

## Abstract

This project examined network traffic on a virtual machine by generating both encrypted and unencrypted traffic. Unencrypted packets were sent between a client and host server, and a connection was established to an unencrypted website generated using python. All packets were captured and analyzed to see what information could be accessed by a third party. The results indicate that most data is visible to other users on the network when the traffic is unencrypted.

## Introduction

Kali Linux was used and run in a VirtualBox virtual machine, managed by Oracle VirtualBox, these provide the following applications. Netcat is used to send and receive unencrypted data between a host server and client. WireShark will be used for packet capture, as well as analysis. Python console is used to host a saved webpage (<https://www.google.com/>) as a local, unencrypted web page with the url (<http://localhost:8080>)

Netcat host server (listener) commands used:

```
nc -l 10.0.2.15 -p 31337
```

Netcat client (sender) commands used:

```
nc 10.0.2.15 -p 31337
```

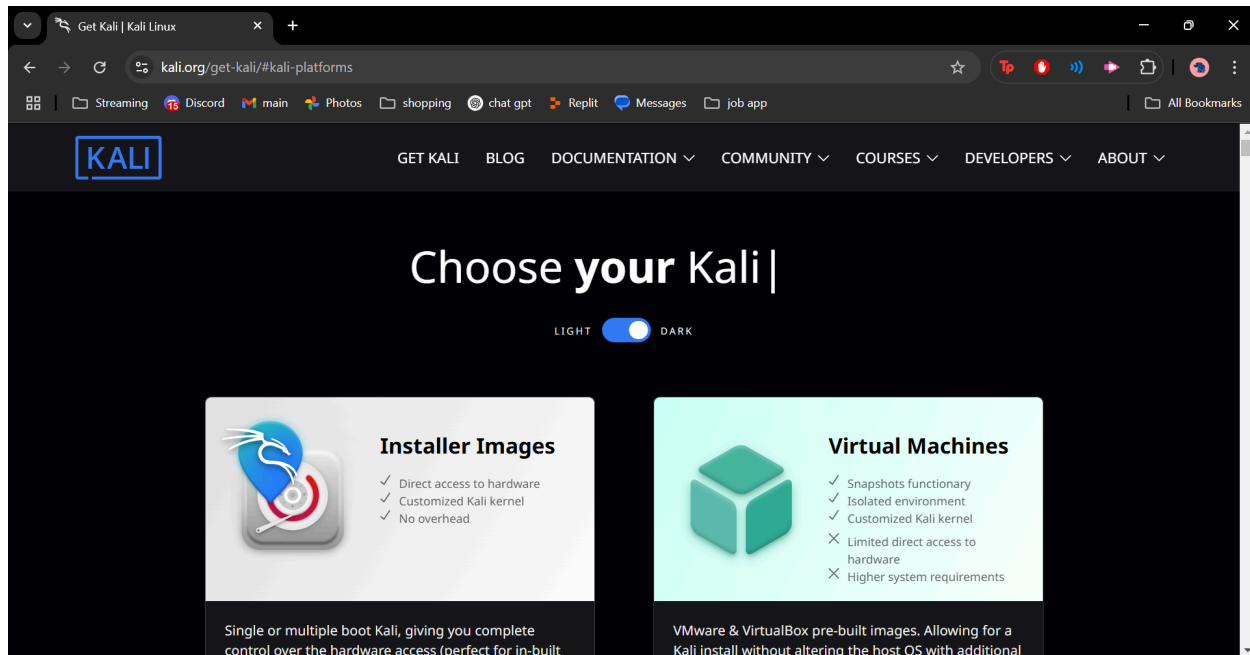
Python console commands used:

```
python -m http.server 8080
```

