

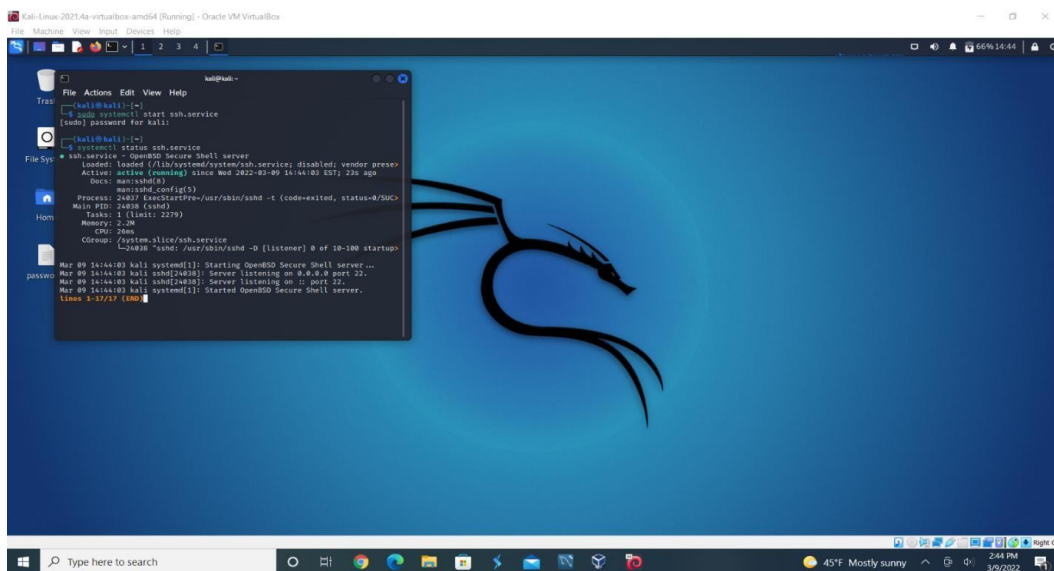
Overview:

The goal of this lab was to get familiar with setting up and configuring a firewall using **iptables** and gain a better understanding of how firewall rules work. I configured iptables rules to allow various types of traffic, including DNS lookups, inbound SSH connections, ping requests, HTTP/HTTPS traffic, and access to a Python web server running on port 8080.

Once the firewall rules were applied, I used the **CMD** to connect to the Kali machine via SSH, browse the website hosted on port 8080, and send ping packets to the Kali machine. Additionally, I verified that the Kali VM was able to browse the internet, ensuring the firewall rules were correctly set to allow the desired traffic while blocking others. This lab provided hands-on experience in configuring and testing firewall settings to control network access.

Analysis:

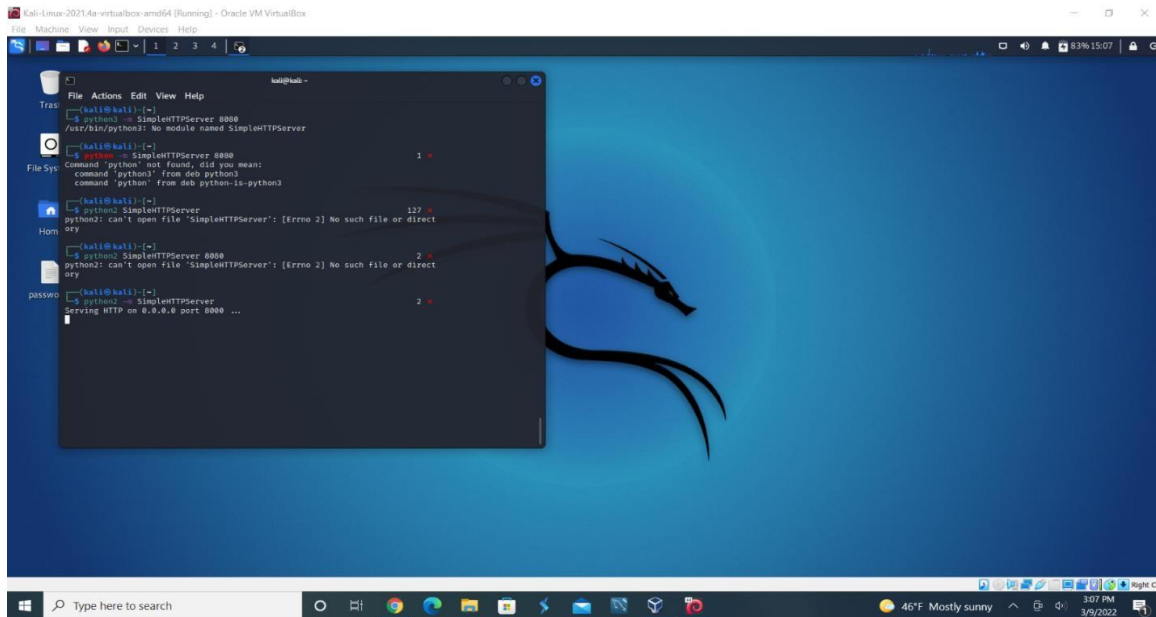
The first step in the process was to configure the **ssh.service** on the Kali VM, enabling remote SSH access.



```
Kali- Linux-2021.4a-virtualbox-amd64 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
1 2 3 4
kali@kali:~$ sudo systemctl start ssh.service
[sudo] password for kali:
kali@kali:~$ systemctl status ssh.service
systemd: start ssh.service
ssh.service - OpenSSH Secure Shell server
Loaded: loaded (/lib/systemd/systemd/ssh.service; disabled; vendor preset:
Active: active (running) since Wed 2022-03-09 14:44:03 EST; 23s ago
Docs: man:ssh(8)
      man:ssh_config(5)
Process: 24832 ExecStartPre=/usr/sbin/sshd -t (code=exited, status=0/SUC>
Main PID: 24838 (sshd)
Tasks: 2 (limit: 229)
Memory: 2.2M
CGroup: /system.slice/ssh.service
        └─24838 "sshd: /usr/sbin/sshd -D [listener] 0 of 10-100 startups

Mar 09 14:44:03 kali systemd[1]: Starting OpenSSH Secure Shell server...
Mar 09 14:44:03 kali sshd[24838]: Server listening on 0.0.0.0 port 22.
Mar 09 14:44:03 kali sshd[24838]: Server listening on :: port 22.
Mar 09 14:44:03 kali systemd[1]: Started OpenSSH Secure Shell server.
lines 1-17/17 (END)
```

