Brooklyn Dressel

Assignment 6

# **Abstract**

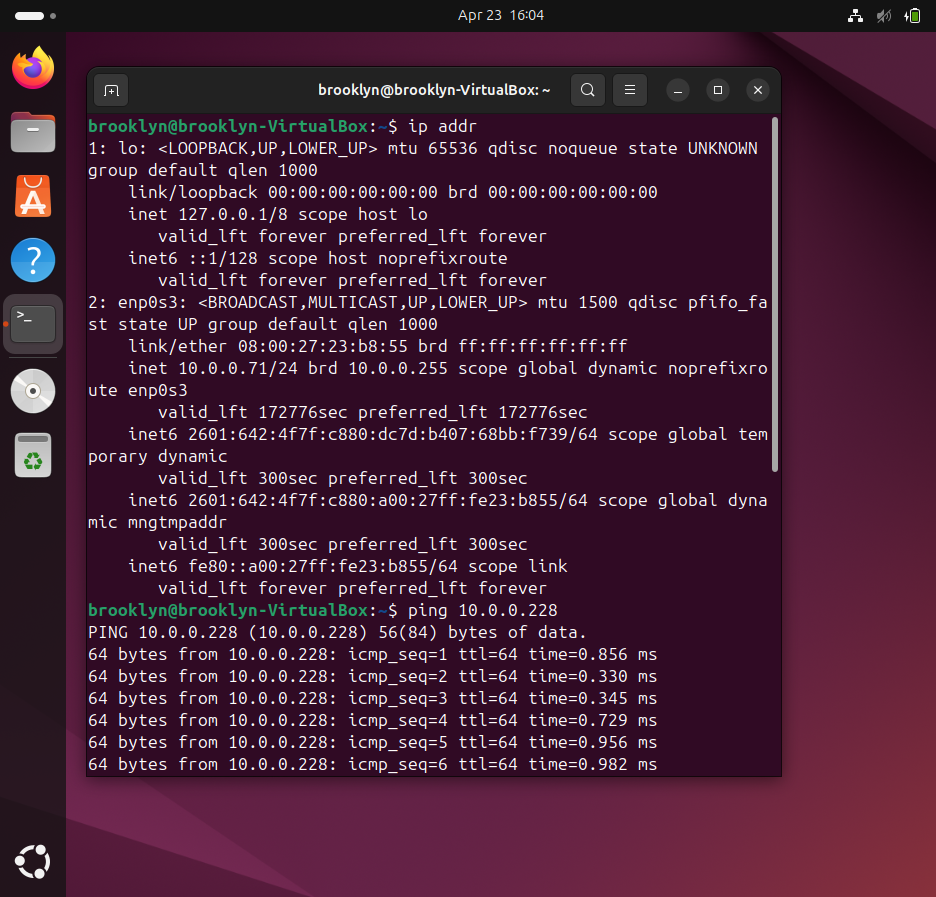
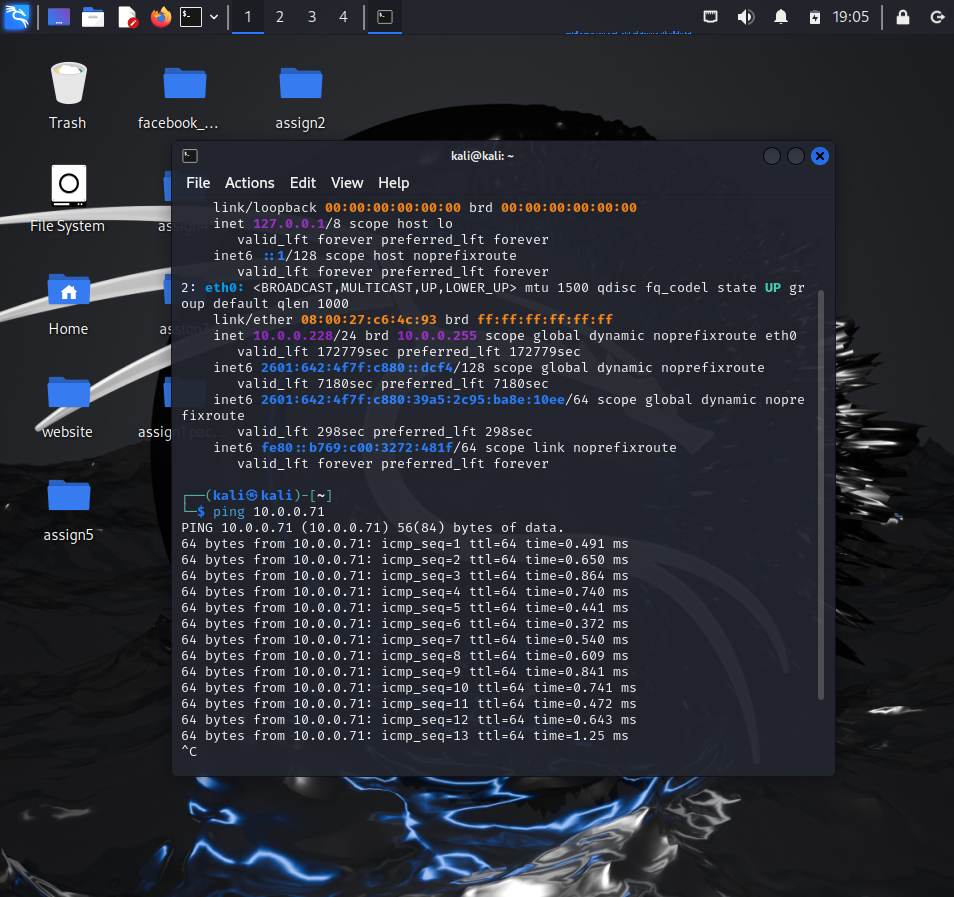
This assignment sets up a honeypot server in a virtual machine by disabling the firewall. We then proceed to attack this from a separate virtual machine over a network using five different network services. These were HTTP, FTP, SSH, SMTP, and Telnet traffic. All communications and attackers were recorded in logs by the honeypot server. In addition to this we wrote two python scripts to create two more, more sophisticated imitations of network services, HTTP on port 8080 and an echo server on port 9009. From these we can tell the breadth and depth of information a honey pot server is able to record, exposing attackers ip addresses and methods in plaintext.

# **Introduction**

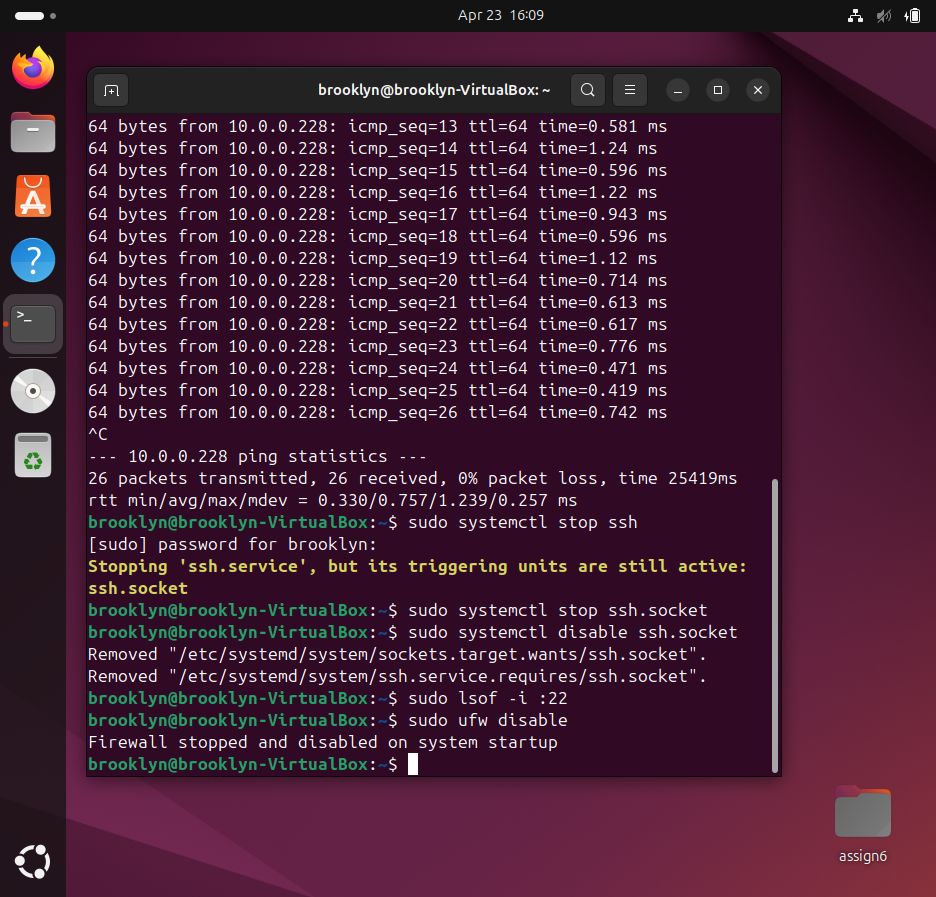
This assignment will use an Ubuntu virtual machine with a disabled firewall to play the part of the honeypot. A Kali Linux virtual machine will play the role of the attacker. Netcat listeners will be set up on Ubuntu and log all activities on five separate types of traffic (ports). WireShark will run on the Kali machine and capture all network traffic to be analyzed after the attacks. Kali will connect to the Ubuntu machine using a Firefox web browser to generate HTTP traffic (port 80), FTP traffic (port 21), a SSH connection (port 22), SMTP traffic (port 25), and finally Telnet traffic (port 23). Additionally, two python scripts were used to simulate A list of all commands used during this assignment is attached separately.