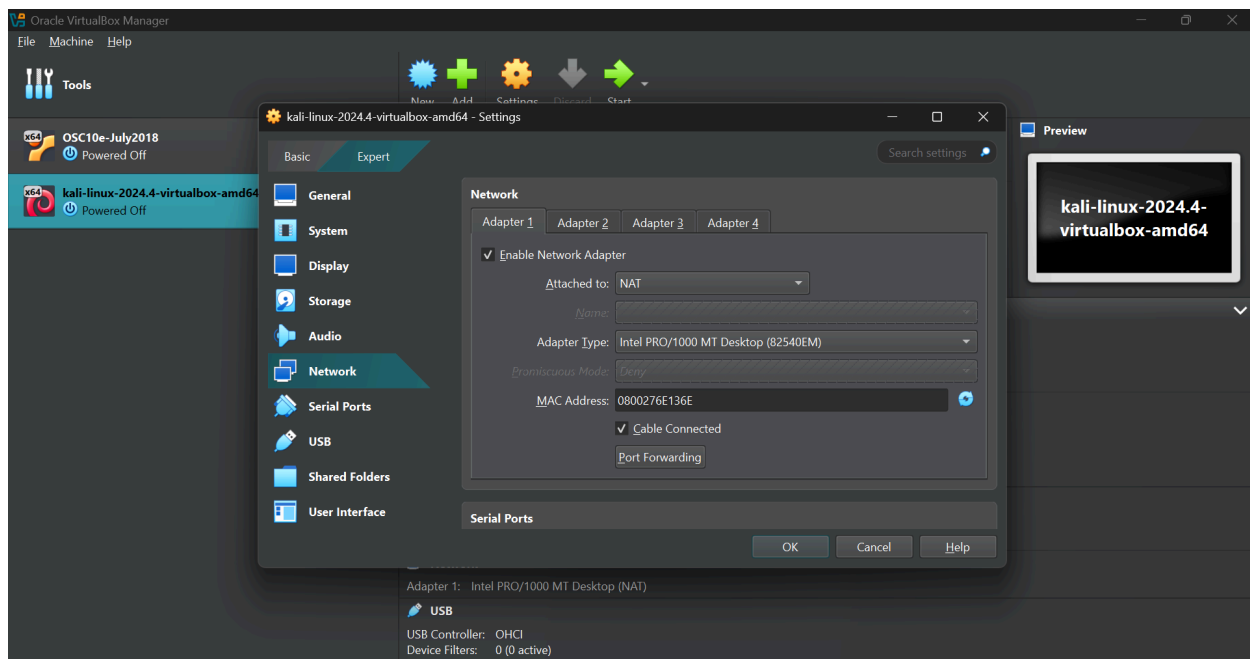
WireShark run as root while capturing on the 'any' adapter.

## Summary of Results

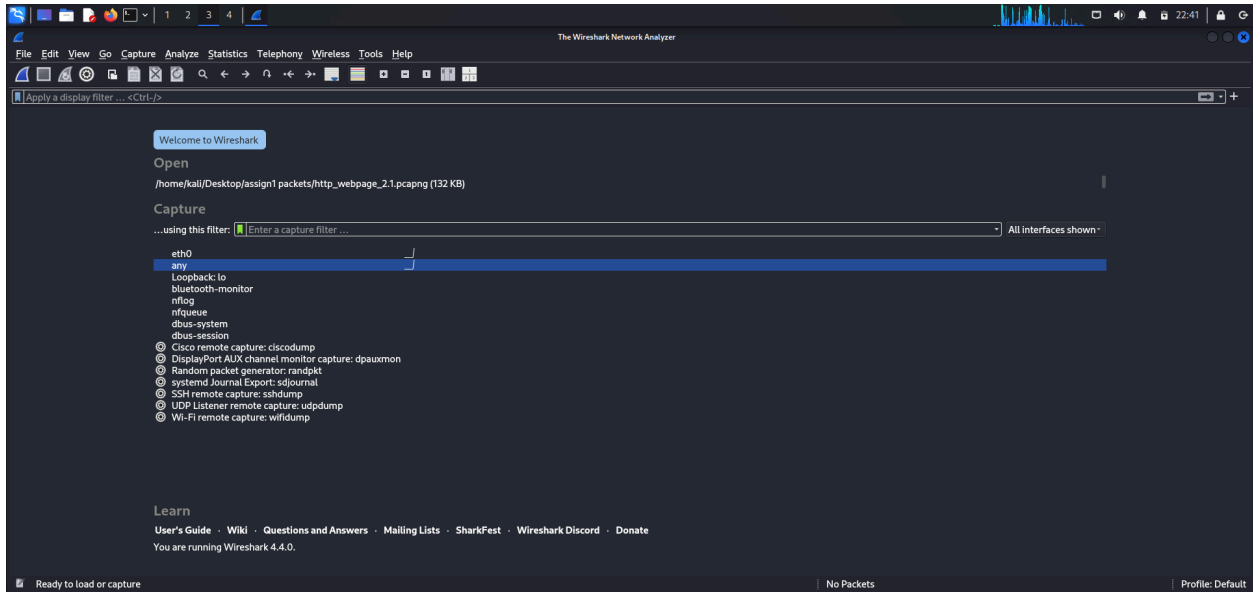
To begin, I downloaded Kali Linux, VirtualBox, as well as Oracle VirtualBox Virtual Machine Manager in order to get started. After clearing space on my laptop, I was able to install and run Oracle.



