Naima Mumin
https://www.linkedin.com/in/naima-mumin-a48750270/

ARP Spoofing Exploration:Wireshark Analysis With Bettercap

Introduction:

- This project explores Bettercap and Wireshark to understand network traffic interception and analysis.

Understanding ARP Spoofing:

- ARP spoofing is also known as Man In The Middle attack, involves tricking devices on a network into sending data to the wrong place, and today I'm exploring this for educational purposes within my own network.

Requirements:

- Operating System:
  - Linux(host) and Ubuntu(target)
- Tools Needed:
  - Bettercap
  - Wireshark

Understanding the Tools:

- Bettercap is a powerful tool that is used for testing and attacking networks to find security weaknesses
- Wireshark is a tool used to look at and understand the data traveling through a network to fix problems or undersntad how it works.

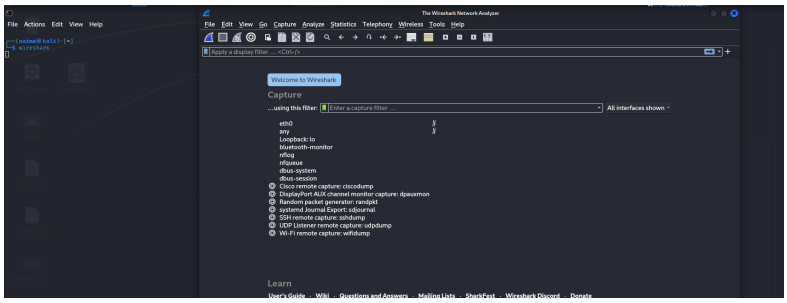
How to install Bettercap in Linux?

- **sudo apt-get install bettercap**

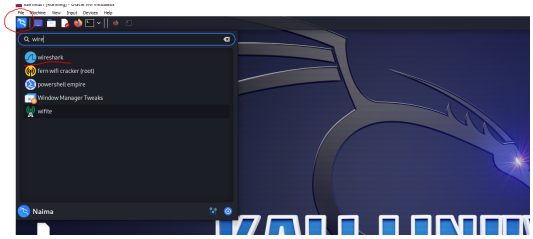
How do I check what version do I have?

- **sudo bettercap - -version**

Wireshark is already included in Linux, so there's no need for a separate installation.If I just type "Wireshark" in the terminal, it immediately launches the application

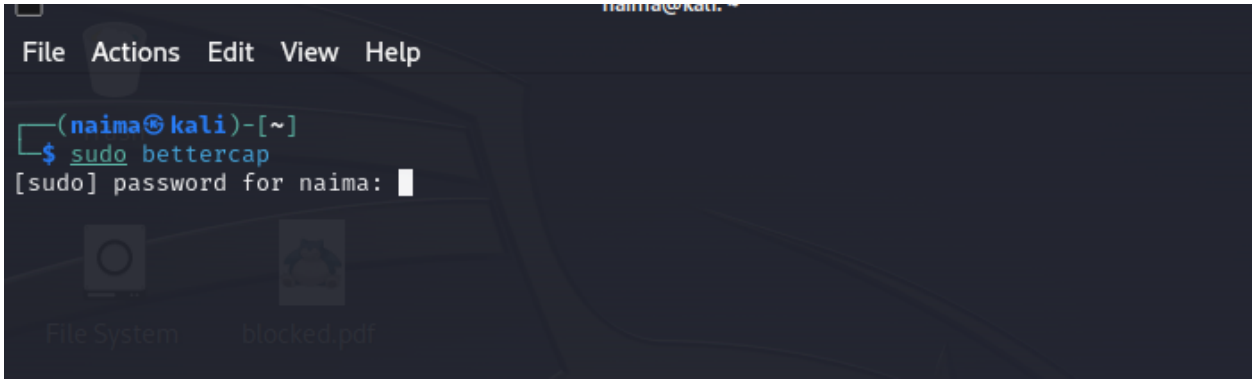


The other way that I can run Wireshark is clicking "linux" image and searching "Wireshark" in there like this picture bellow