Next, I set up a simple Python web server to listen on port 8080. This allowed me to host a basic website for testing access through the firewall.



```
kali@kali:~$ python3 -m SimpleHTTPServer 8080
/usr/bin/python3: No module named SimpleHTTPServer

kali@kali:~$ python3 -m SimpleHTTPServer 8080
1
python3: can't open file 'SimpleHTTPServer': [Errno 2] No such file or directory

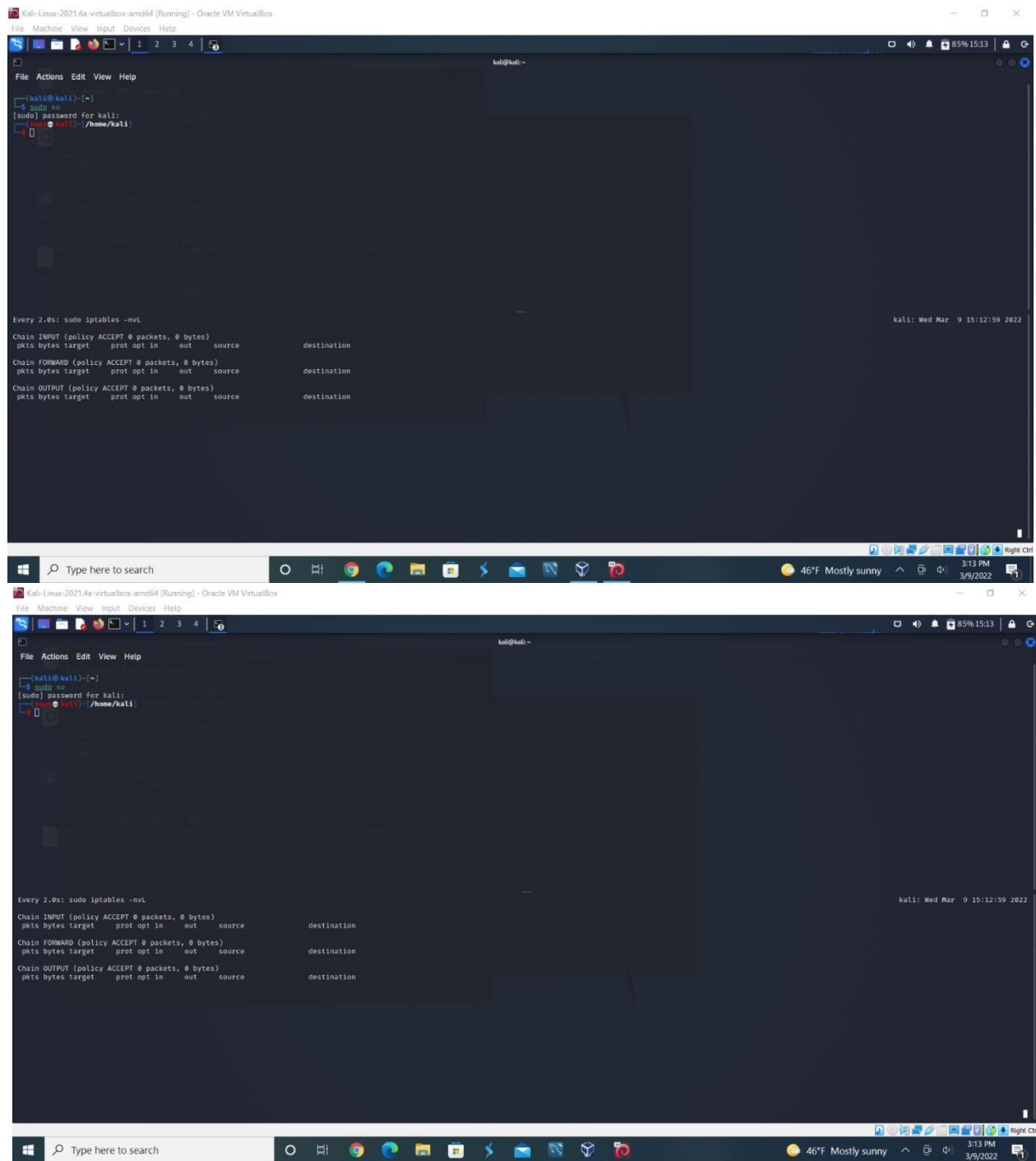
kali@kali:~$ python3 --help
python3 --help
python3: can't open file 'SimpleHTTPServer': [Errno 2] No such file or directory

kali@kali:~$ python3 -m http.server 8080
2
python3: can't open file 'SimpleHTTPServer': [Errno 2] No such file or directory

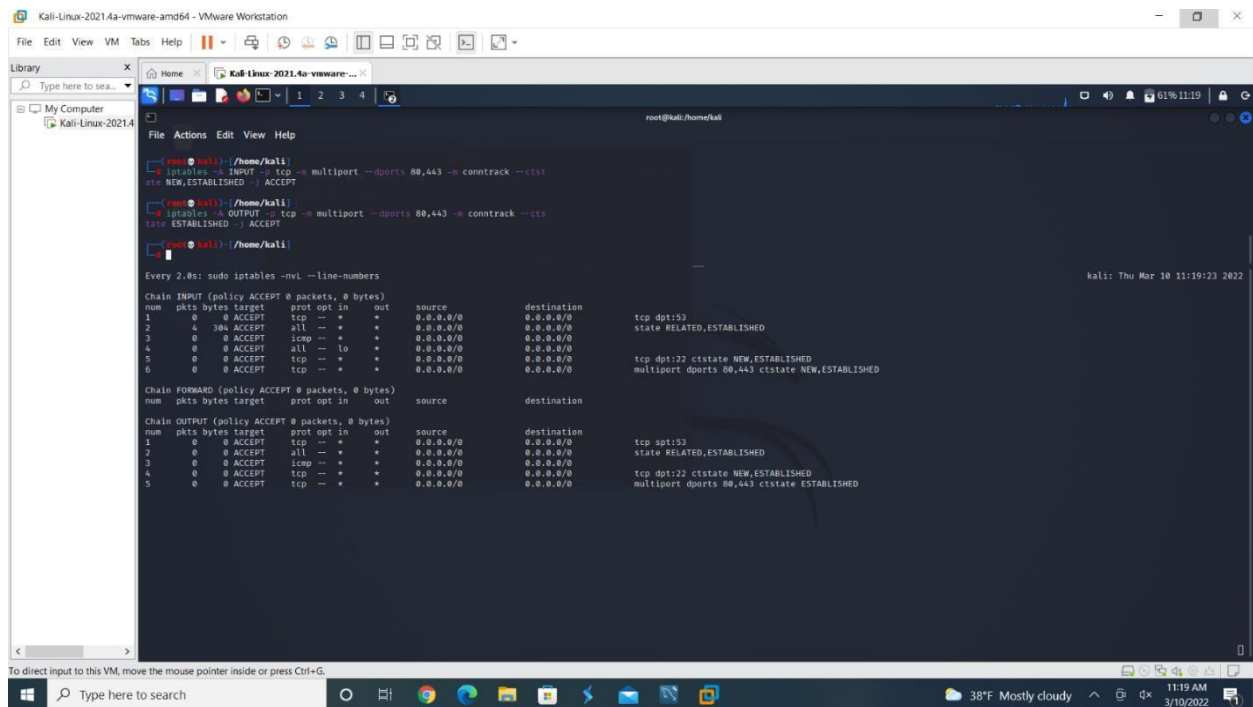
kali@kali:~$ python3 -m http.server 8080
2
Serving HTTP on 0.0.0.0 port 8080 ...
```