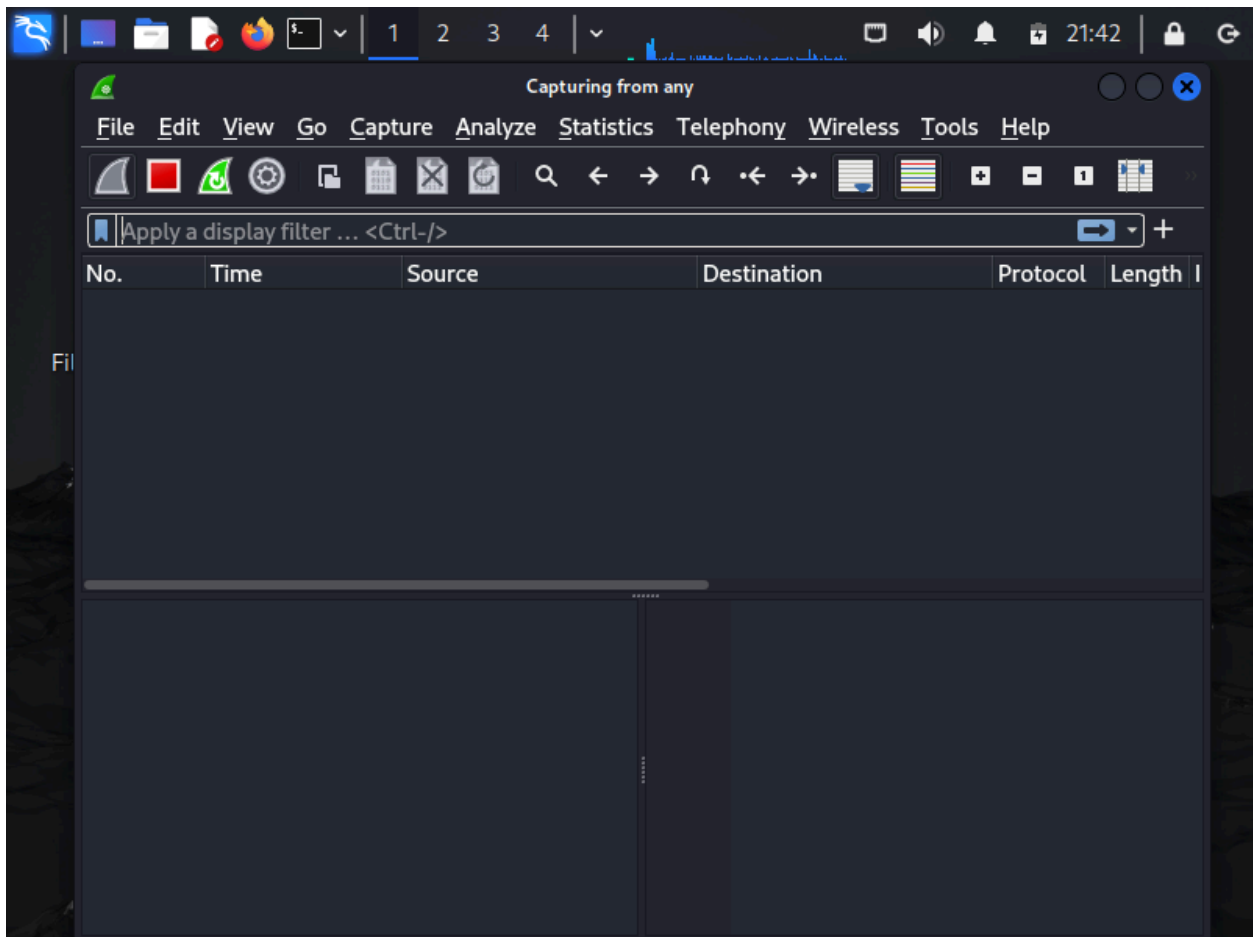
Once inside Oracle VirtualBox, I added the Kali Linux VirtualBox and adjusted the settings. The network was set to NAT, since communication was only needed between my host laptop and the virtual machine. All other settings were left at their defaults, and the virtual machine was started.