# **Summary of Results**

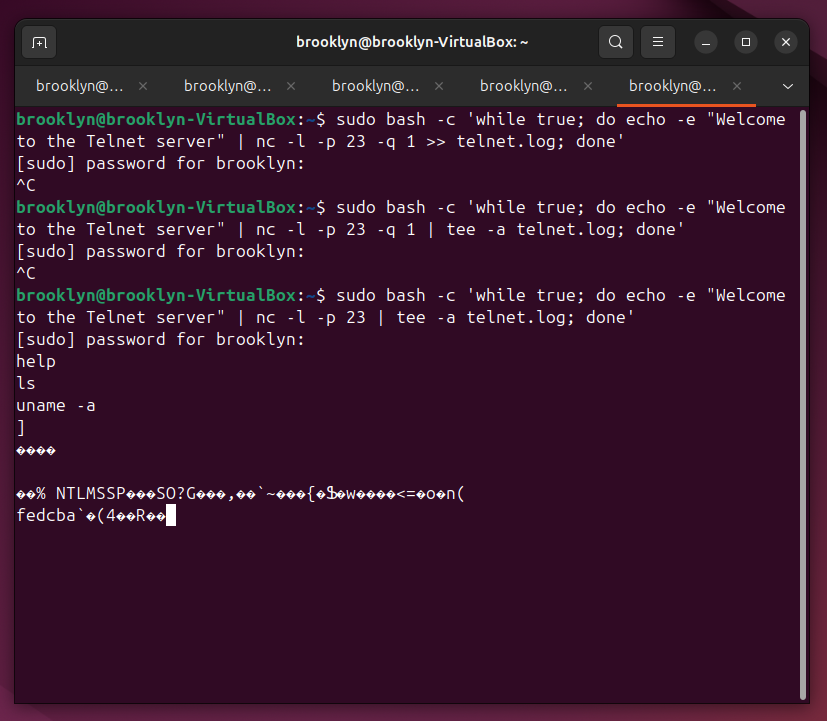
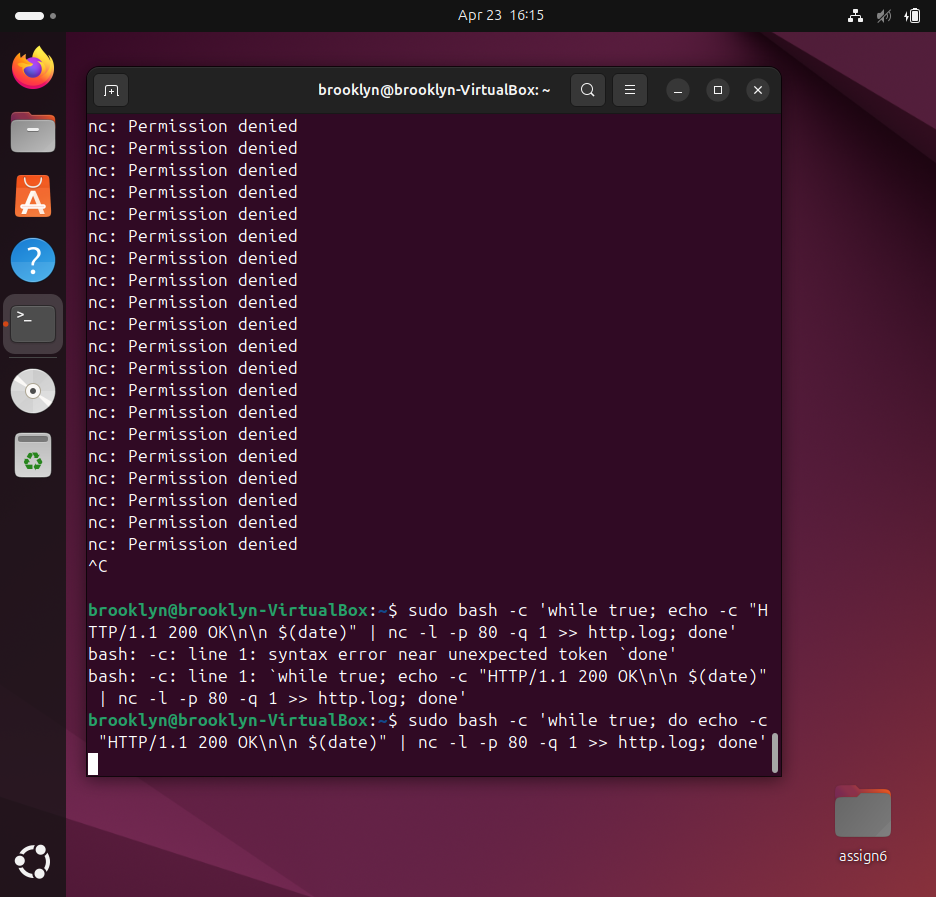
The initial step in this assignment was to boot up a Kali and an Ubuntu virtual machine, configured to a Bridged Network to allow for communication between the virtual machines. Upon startup, command 1 to find the respective ip addresses of these virtual machines, I then ran commands 2 and 3 and to confirm the two could communicate. I confirmed the pings were successful and terminated the pings.