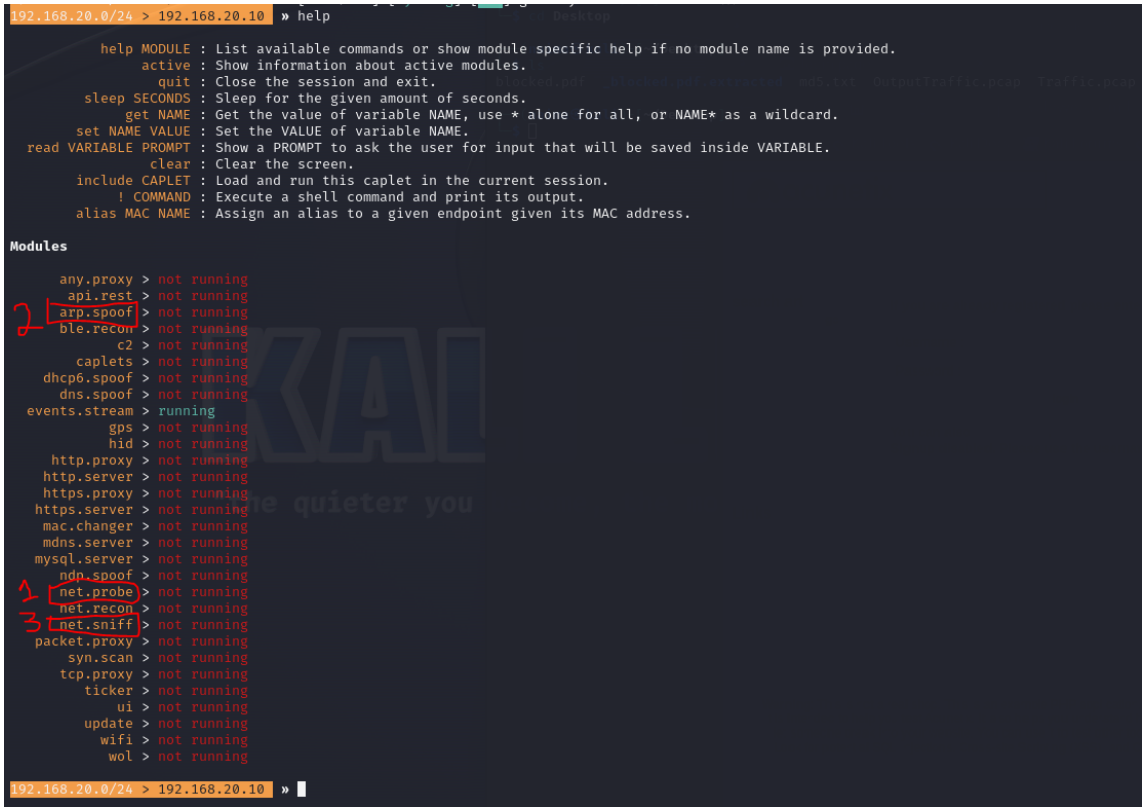


Let's get start it!
First thing I need is, to open bettercap

Command: **sudo bettercap** then I Enter my linux password



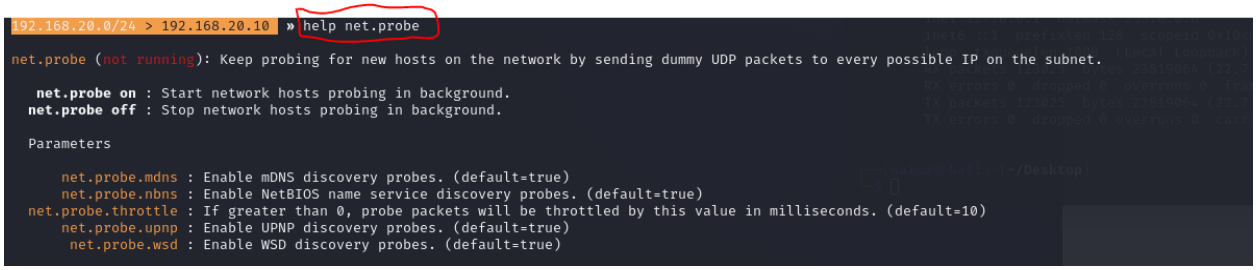
1. Understand the command before using it
   - It's really important to understand the commands I need before using it. I'll start by running "help" which shows all the available commands. Moreover, using "help" before specific command provides detailed information about what it does.



For example:

- Let use "help net.probe" and see what it displays

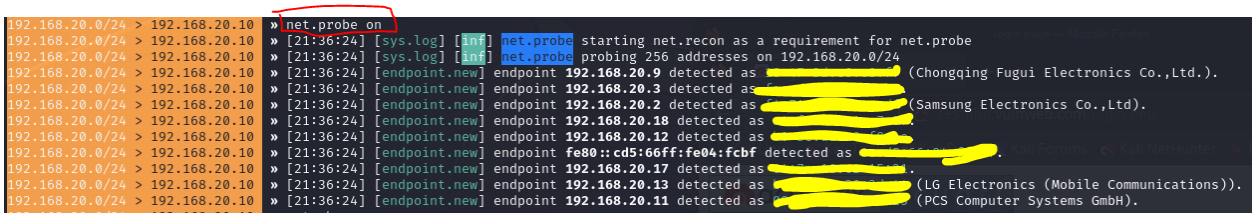
- As we can see here, it displayed what the net.probe command does and how you can I use it in multiple way
- The option I'm going to use is "net.probe on" to start the host