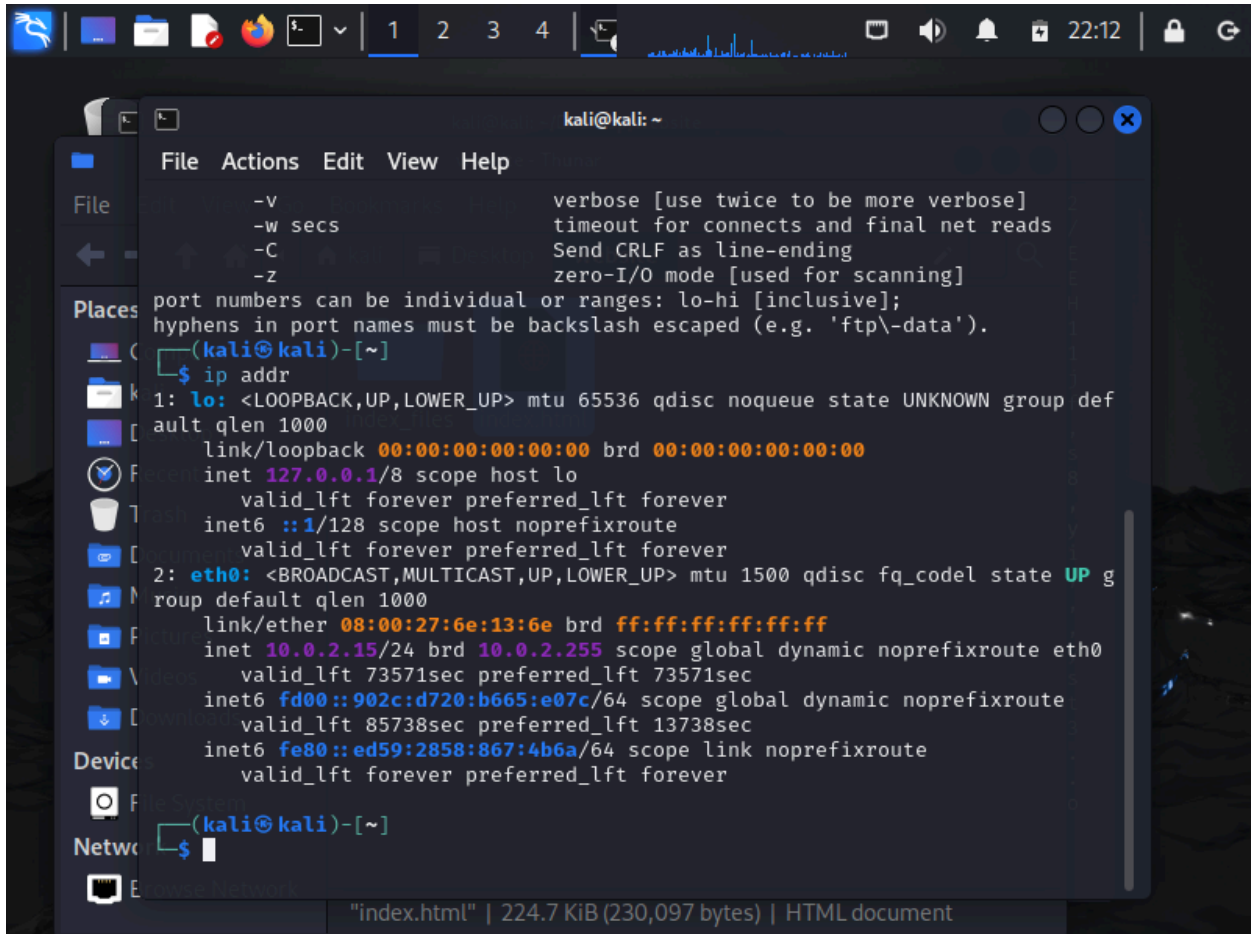
Upon startup Kali prompted me to enter the username and password, both of which are defaulted to 'kali.' After logging in, I launched the virtual machine and started up WireShark to be able to capture all packets as soon as network traffic was generated.



WireShark was set to capture from 'any,' also known as promiscuous mode. Once WireShark started listening no packets were captured as there was no traffic on the network yet.

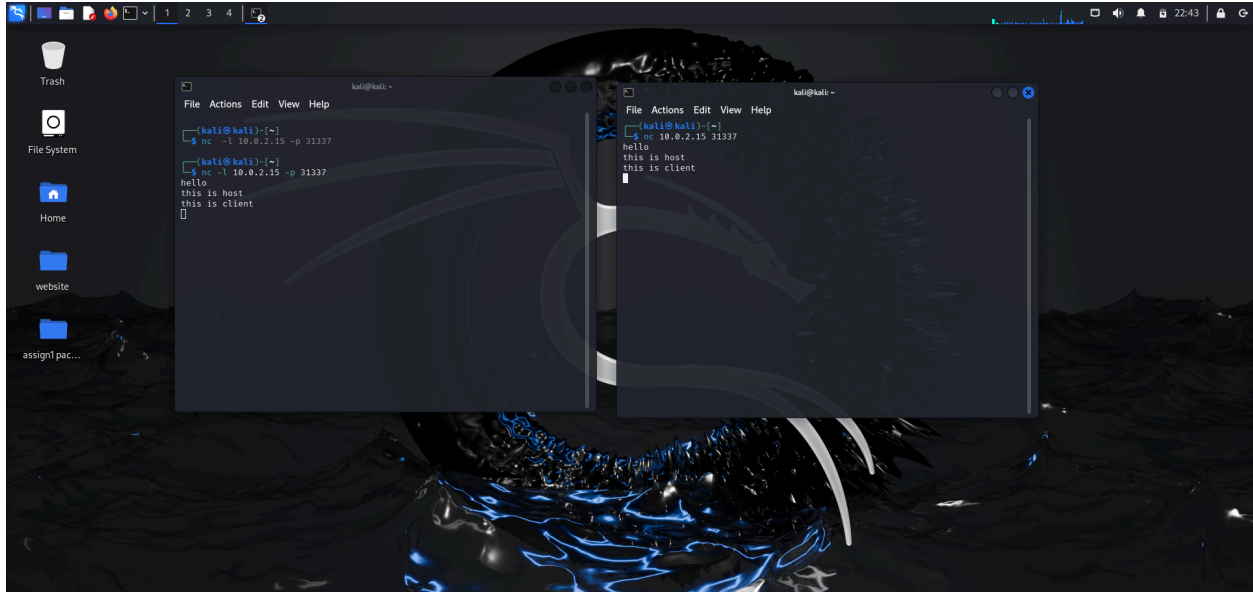


Before I could begin making unencrypted traffic, I needed to find my IP address. So I opened a terminal and used the command `ip addr`. From the feedback I received I was able to locate the ethernet connection “eth0” and locate my IP address listed after “inet”.