After setting up the web server, I switched to the **root** user and used the command `watch 'sudo iptables -nvL --line-numbers'` to display the current iptables rules in real-time. From there, I added firewall rules to allow the following traffic:

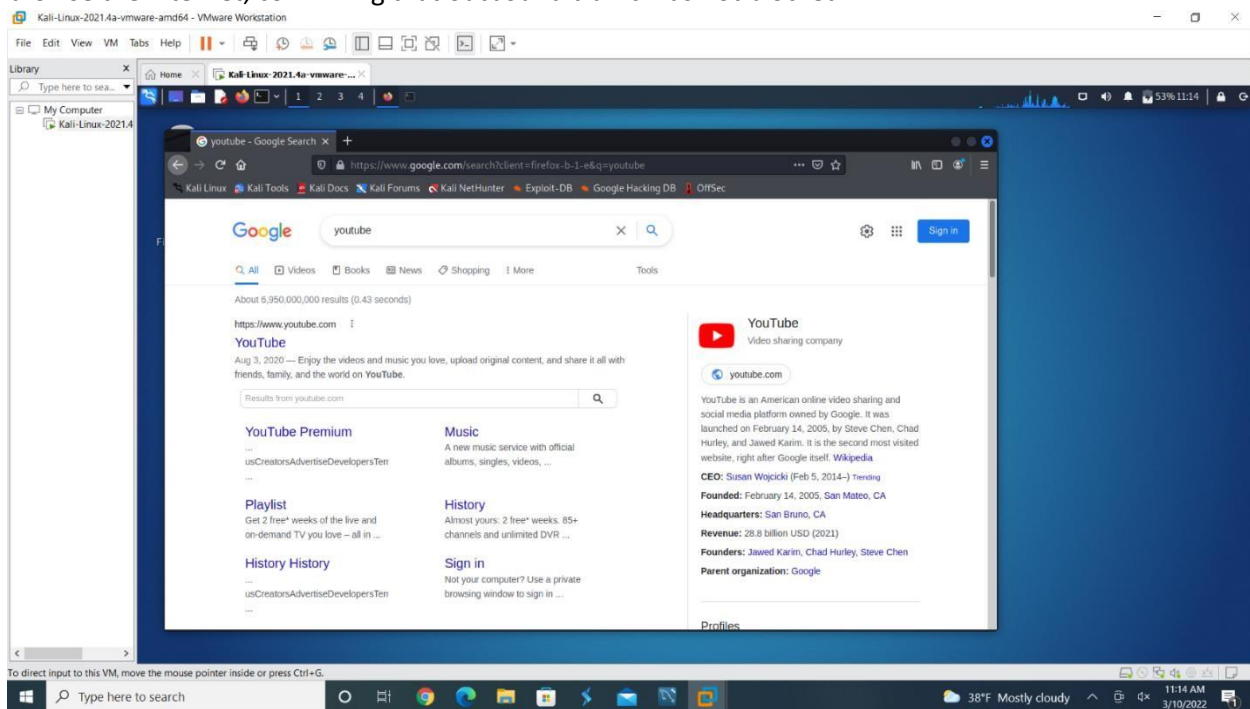
- DNS lookups
- Inbound SSH connections
- Ping requests
- HTTP/HTTPS traffic
- Access to the Python web server running on port 8080