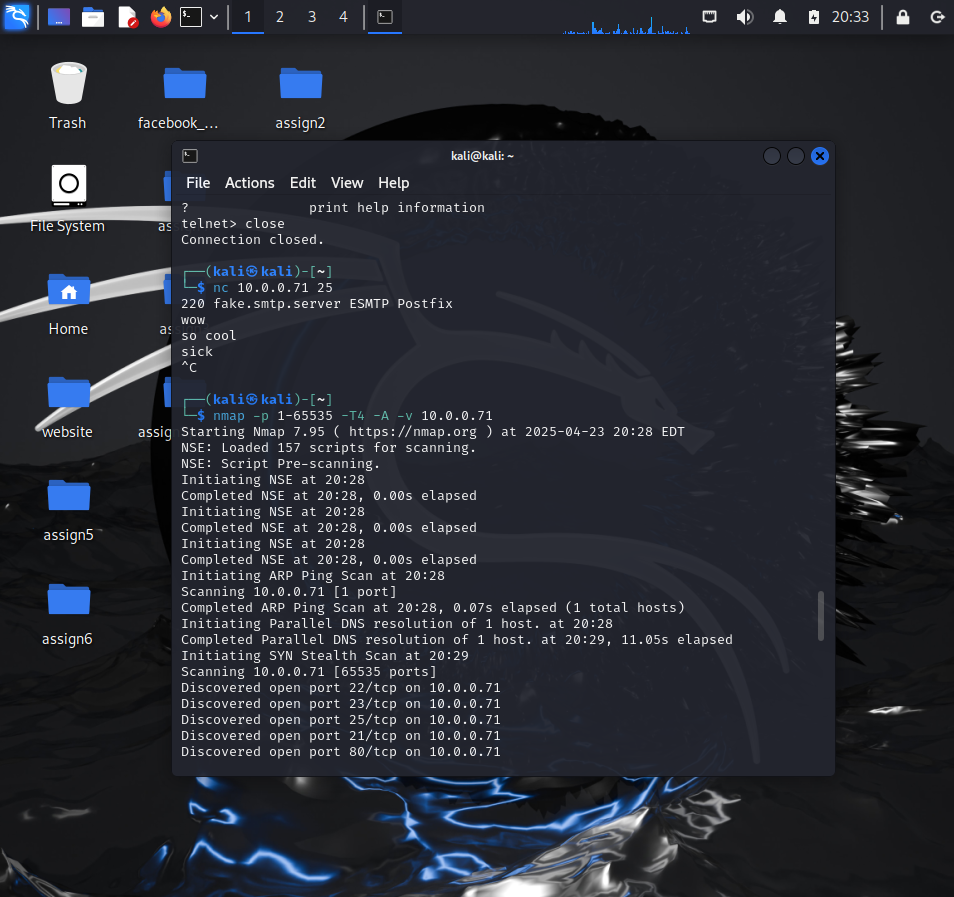
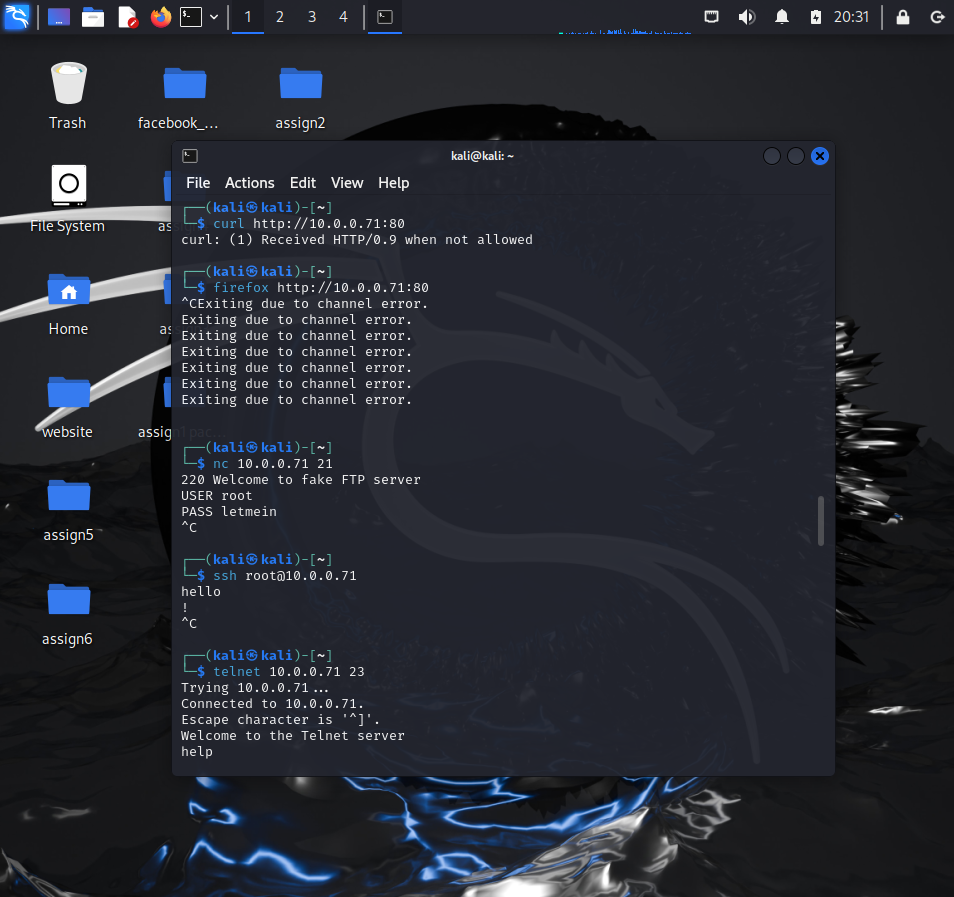
In Ubuntu, I ran command 4 to stop any previous running ssh services, to ensure a clean slate. I had to run commands 5 and 6 to close the socket service as well, and confirmed that all services were terminated using command 7. Finally, to make this a true honeypot I ran command 8 in order to disable my firewall and make the Ubuntu machine vulnerable to attacks and foreign connections. I confirmed this using command 9.



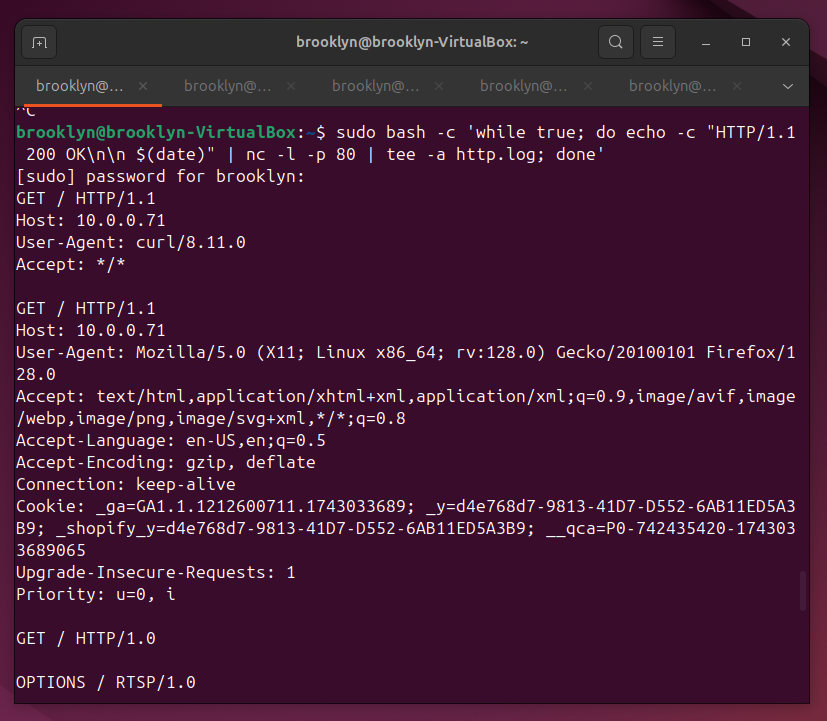
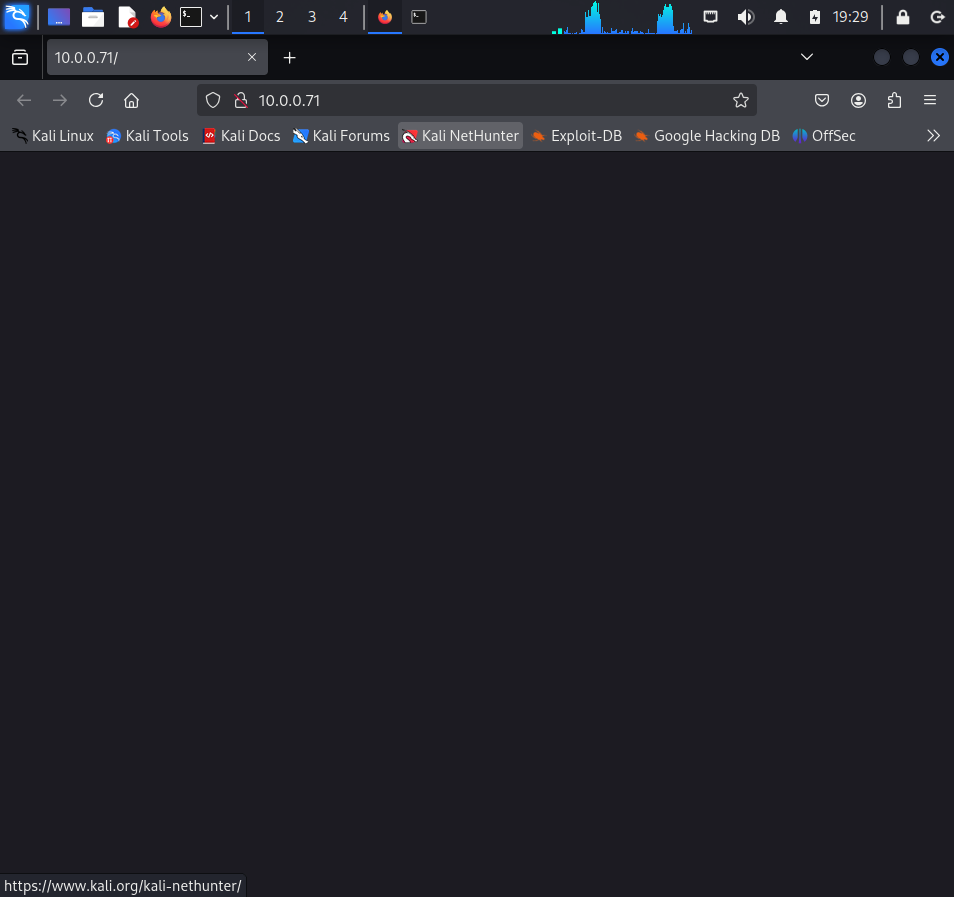
The next part of the assignment required Netcat listeners on Ubuntu to catch all traffic and log it. At first I ran into many problems getting the listener set up correctly. The first problem I encountered was needing to run the command as root, using sudo. Then after a small typo, and a few adjustments to ensure traffic from the attacker as well as from the honeypot were logged. Finally settled on commands 10-14, which set up the listeners for HTTP, FTP, SSH, SMTP, and Telnet traffic respectively. These commands also sent fake replies to the attacker, intended to look (relatively) real so as to not raise the attacker’s suspicion.