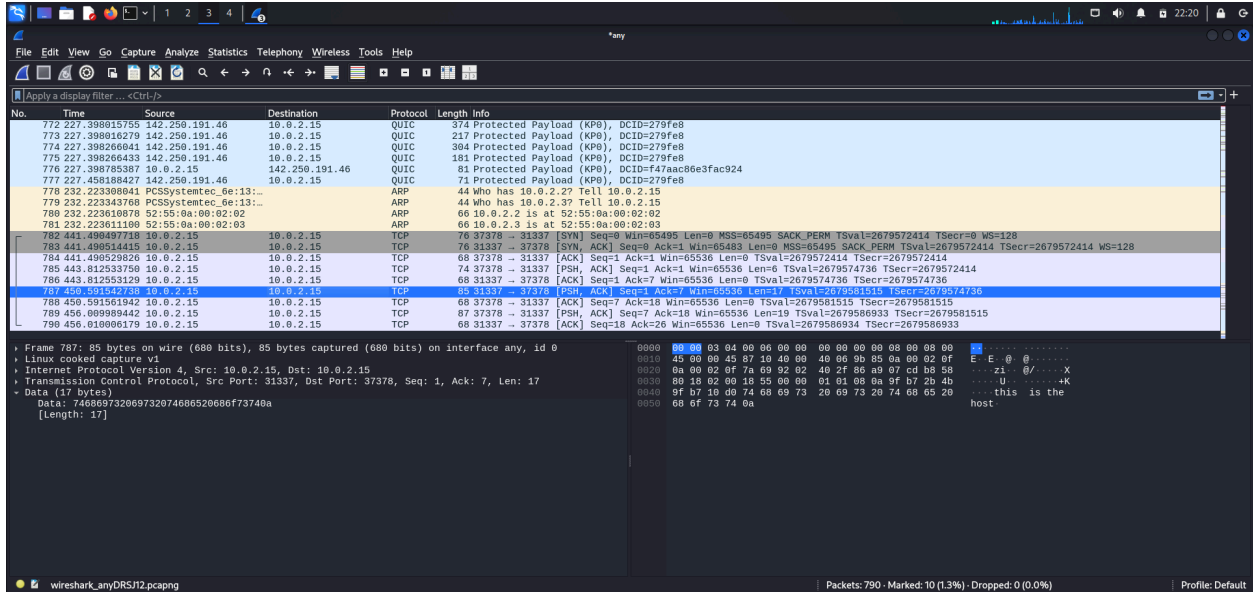


```
kali@kali: ~  
File Actions Edit View Help  
-v verbose [use twice to be more verbose]  
-w secs timeout for connects and final net reads  
-C Send CRLF as line-ending  
-z zero-I/O mode [used for scanning]  
port numbers can be individual or ranges: lo-hi [inclusive];  
hyphens in port names must be backslash escaped (e.g. 'ftp\ -data').  
(kali@kali)-[~]  
$ ip addr  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group def  
ault qlen 1000  
    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 noprefixroute  
        valid_lft forever preferred_lft forever  
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP g  
roup default qlen 1000  
    link/ether 08:00:27:6e:13:6e brd ff:ff:ff:ff:ff:ff  
    inet 10.0.2.15/24 brd 10.0.2.255 scope global dynamic noprefixroute eth0  
        valid_lft 73571sec preferred_lft 73571sec  
    inet6 fd00::902c:d720:b665:e07c/64 scope global dynamic noprefixroute  
        valid_lft 85738sec preferred_lft 13738sec  
    inet6 fe80::ed59:2858:867:4b6a/64 scope link noprefixroute  
        valid_lft forever preferred_lft forever  
(kali@kali)-[~]  
$
```

In order to create some unencrypted network traffic I used the Netcat listener command, which uses my newly found IP address, mentioned before to start up a server. Once I entered the command it started listening for messages but no clients were connected yet. So I opened a new Netcat browser and used the before mentioned sender command to connect to the server as a client. I sent a message from the client, received it on the server side, and then sent a message from the server and received it on the client side as well.