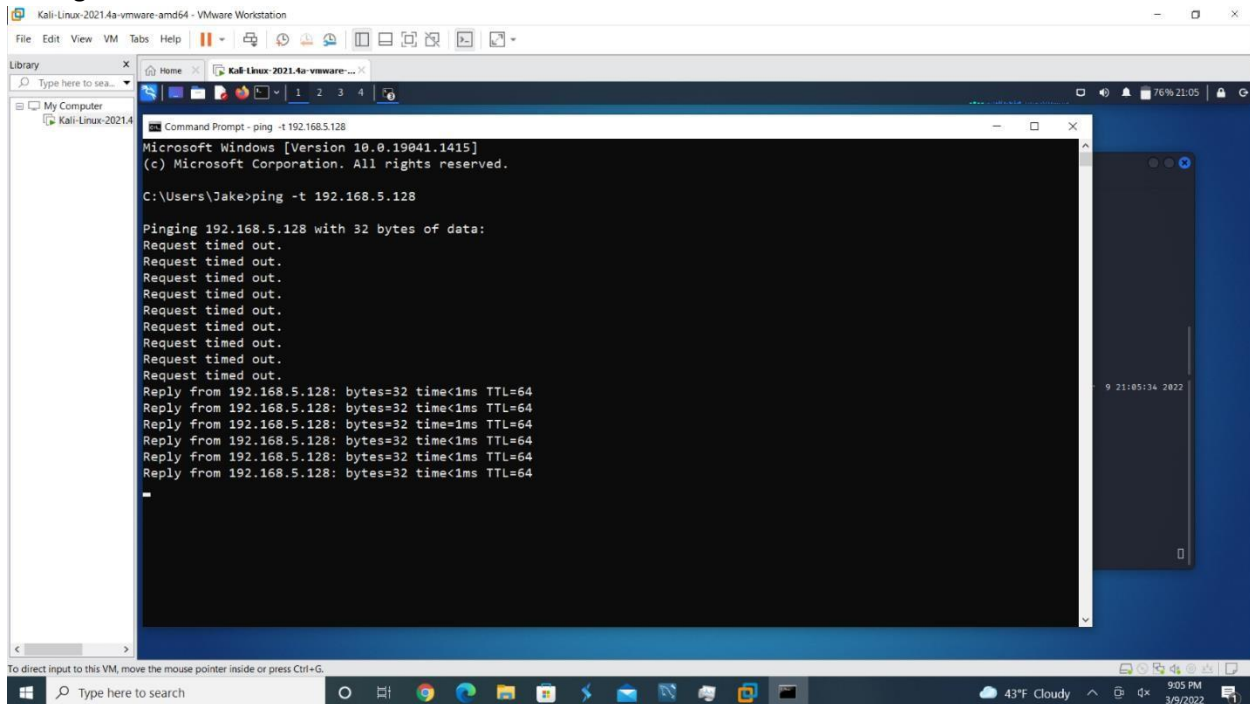
Here are those updated rules



Once the iptables rules were applied, I tested the connectivity. Using **Firefox** on the Kali VM, I was able to browse the internet, confirming that outbound traffic was not blocked.



On my personal machine, I sent **ping** requests to the Kali VM to ensure that ICMP traffic was allowed through the firewall.