2. Start the network by using the following command

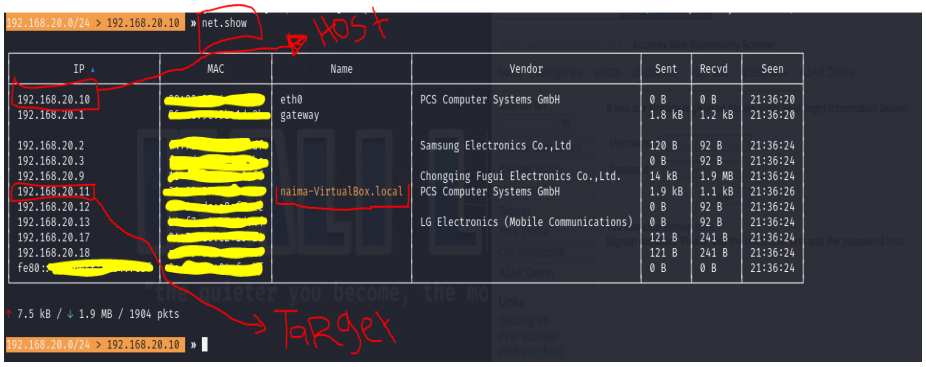
Command: **net.probe on**

- This command allows me to uncover every device connected to my network, providing essential details such as MAC addresses, IP addresses, names, etc.



3. Determine your Ip addresses

- Execute the command "net.show" to reveal stored network information
- Once I execute the command, I'm able to identify both the host and target IP addresses. In the image below, you can see that my Linux machine is assigned the Ip address 192.168.20.10, serving as the host, while my Ubuntu machine is identified with the IP address 192.168.20.11, acting as the target in this project.



4. Start the ARP spoofing

Command:

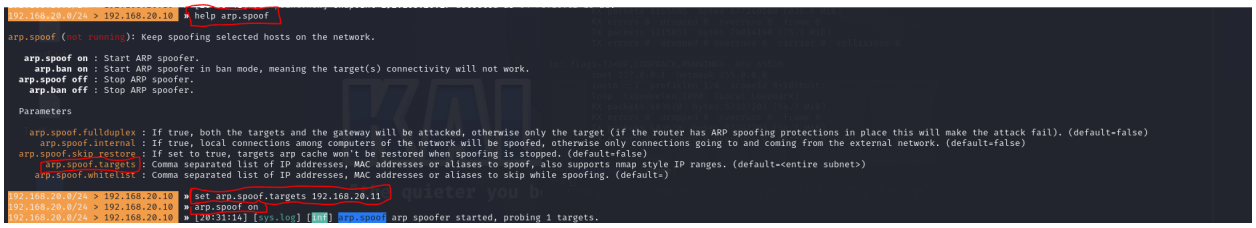
- **help arp.spoof**
  - I'm going to use this because I want to see all the options that is available for me that I can use it before i do the arp.spoof
- **Arp.spoof.targets <Ip address>**

- Here, I'm saying start the arp spoof with this specific target which I will be using my ubuntu IP address as my target. Here is an example of how I would perform this.

Example: **set arp.spoof.targets 192.168.20.11**

-

- **arp.spoof on**
  - After that, I will start the arp spoof by using "arp.spoof on"



To get ready for ARP spoofing, I'll set up a folder to store all the data I collect using Bettercap. Later on, I can use Wireshark to review the data I have gathered during the sniffing process

