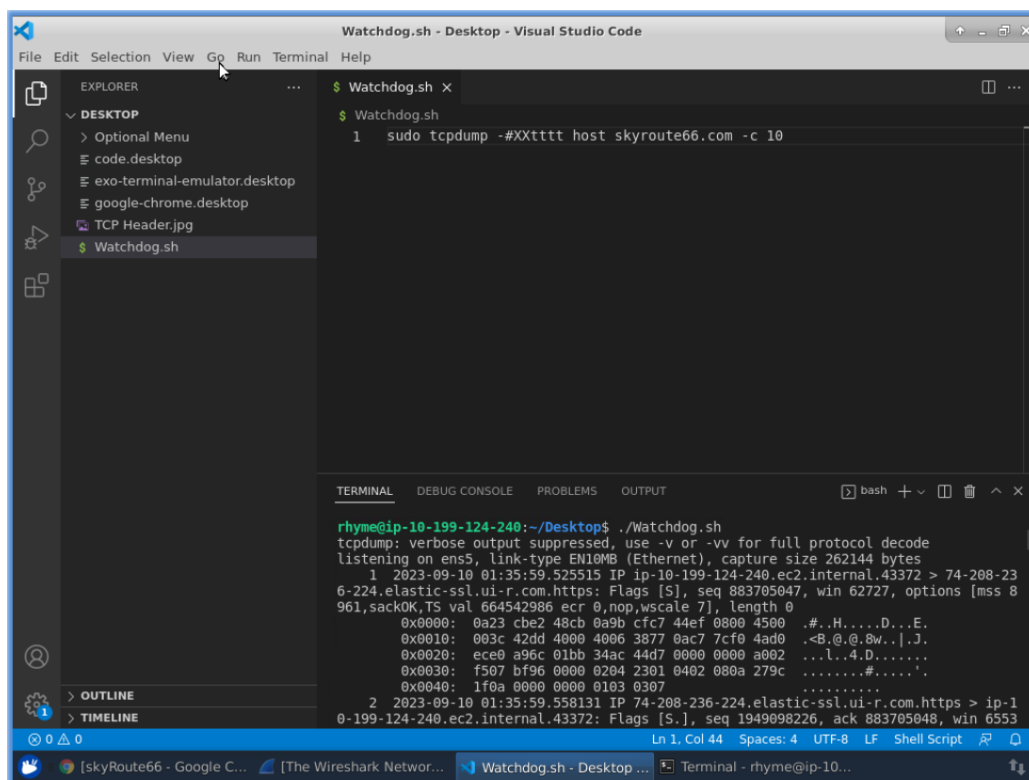
So I wanted to create a shell script file to capture 10 packets coming from *skyroute66.com* with TCPDump in which the timestamp of each captured packet is shown in human-readable format, to do that I used the command: `sudo tcpdump -#XXtttt host skyroute66.com -c 10`

After that, I gave it executable permissions in the terminal with the command: `chmod +x Watchdog.sh`