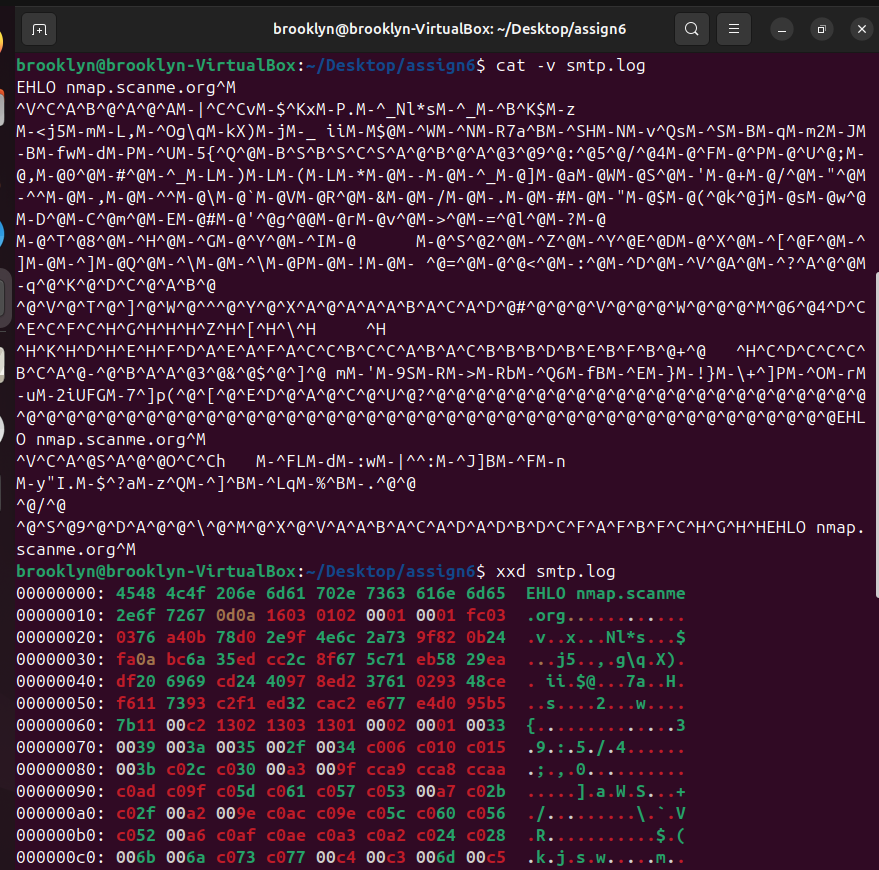
Now that all of the listeners and Ubuntu side of these assignments, it was time to begin the ‘attack’ on the Kali side of things. In order to connect to the five services being hosted on the Ubuntu machine, I used commands 16-21 to achieve this, once connected to the services I filled the connection with random traffic to discover it in the logs as well as WireShark. As an additional attack I ran a scan of the Ubuntu machine using command 22. As shown in the image below on the left, this scan revealed the five open ports running on the Ubuntu machine, posts 22, 23, 25, 21, and 80.