5. Create a Pcap file to analyze the information later on wireshark

- Open new linux terminal and use the following

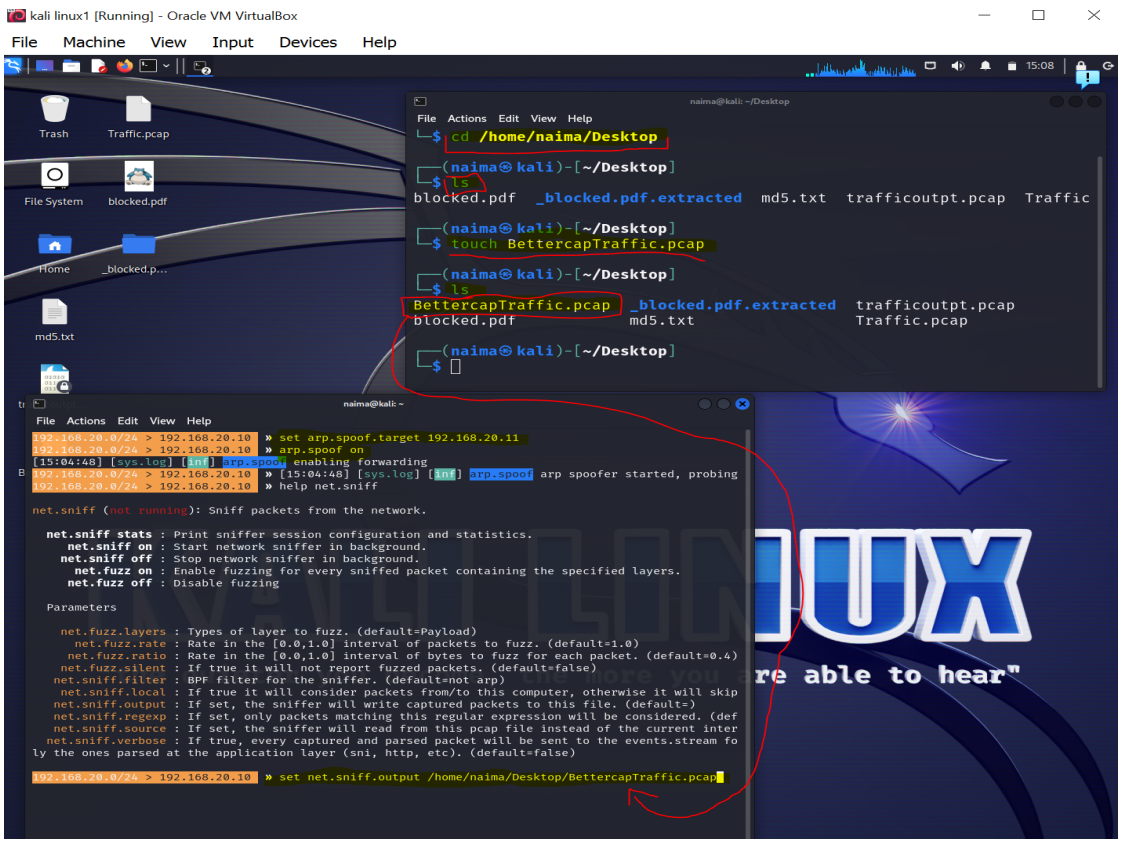
Command:

- **cd ~**
- **cd Desktop**
- **touch <filename.pcap>** example: **touch BettercapTraffic.pcap**
- **ls**

6. Start the ARP spoofing

Commands:

- **set arp.spoof.target 192.168.20.11**
  - I specified the target IP address for ARP spoofing as "192.168.20.11" which this IP address is my Linux IP address
- **arp.spoof on**
  - I activated ARP spoofing to begin the spoofing process
- **help net.sniff**
  - Here, I'm requesting to see all the options available in net.sniff command that I can use
- **set net.sniff.output /home/naima/Desktop/BettercapTraffic.pcap**
  - I configured the bettercap to save captured network data into "BettercapTraffic.pcap" file on my desktop