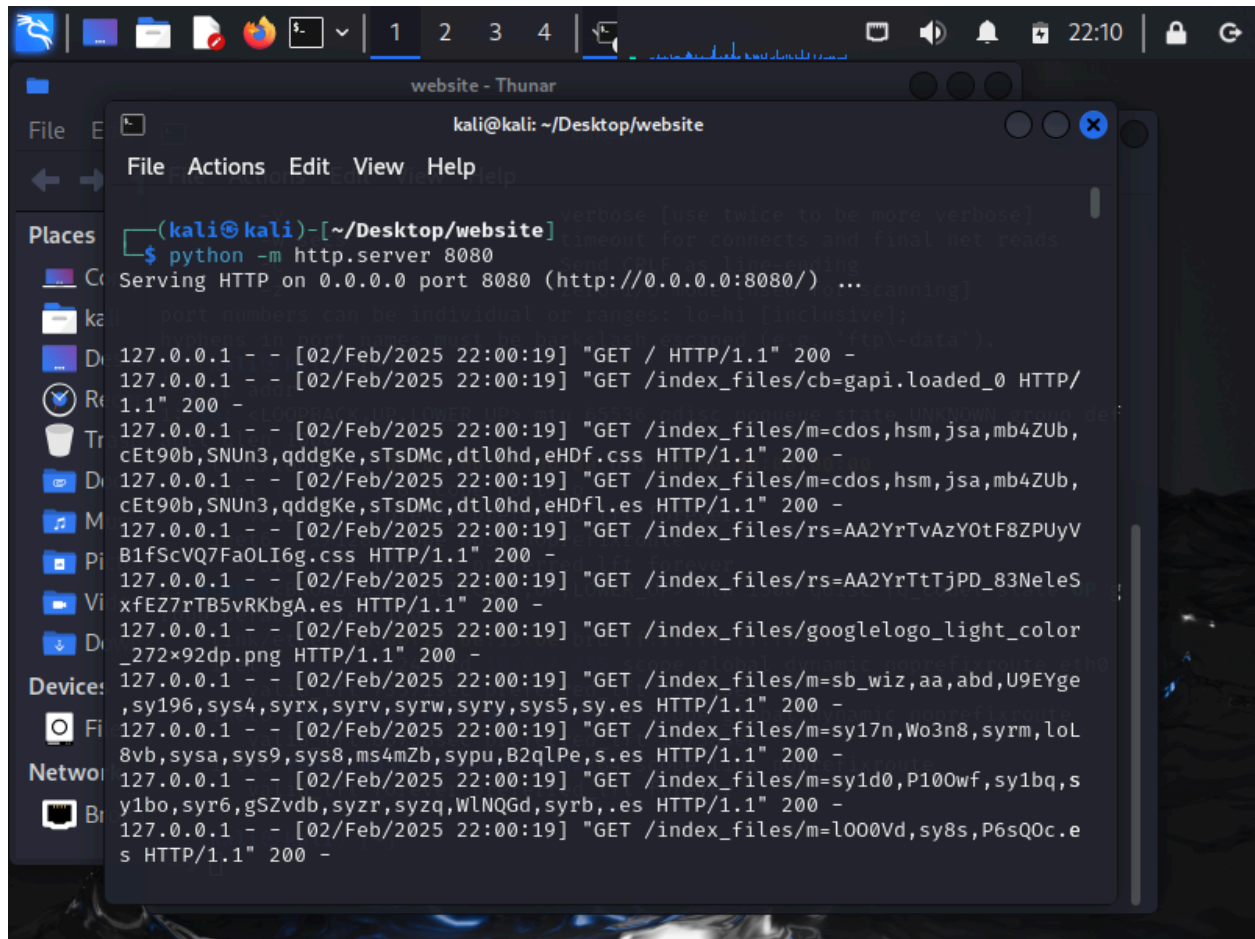


Wireshark captured all of these packets. From the start of the server to the connection of the client as well as the unencrypted messages sent between the two. Since these messages were unencrypted I was able to view not only the IP address destination, but also the port number, as well as the actual data of the messages as well. I stopped Wireshark's packet capture in order to export selected packets.



After this I switched methods and opened the Firefox browser. I went to <https://www.google.com/> and saved the webpage to a folder labeled "website" on my desktop, changing the name of the saved webpage to "index". Inside the "website" folder I right clicked the file inside and selected "open in terminal." Once inside the Python terminal I used the Python console command mentioned before to run this