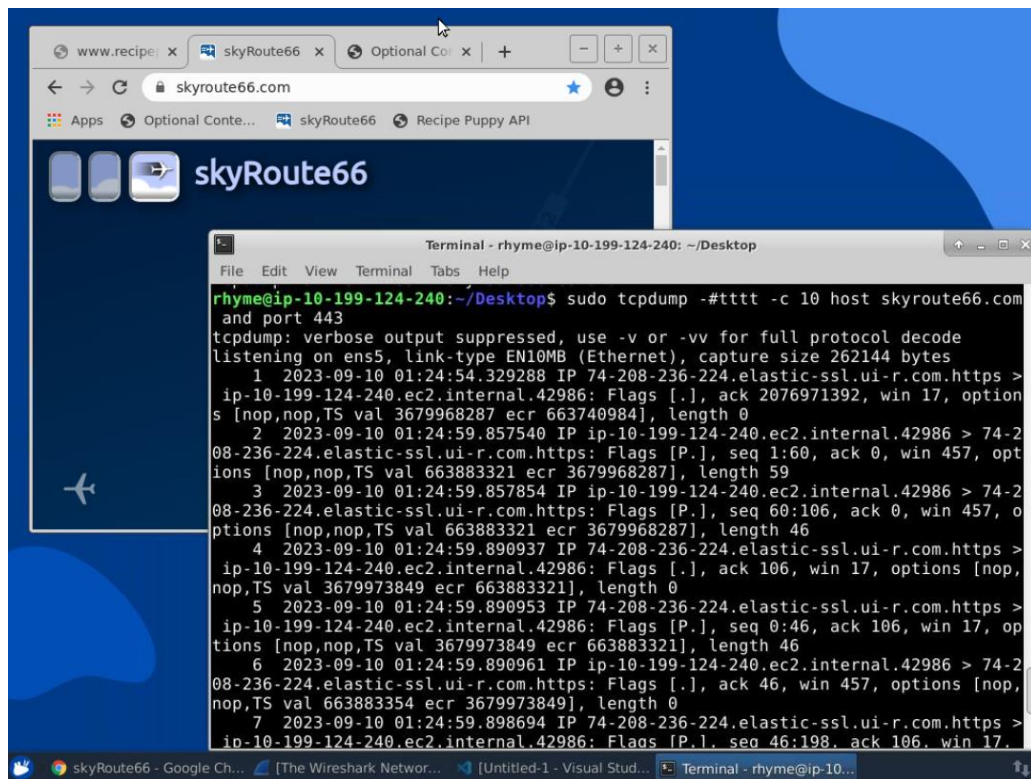


## Test the script

To test the script I started it in the terminal and opened the website *skyroute66.com*



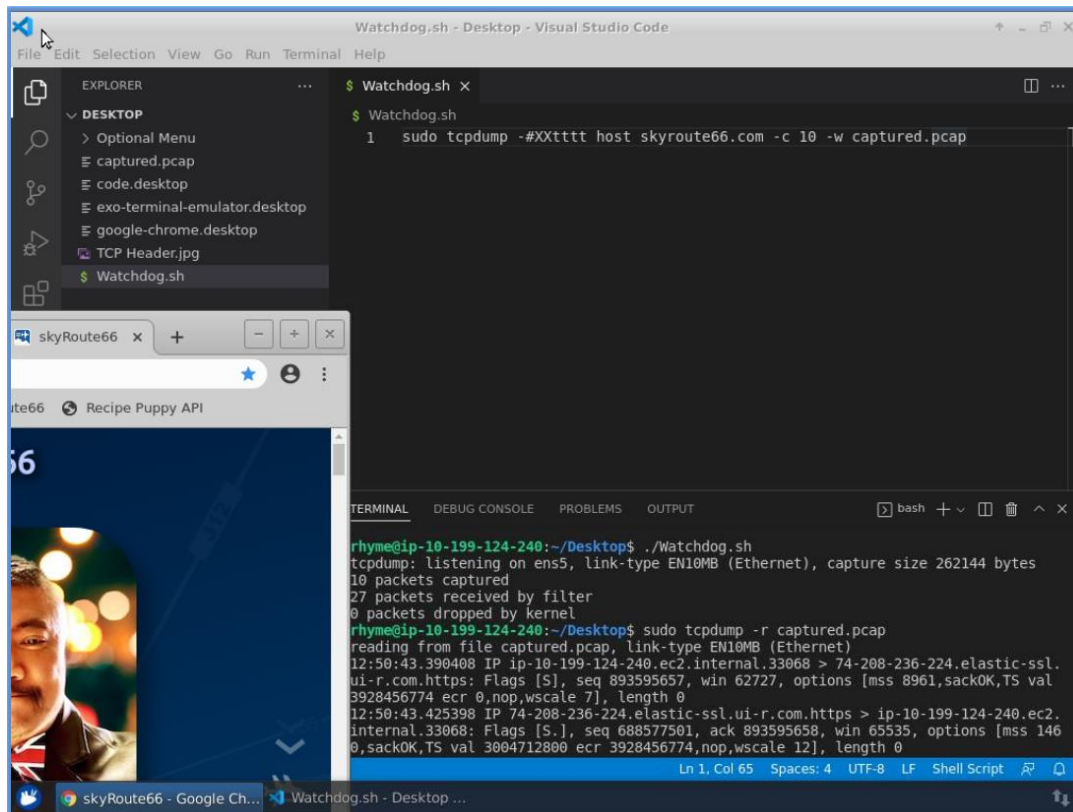
I could also use the following command to capture the same traffic with the same filters but only the traffic coming through port 443 (HTTPS traffic): `sudo tcpdump -#tttt -c 10 host skyroute66.com and port 443`