The screenshot shows a Windows Command Prompt window titled "Command Prompt - ping -t 192.168.5.128". The window displays the output of a continuous ping command. The first eight attempts result in "Request timed out.", while the subsequent eight attempts are successful, showing "Reply from 192.168.5.128: bytes=32 time<1ms TTL=64". The background shows a VMware Workstation interface with a Kali Linux VM running.

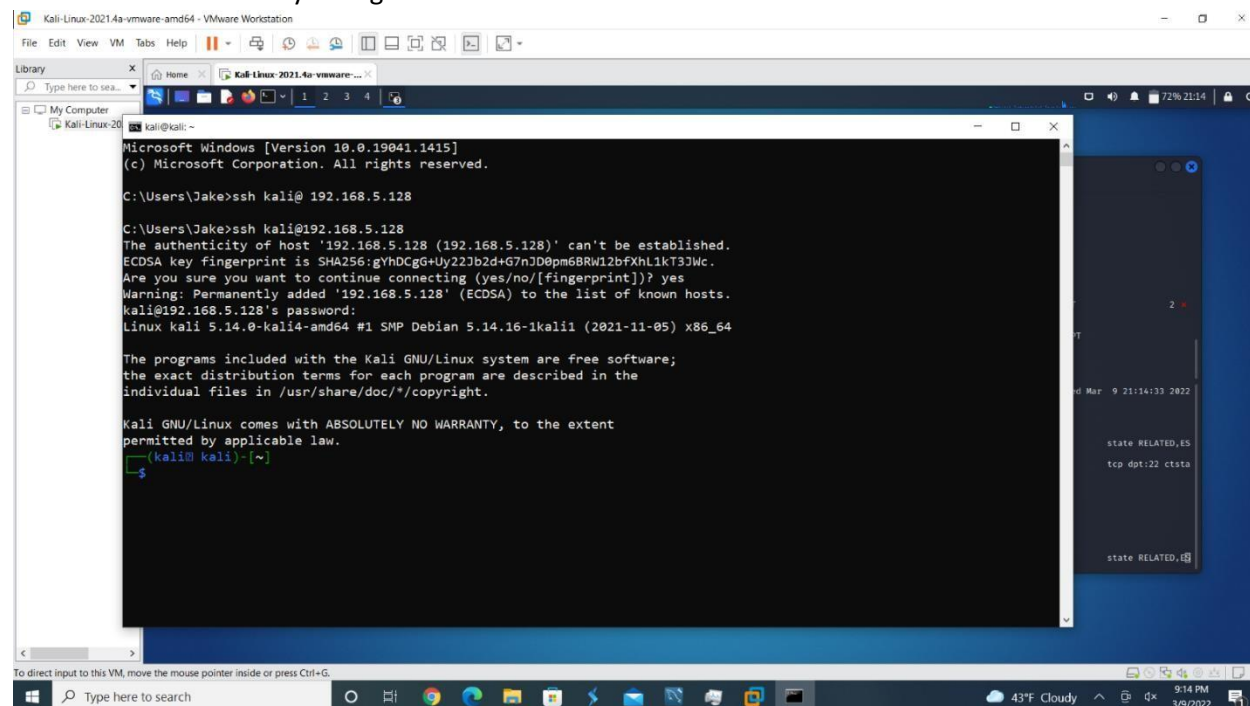
```
Microsoft Windows [Version 10.0.19041.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Jake>ping -t 192.168.5.128

Pinging 192.168.5.128 with 32 bytes of data:
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Request timed out.
Reply from 192.168.5.128: bytes=32 time<1ms TTL=64
Reply from 192.168.5.128: bytes=32 time<1ms TTL=64
Reply from 192.168.5.128: bytes=32 time<1ms TTL=64
Reply from 192.168.5.128: bytes=32 time<1ms TTL=64
Reply from 192.168.5.128: bytes=32 time<1ms TTL=64
Reply from 192.168.5.128: bytes=32 time<1ms TTL=64
Reply from 192.168.5.128: bytes=32 time<1ms TTL=64

```

Following that, I used **CMD** to successfully establish an **SSH** connection to the Kali VM, confirming that the firewall was correctly configured to allow inbound SSH traffic.