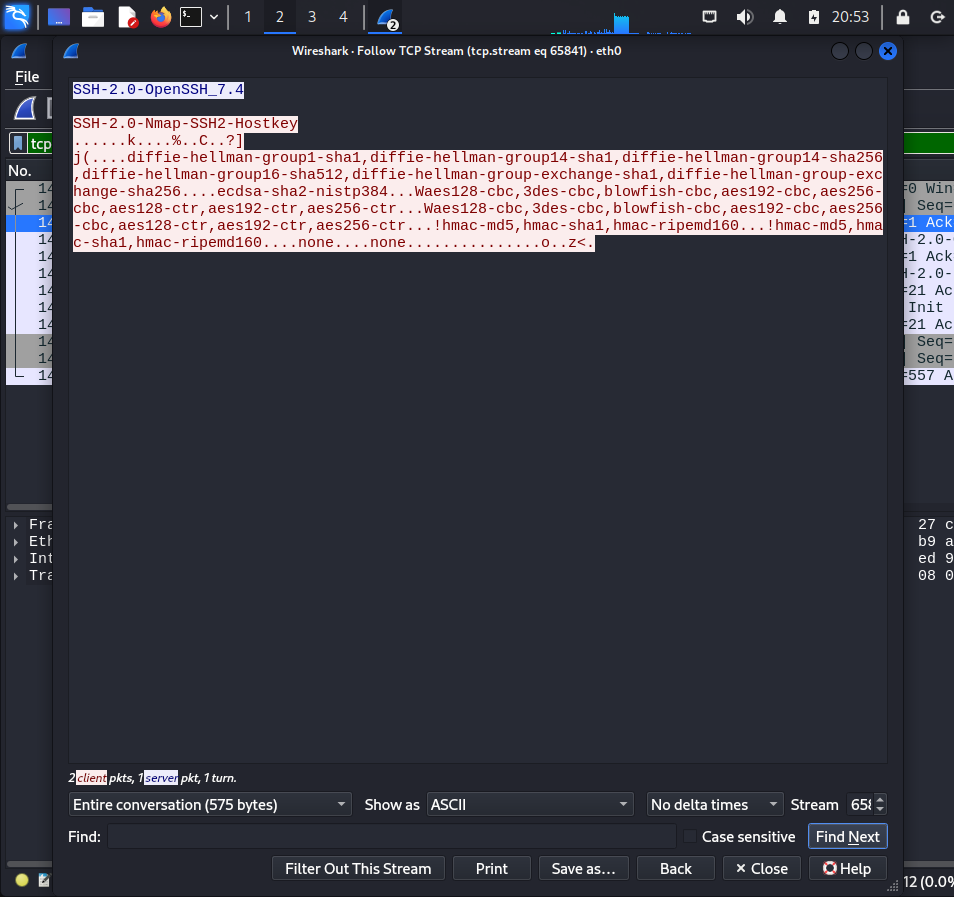
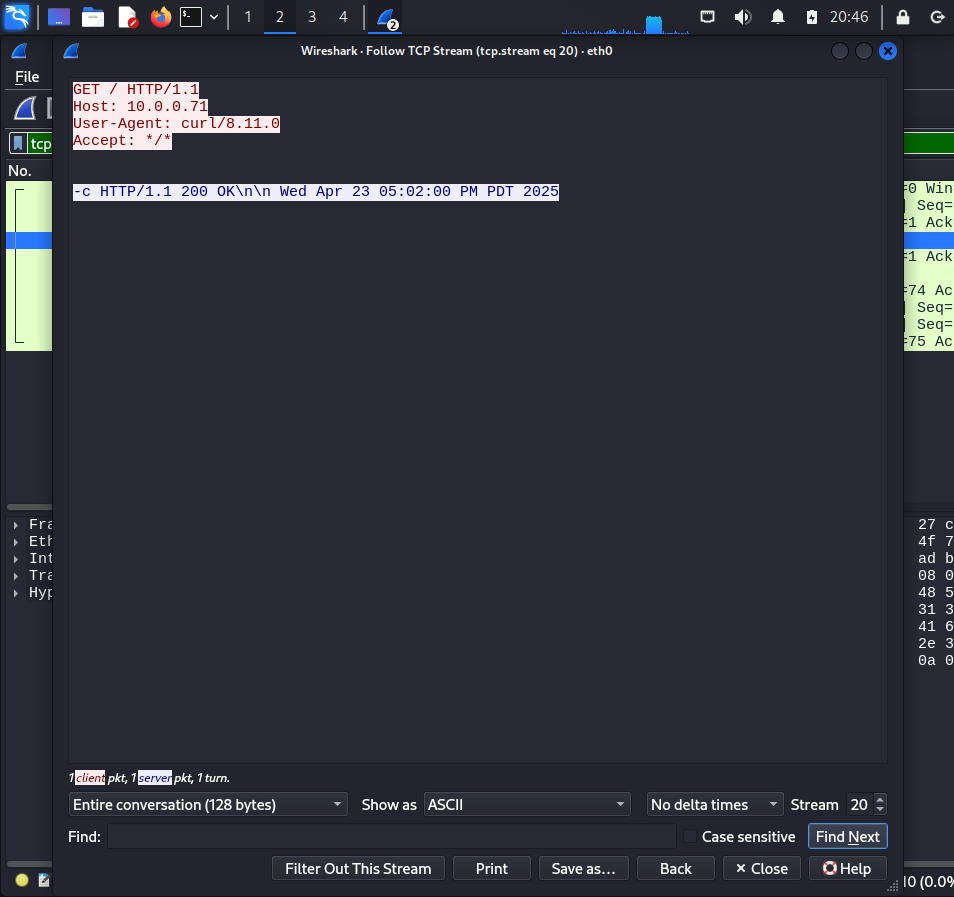
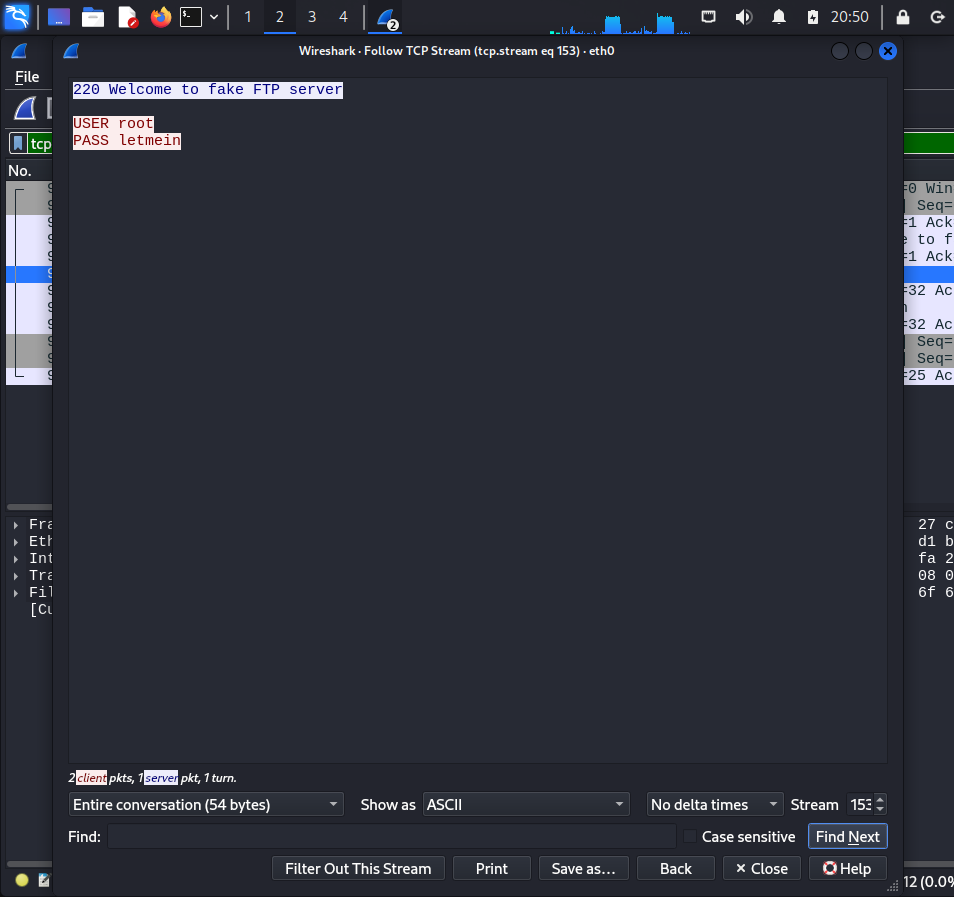
The images below show the feedback received from running commands 16 and 17 respectively. The left image shows the connection to the locally hosted web page from the Kali side, with a little more details there would be no red flags showing Kali that this service was being logged. However, the image on the right shows the feedback from the connection in the Ubuntu terminal.



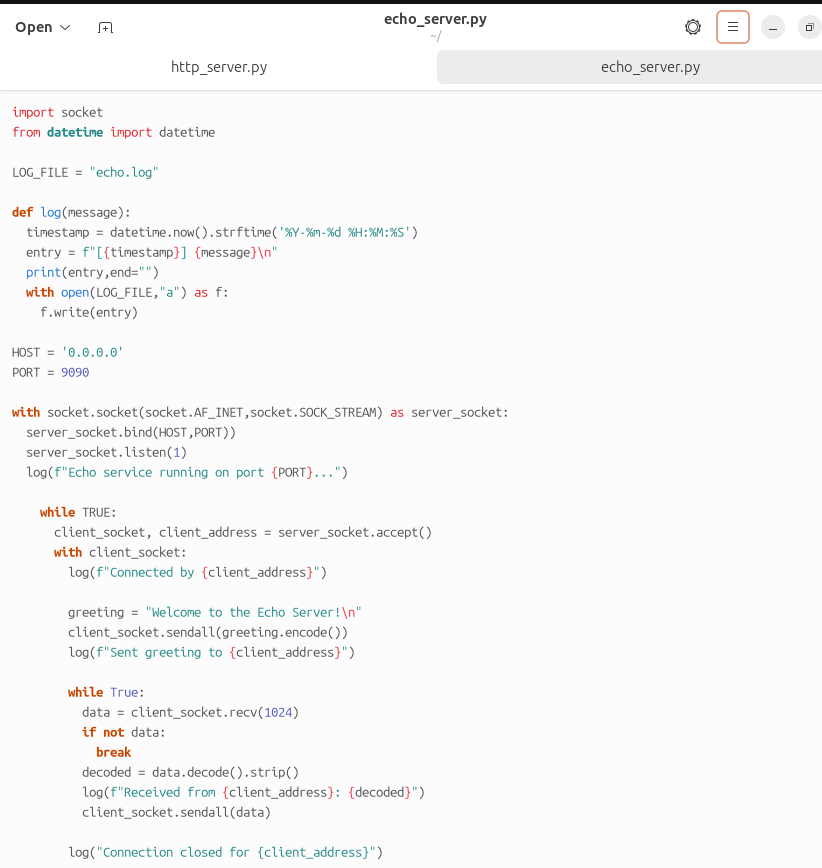
Next, I checked the logs produced by the various connections. Firstly, since the listener commands were run using sudo that make the permissions root, so I had to run command 23 to change the permissions of the file before I could open it. To ensure that the command ran properly, I ran command 24 to check if the permissions had updated. Once that was done I opened the smtp (port 23) log using command 25. The results of this, shown below on the left, revealed one plain text command I had typed in when connected, followed by many characters that could not be converted to ascii. These characters are most likely partial inputs that are the result of the nmap scan. In order to get a better understanding of these seemingly random values in the log, I ran commands 26 and 27 to view the results of the log in a different manner. The results of these are shown below on the right.