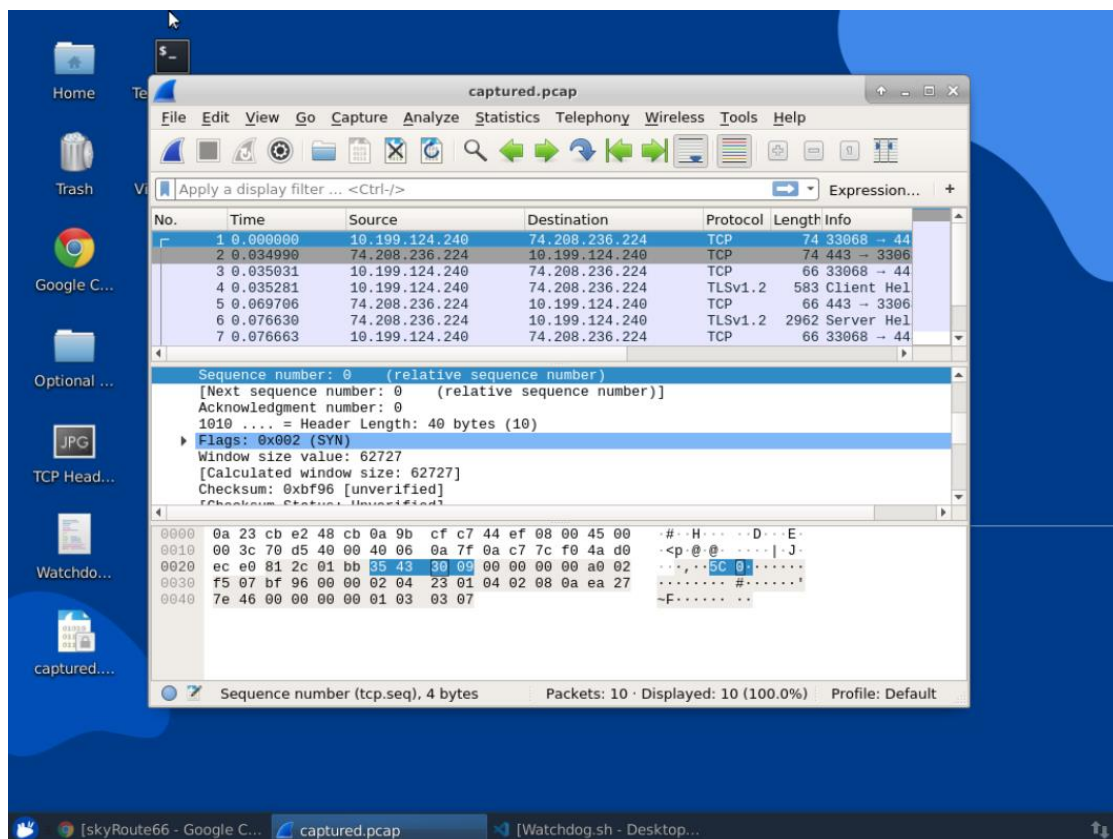
## Add the option to create a dump file to the script and test

For the script to create a file with the captured packages I need to add the `-w` option, making the script look like this: `sudo tcpdump -#XXtttt host skyroute66.com -c 10 -w captured.pcap`

Then I ran the script, opened the website and read the file in the terminal with the command: `sudo tcpdump -r captured.pcap`



For a better analysis of the packets, I could analyze the dump file with Wireshark



## Advanced Filtering: Zooming into Specific Narratives

Simulating that I just wanted to capture a specific part of a protocol, I used the following command to capture only HTTP GET requests: `sudo tcpdump -#XXtttt 'tcp[((tcp[12:1] & 0xf0) >> 2):4] = 0x47455420'`