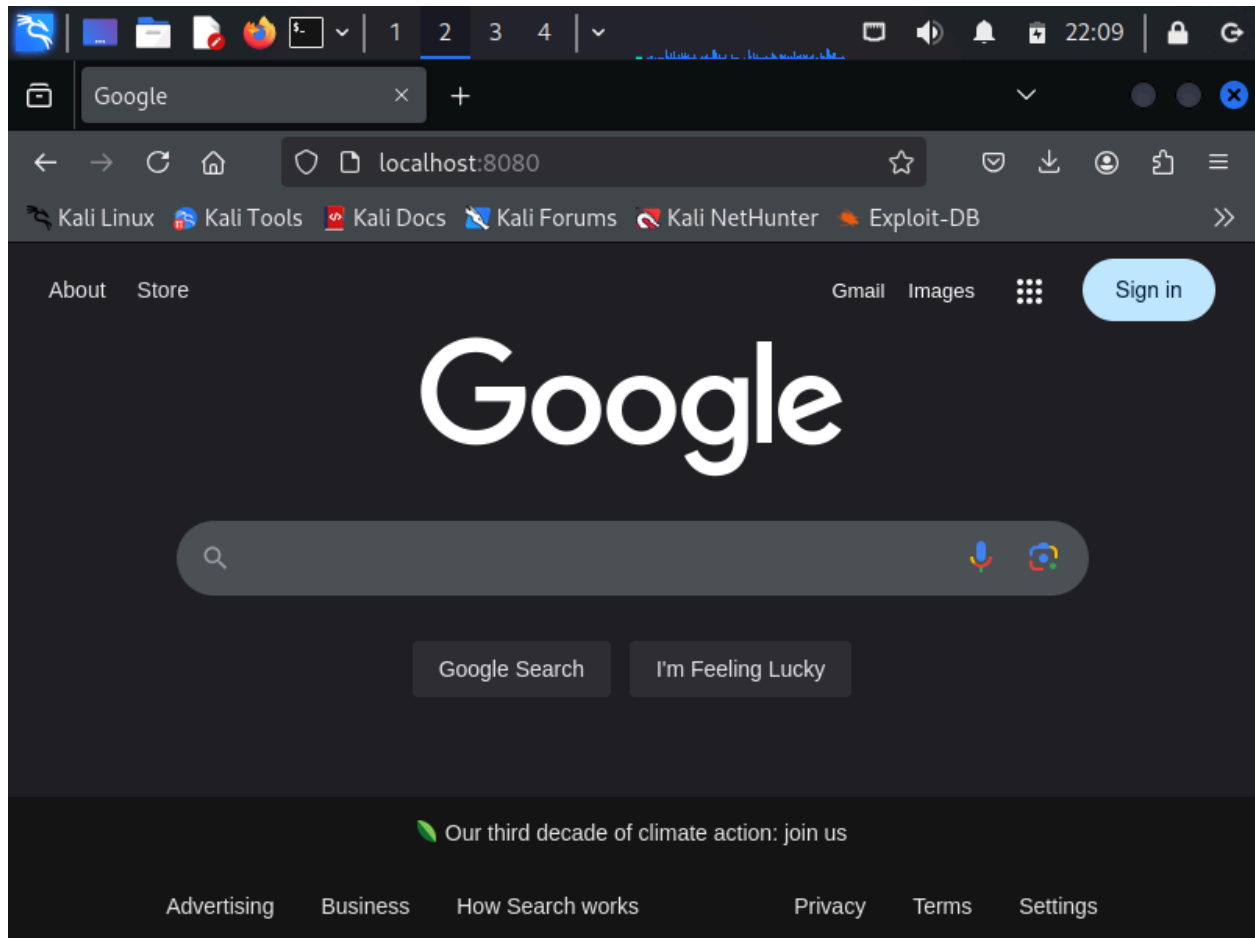
copycat webpage as an unencrypted http site using the port number 8080. After sending this command the terminal received a feedback message saying that the website was started.