The screenshot shows a Windows Command Prompt window titled "Command Prompt - ssh -t 192.168.5.128". The window displays the output of an SSH connection attempt. It shows the host key fingerprint, a warning to permanently add the host to the list of known hosts, and the user's password. The connection is successful, and the prompt changes to "kali@kali:~". The background shows a VMware Workstation interface with a Kali Linux VM running.

```
Microsoft Windows [Version 10.0.19041.1415]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Jake>ssh kali@ 192.168.5.128

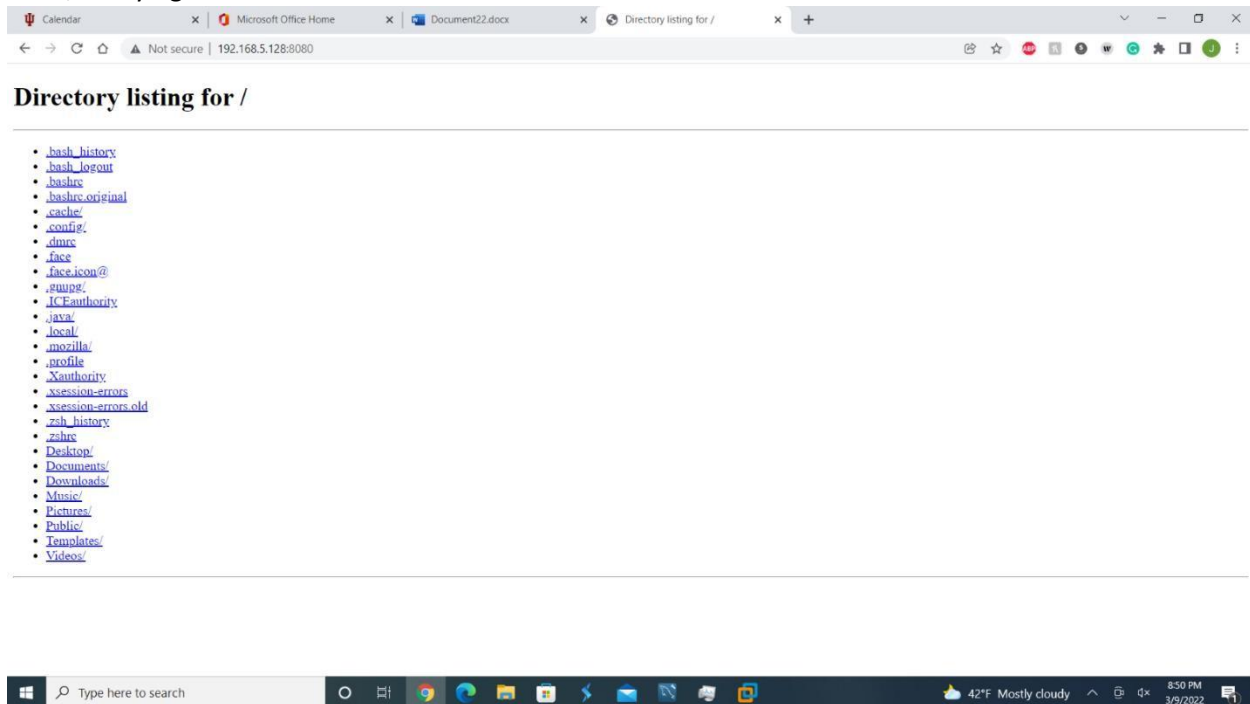
C:\Users\Jake>ssh kali@192.168.5.128
The authenticity of host '192.168.5.128 (192.168.5.128)' can't be established.
ECDSA key fingerprint is SHA256:gYhDCgG+Uy22Jb2d+G7n3D8pm6BRW12bfXhL1KT3Jwc.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.5.128' (ECDSA) to the list of known hosts.
kali@192.168.5.128's password:
Linux kali 5.14.0-kali4-amd64 #1 SMP Debian 5.14.16-1kali1 (2021-11-05) x86_64

The programs included with the Kali GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Kali GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
kali@kali:~$

```

Lastly, I used **Google Chrome** on my personal computer to access the Python web server running on port 8080, verifying that the firewall rules allowed access to the server.



In conclusion, all the configured firewall rules worked as expected, enabling the desired services and blocking all unauthorized traffic.