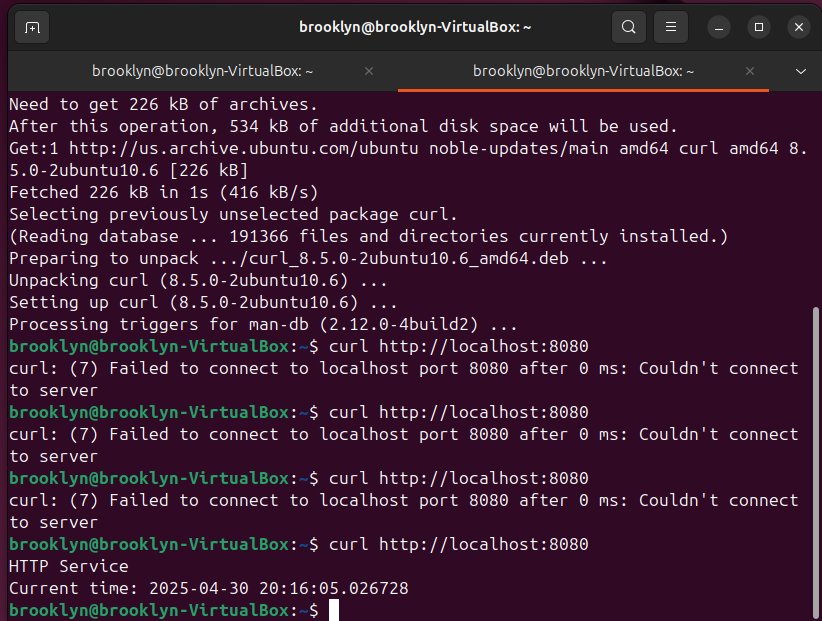
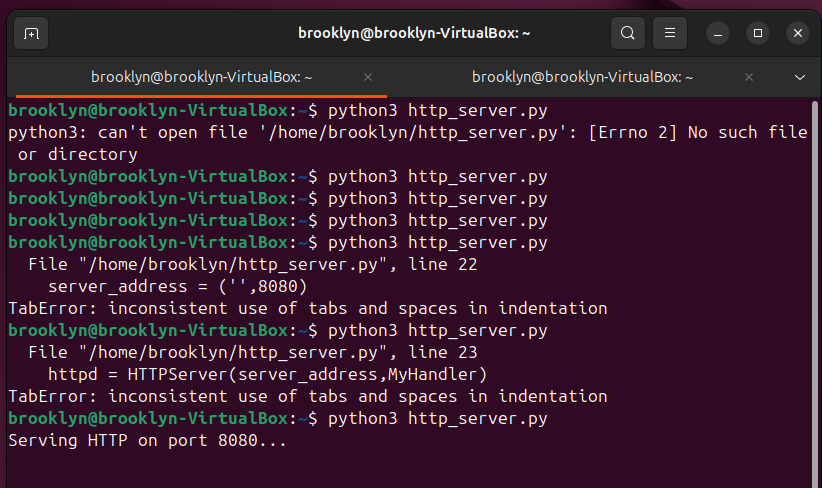
The images below show the WireShark results on ports 21, 80, and 22, respectively going left to right and top to bottom. From this we can see all traffic sent between the two machines. Most of which is available to read in plain text and is unencrypted. Blue scripts are from the Ubuntu, honeypot server, with red from the Kali, attacker server.



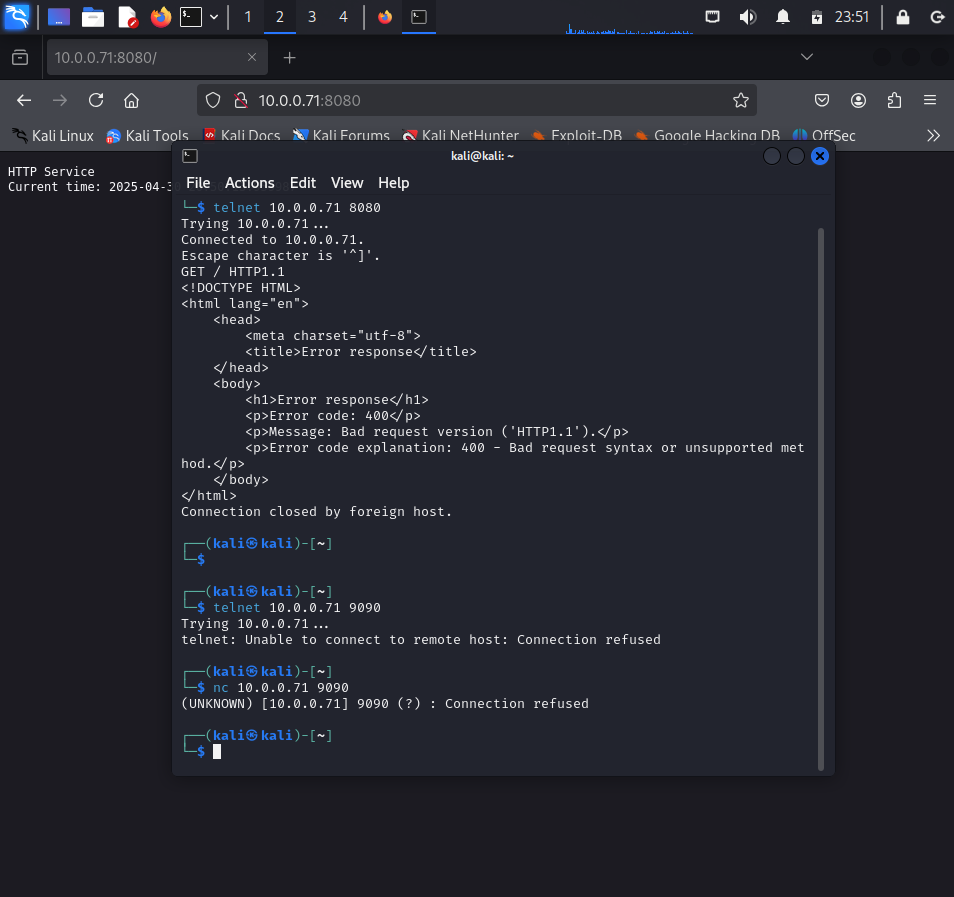
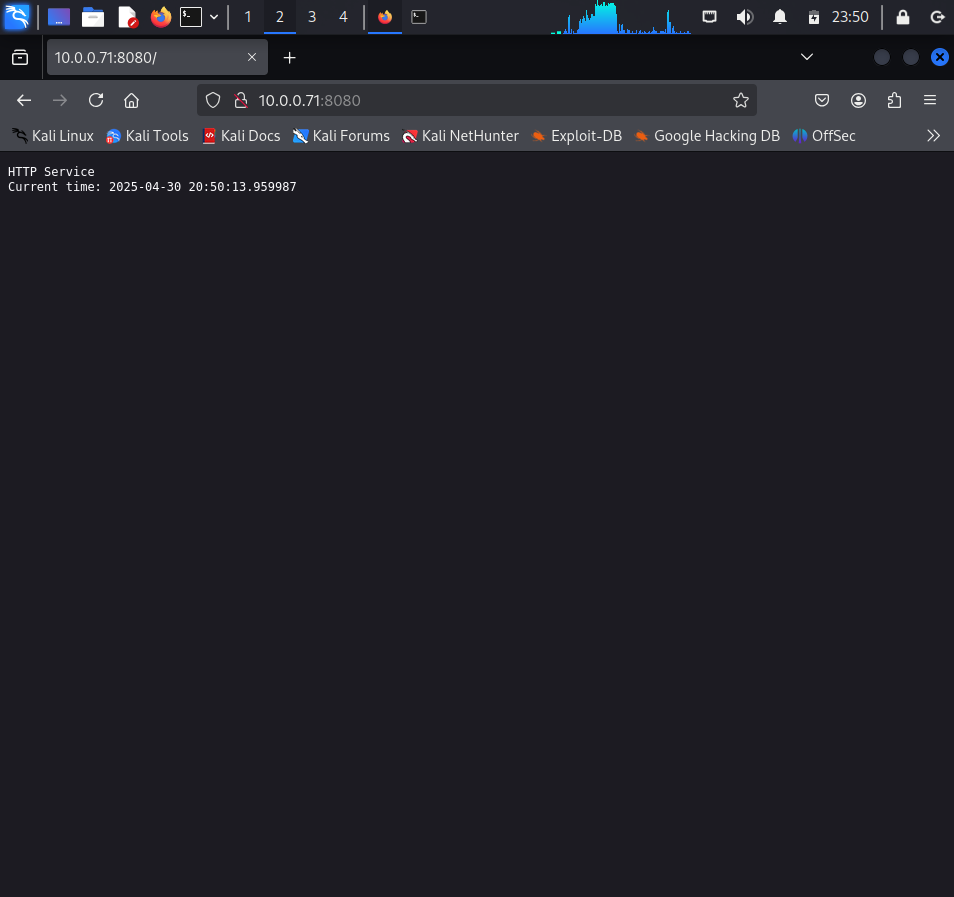
For further analysis, I attempted to initialize my honey pot server to make it more realistic and dynamic. In order to do this, I created two python scripts, http\_server.py and echo\_server.py. These scripts would simulate an http web server on port 8080 and an “echo” server set up on port 9090. Both of these scripts involved a segment to log incoming and outgoing traffic, making sure to note the time, the sender and receiver of the traffic, as well as the message or traffic itself.