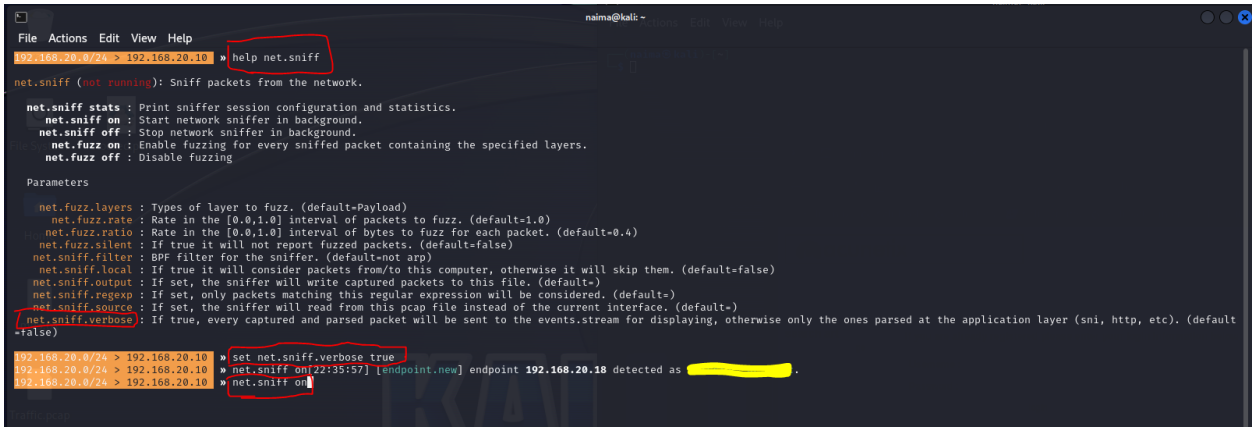


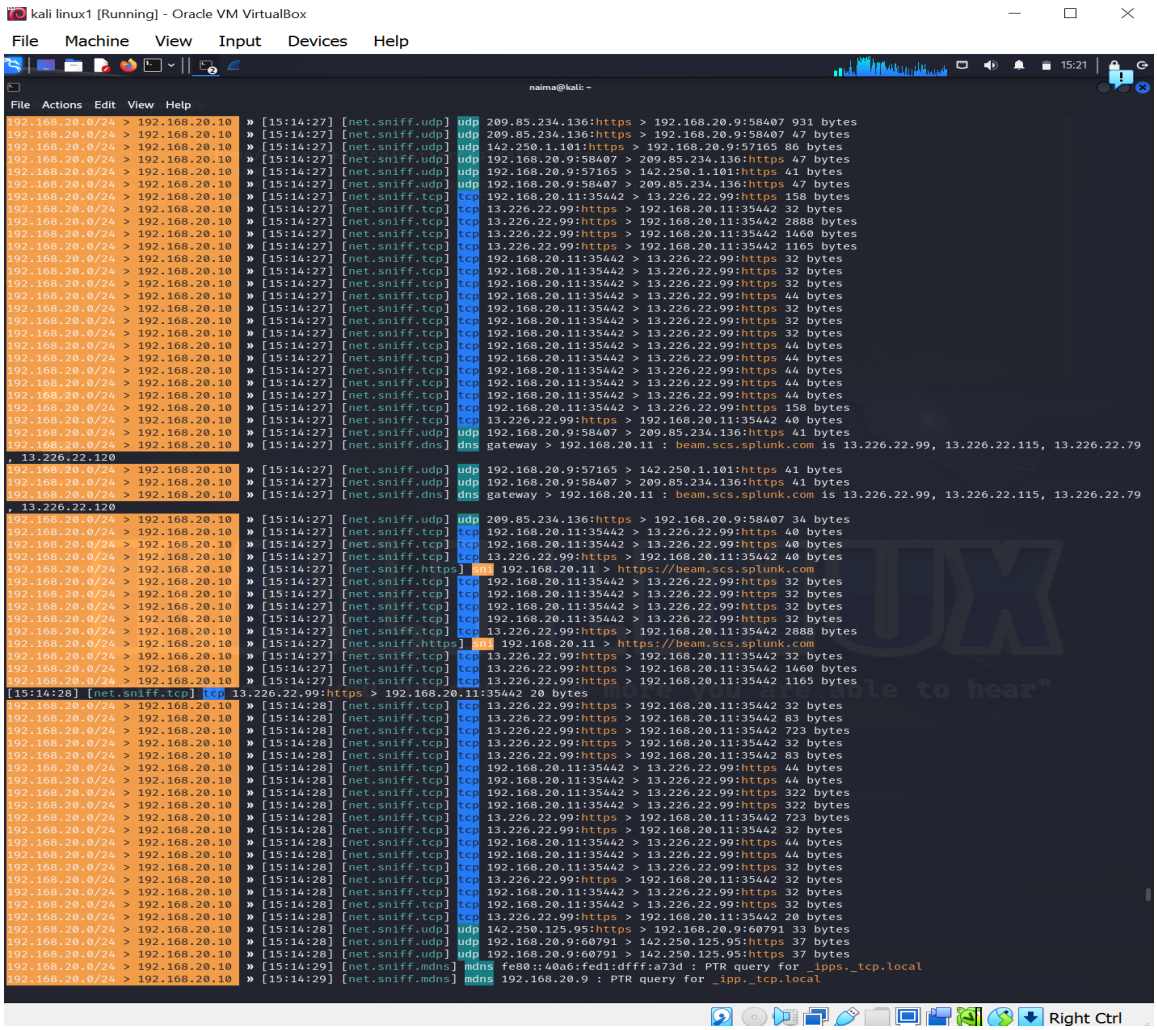
7. Start the sniffing mode

Commands:

- **help net.sniff**
  - Here, I want to see how many options are available within net.sniff. This will helps me to figure it out what net.sniff command I should use next
- **net.sniff.verbose**
  - Using “net.sniff.verbose” allows me gather detailed network information and sends all captured data to the file “BettercapTraffic.pcap” that I created earlier
- **set net.sniff.verbose.true**
  - I used that command above because I want to see more details about what’s happening on the network. It helps me understand everything going on between devices better
- **Net.sniff on**
  - This allows bettercap to start sniffing all the network traffic, including websites visited and other data transmitted