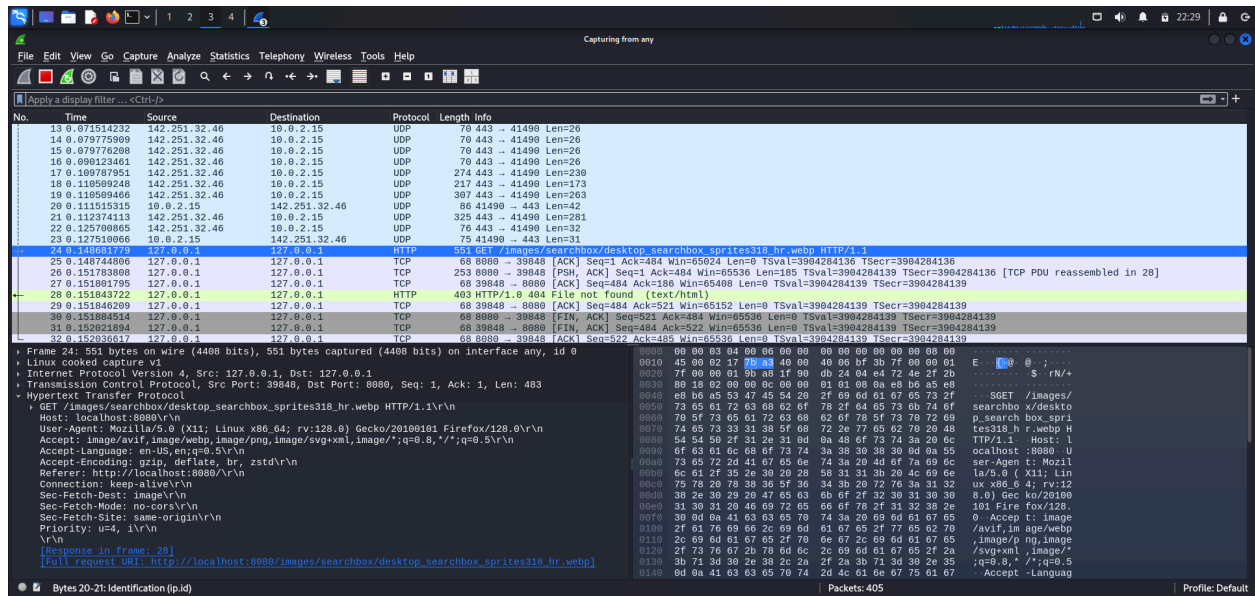
The screenshot shows a Kali Linux desktop environment. In the foreground, a terminal window titled 'kali@kali: ~/Desktop/website' is open. The terminal output shows the command `python -m http.server 8080` being executed, followed by the message 'Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...'. Below this, a series of HTTP requests from 127.0.0.1 are logged, all returning a 200 status code. The requests include paths for various files and directories, such as `/index_files/cb=gapi.loaded_0`, `/index_files/m=cdo`, `/index_files/m=sb_wiz`, and `/index_files/m=sy17n`. In the background, a web browser window titled 'website - Thunar' is visible, showing the local website being served by the terminal.

```
(kali@kali)-[~/Desktop/website]
$ python -m http.server 8080
Serving HTTP on 0.0.0.0 port 8080 (http://0.0.0.0:8080/) ...
127.0.0.1 - - [02/Feb/2025 22:00:19] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [02/Feb/2025 22:00:19] "GET /index_files/cb=gapi.loaded_0 HTTP/1.1" 200 -
127.0.0.1 - - [02/Feb/2025 22:00:19] "GET /index_files/m=cdo, hsm, jsa, mb4ZUb, cEt90b, SNU3, qddgKe, sTsDMc, dtl0hd, eHDF.css HTTP/1.1" 200 -
127.0.0.1 - - [02/Feb/2025 22:00:19] "GET /index_files/m=cdo, hsm, jsa, mb4ZUb, cEt90b, SNU3, qddgKe, sTsDMc, dtl0hd, eHDF.es HTTP/1.1" 200 -
127.0.0.1 - - [02/Feb/2025 22:00:19] "GET /index_files/rs=AA2YrTvAzY0tF8ZPUyVB1fScVQ7Fa0LI6g.css HTTP/1.1" 200 -
127.0.0.1 - - [02/Feb/2025 22:00:19] "GET /index_files/rs=AA2YrTtTjPD_83NeleSxfEZ7rTB5vRKbgA.es HTTP/1.1" 200 -
127.0.0.1 - - [02/Feb/2025 22:00:19] "GET /index_files/googlelogo_light_color_272x92dp.png HTTP/1.1" 200 -
127.0.0.1 - - [02/Feb/2025 22:00:19] "GET /index_files/m=sb_wiz, aa, abd, U9EYge, sy196, sys4, syrx, syrv, syrw, syry, sys5, sy.es HTTP/1.1" 200 -
127.0.0.1 - - [02/Feb/2025 22:00:19] "GET /index_files/m=sy17n, Wo3n8, syrm, lol8vb, sysa, sys9, sys8, ms4mZb, sypu, B2qlPe, s.es HTTP/1.1" 200 -
127.0.0.1 - - [02/Feb/2025 22:00:19] "GET /index_files/m=sy1d0, P10Owf, sy1bq, sy1bo, syr6, gSZvdb, syzr, syzq, WlNQGd, syrb, .es HTTP/1.1" 200 -
127.0.0.1 - - [02/Feb/2025 22:00:19] "GET /index_files/m=l000Vd, sy8s, P6sQ0c.es HTTP/1.1" 200 -
```

With the site up I reopened the Firefox browser and typed the url `http://localhost:8080` into the search bar. Since this server was now up and running I was brought to a remake of google.com homepage, except this connection was run by a local host and also unencrypted.



Similar to before with the Netcode exchanges, all information passed between the server (python terminal) and the client (the browser) such as get requests and urls, were caught by WireShark. Additionally, since the webpage was unencrypted all the data of these requests are visible in the packet captures.



## Conclusion

From this project I have found Wireshark to be an application that connects to the network, capturing, or making copies of, packets that are sent and received through the shared network. Wireshark can be configured to only capture packets transmitted to or from specific devices on the network, additionally packets captured can be filtered through many means, for example their port number. Wireshark displays all captured packets, numbering and timestamping them, as well as displaying all the information within the packets themselves, including the source, destination, protocol, length, as well as the info. In the case that this information is encrypted the data is not intelligible. However, if these packets that are captured are not encrypted, the data being transmitted can easily be read while inside Wireshark. Additionally, Wireshark allows the user to export any captured packets. My biggest takeaway from this is the fact that no connection to a network is entirely private, all information must be sent through various networks to reach its destination. And any and all users can have access to packets that are being transmitted, this is why encryption is so important, in order to protect your own information and data from attackers attempting to learn, steal, or tamper with your data being transmitted or received it must be encrypted. However, even when it is encrypted, attackers may still see the packets, but the contents will be safe.