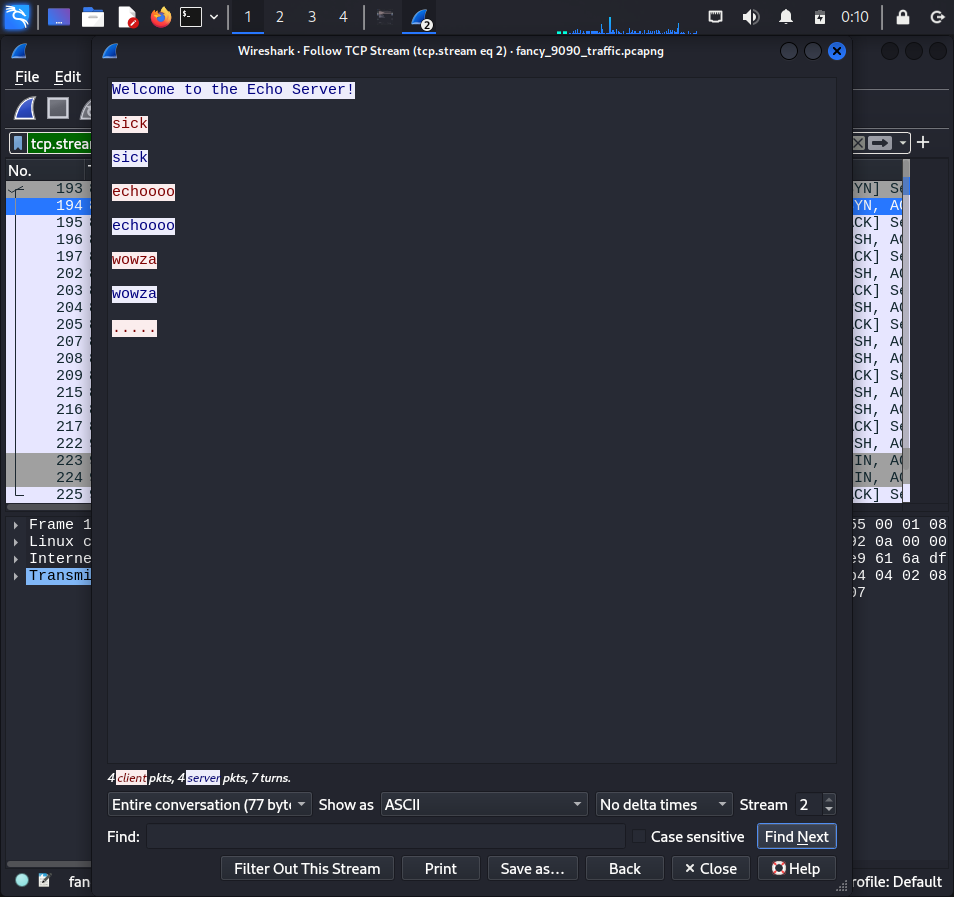
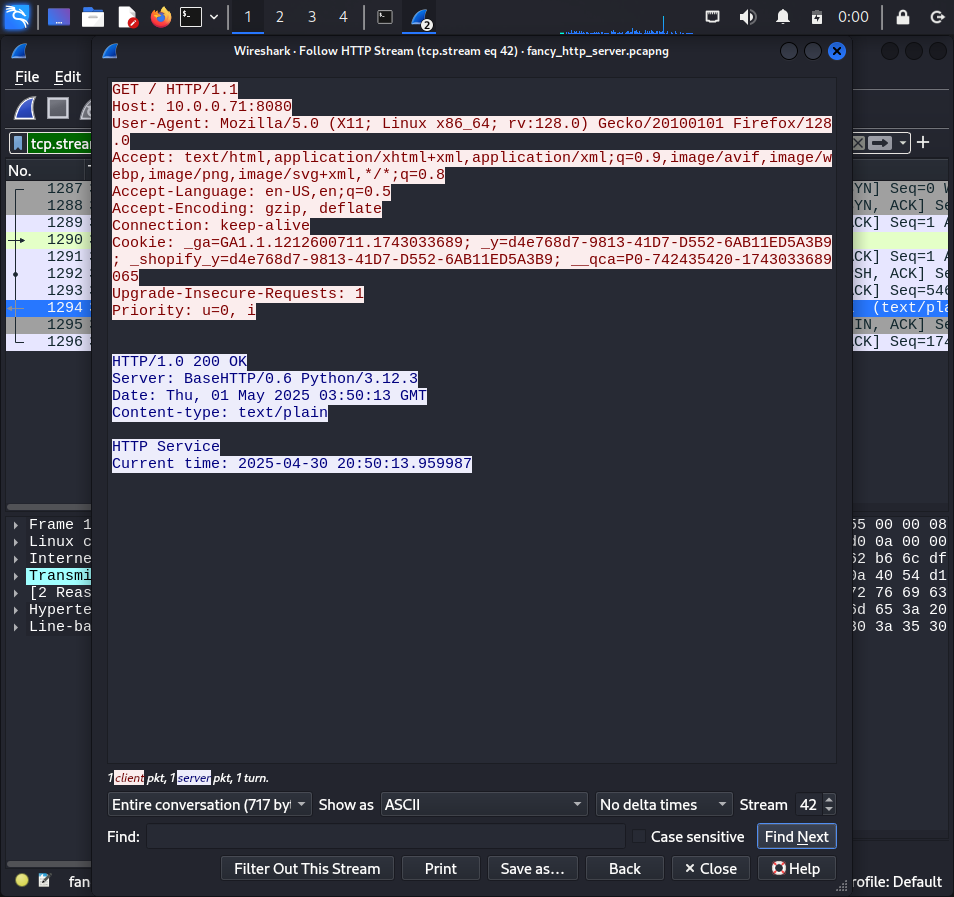
Now that the scripts had been written, I used commands 28 and 29 to run the script and thus start up the services. This involved me debugging the scripts in order to get them to connect, as shown in the left image below, I got a few errors before I was able to get the scripts running well. But eventually received the feedback that the server was up and running, “Serving HTTP on port 8080…” Still I ran command 30 to ensure that the server was able to be connected to, as opposed to just the output message being sent. Once I received the correct feedback I knew it was ready to attack from the Kali side.