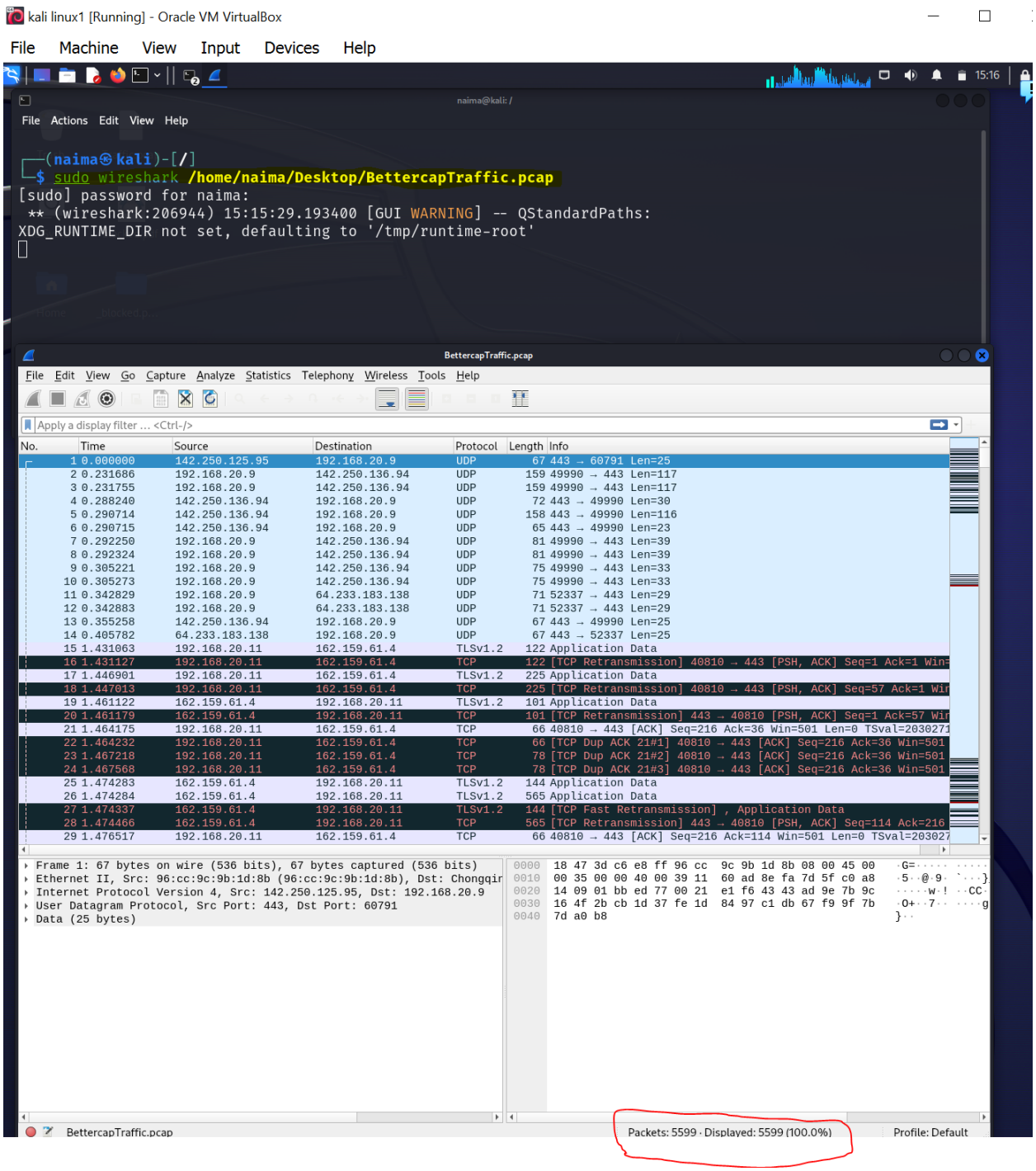
Image that includes all the commands I've used:





8. Let's analyze this information in the wireshark

- open wireshark in a new terminal using **sudo wireshark /home/naima/BettercapTraffic.pcap**



- In this pcap, Wireshark grabbed around 5599 packets. Although more packets could have been captured,I stopped because I think I have enough information to analyze it

Which protocols has Wireshark captured?