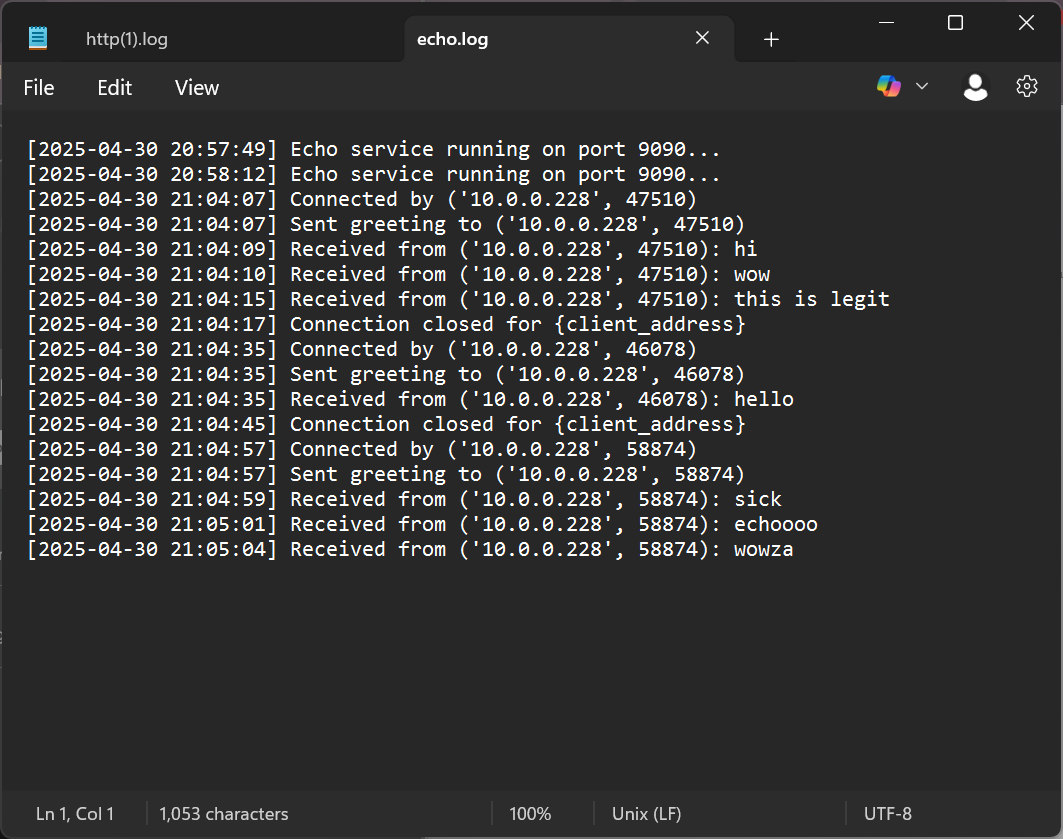
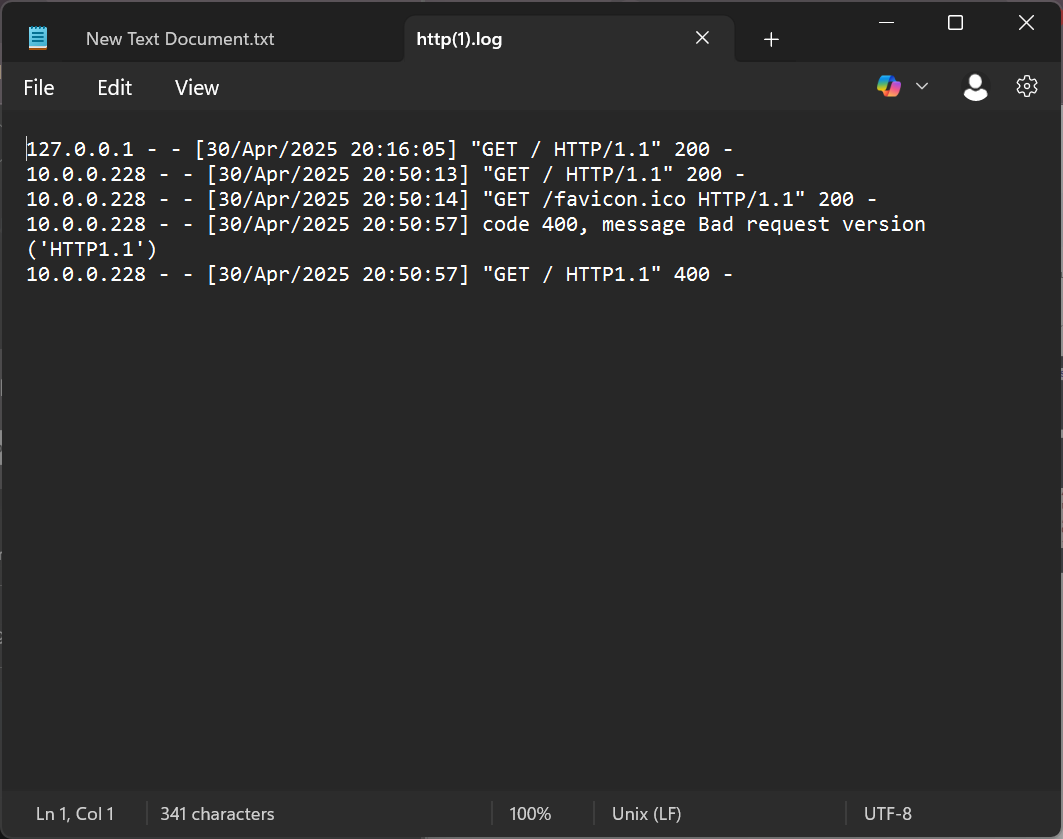
For this part I attempted to connect in more natural ways, so I opened the Firefox browser in Kali and entered the url at command 31. I received the correct feedback and moved on to check this http server from the terminal, and so I ran command 32. This displayed the web traffic information in the form of html script. Next, I moved on to connect to the echo server running on port 9090, so I ran command 33. After working out a few bugs, I was able to connect and all messages I sent into the chat were echoed back at me.



The below images show the WireShark packet captured on port 8080, on the left, and port 9090, on the right. From this we can see that none of the traffic was encrypted and it is all readable in plain text.



The images below are the final results of the logs files, with the http(1).log on the left and echo.log on the right. These logs tell us much more about the interaction between the attacker and the honey pot than the previous simple command run services. In these we can see the ip address of the attacker and the honeypot, which machine is sending which packets, as well as the exact time of the transfers.



# **Conclusion**

From this assignment we have seen the advantages and limitations of setting up a honeypot server. While there are simple ways to start up a server on a specific port, it becomes much more challenging and time consuming to create hyper realistic imitations of these services. However, it is immediately apparent if the advantages of having a honeypot server set up. Through this server, with careful logging, you can determine other users, possibly malicious, you can determine their ip address as well as keep records of messages and data being sent.

In the case of attackers, this can be incredibly helpful to determine the what, where, when, and how of the attackers methods. I do not include the who as it is possible the attacker is using a proxy and in that case this information can be somewhat telling but is not a surefire way. However, there is value in being able to determine which port the attacker is seeking as well as their method of attack, since all messages are recorded.

This can help to improve security as it can help to determine the mere presence of an attacker and what they are targeting. This helps ordinary users and victims to not only prepare for an attack but also isolate any corrupted software and control or remove it. In cases of worms it can also help to determine which users are unknowingly spreading this malware and so there are multiple ways to check and determine which users are corrupted. If a honeypot were to be combined with a firewall, we could almost immediately isolate or kick malicious users from our networks. This helping to mitigate any damage they might be causing.