- UDP TCP, ICMP, TLS, MDNS, HTTP

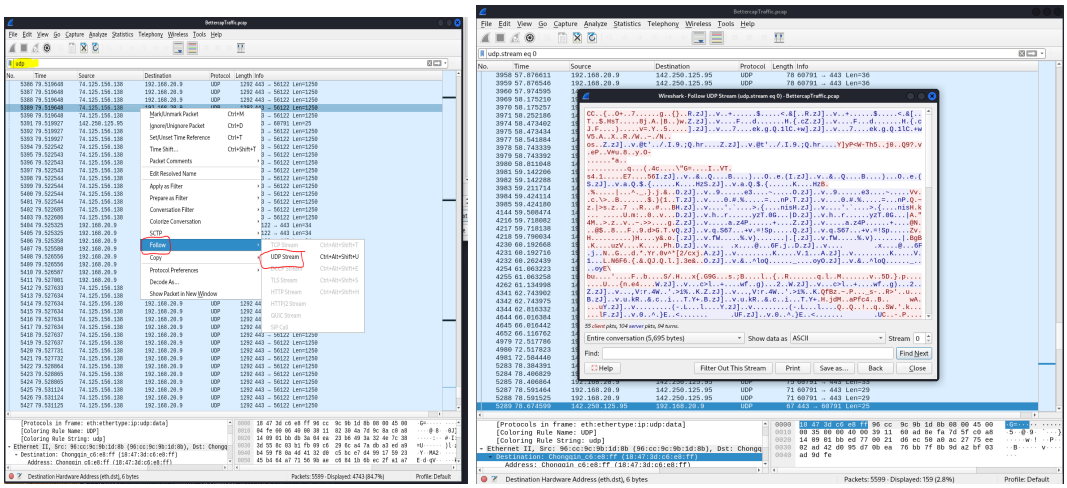
Note: I'm only going to look filter UDP, TCP and HTTP because I think these are the important protocols that I can look at it

9. Filter for each Protocol and find what information that Wireshark capture

- UDP and TCP Filter

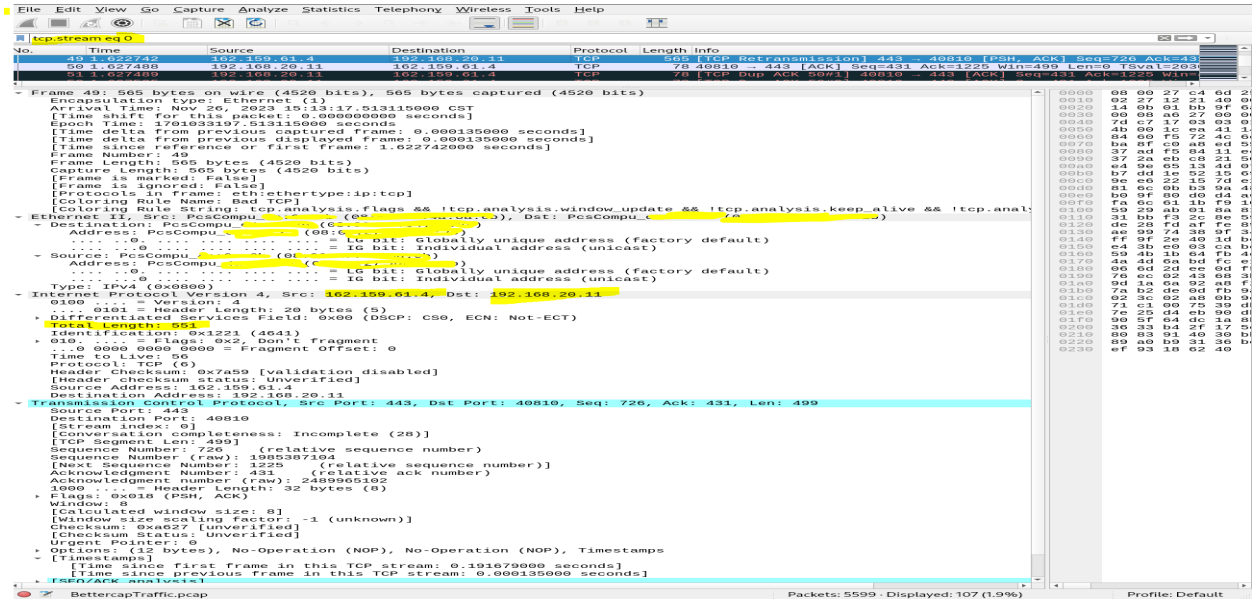
UDP

- While analyzing the UDP and TCP protocol in Wireshark, I aimed to explore the contained information. However, I encountered encrypted data instead.

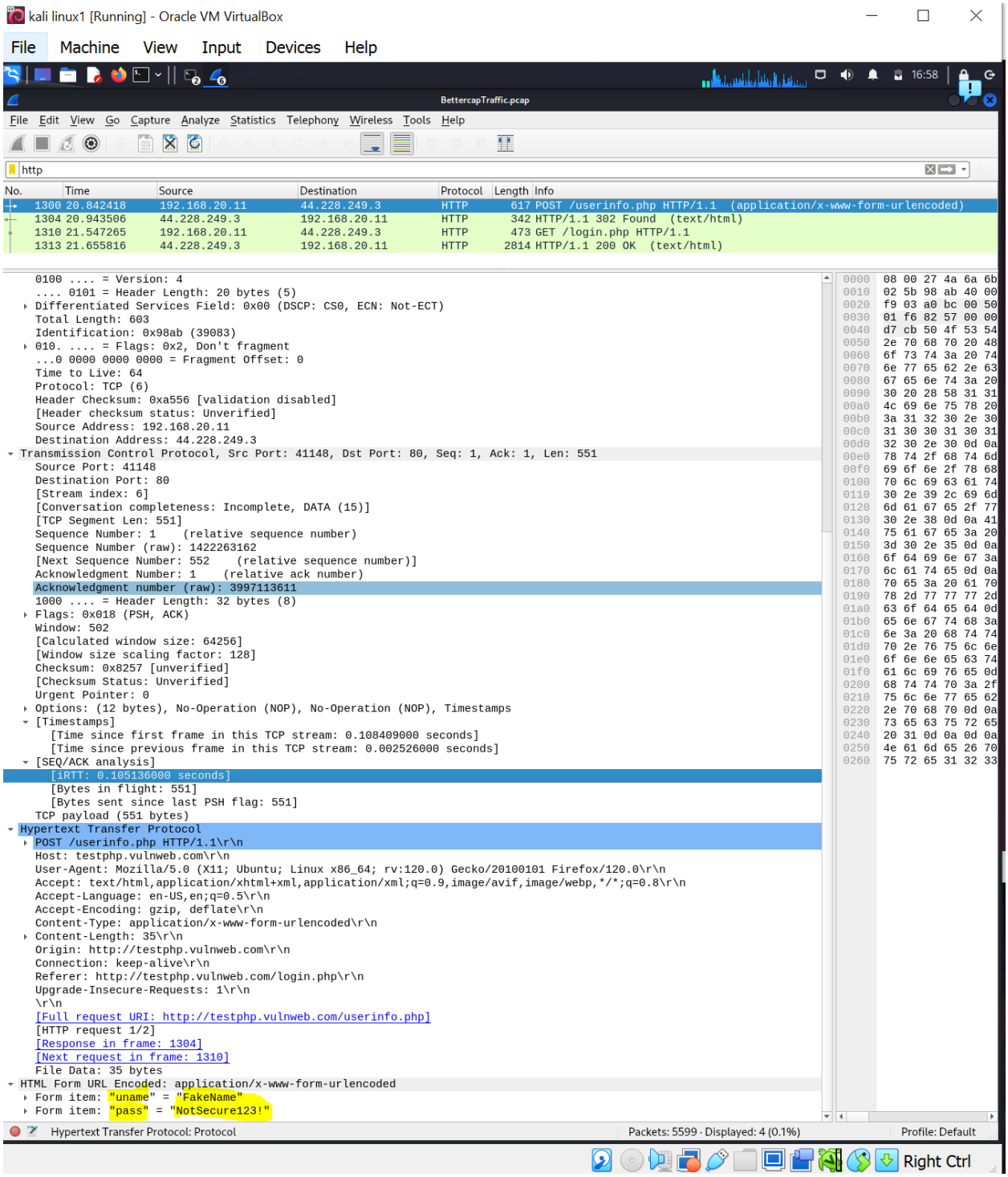


TCP:

- In exploring the TCP filter, I discovered information available for examination. Despite encrypted sections in both UDP and TCP data, I determined the total length to be 551 by examining the details within the TCP packets. Additionally, I observed the source IP(162.159.61.4) and destination IP(192.168.20.11) addresses.Despite the encryption, there's still a lot to explore in the available data.



- Now Let's jump ahead and filter the HTTP Protocol. that it's unencrypted, so we'll get more info for analysis



- During an ARP Spoofing experiment, I visited a website with known vulnerabilities to check if the information I input could be captured via Wireshark. On that HTTP-based site, I entered a username and password: “FakeName” for the username and “NotSecure123!” for the password. I observed that this login data appeared within the captured data under the section labeled “HTML Form URL Encoded:

Application/x-www-form-urlencoded," confirming the visibility of the entered username and password withing the captured packets.

What does this tells?

- Browsing HTTP websites exposes your data, making it susceptible to interception by cyber attackers.

Summary:

- In this project, I got hands-on experience with Bettercap and Wireshark, learning how ARP Spoofing affects networks and the importance of strong security to protect against such threats.