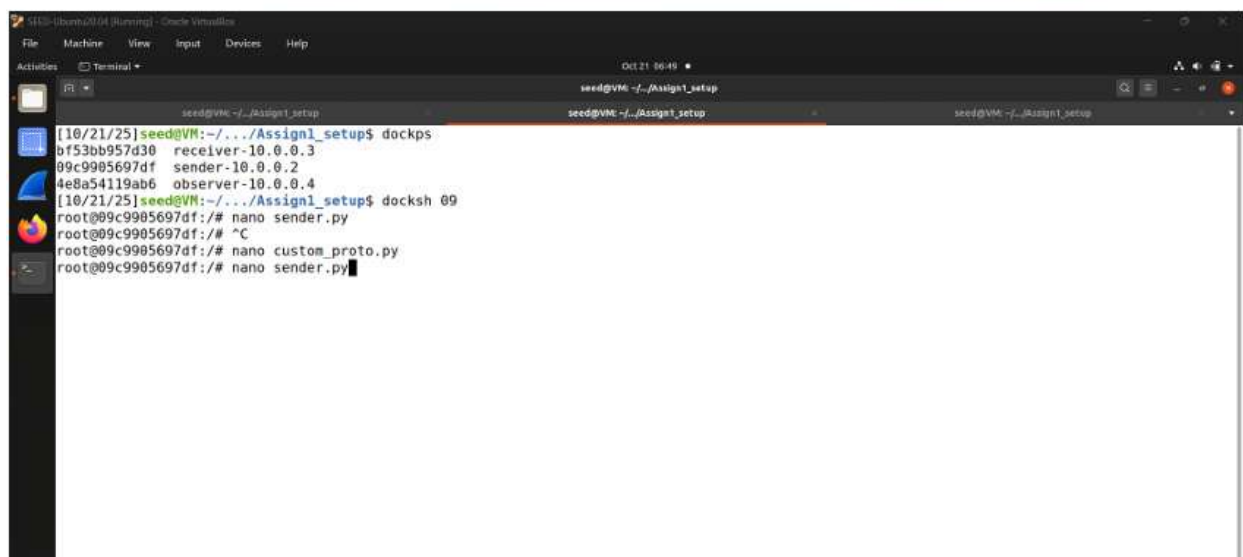
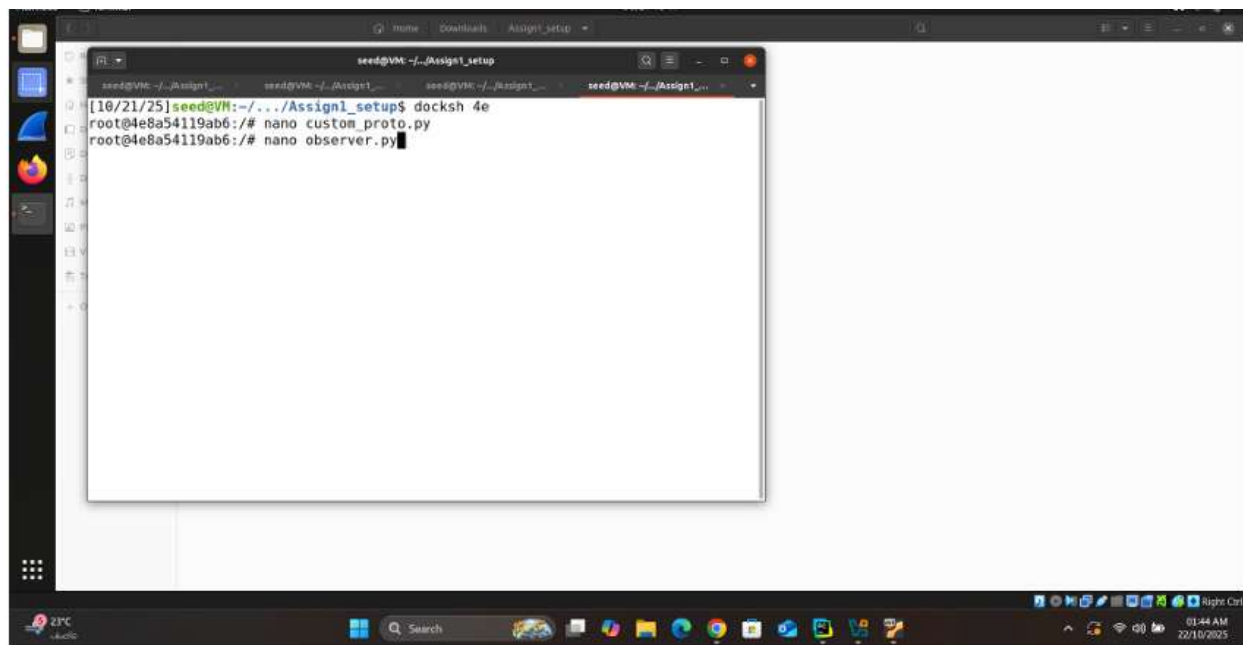


```
seed@VM: ~/.../Assign1_setup
[10/21/25]seed@VM:~/.../Assign1_setup$ ls
docker-compose.yml  scaps  started
[10/21/25]seed@VM:~/.../Assign1_setup$ dcbuild
sender uses an image, skipping
receiver uses an image, skipping
observer uses an image, skipping
[10/21/25]seed@VM:~/.../Assign1_setup$ dcup
Starting observer-10.0.0.4 ... done
Starting sender-10.0.0.2 ... done
Starting receiver-10.0.0.3 ... done
Attaching to receiver-10.0.0.3, sender-10.0.0.2, observer-10.0.0.4
receiver-10.0.0.3 | 10: eth0@if11: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qd
isc noqueue state UP group default link-netnsid 0
receiver-10.0.0.3 |      inet 10.0.0.3/24 brd 10.0.0.255 scope global eth0
receiver-10.0.0.3 |      valid_lft forever preferred_lft forever
sender-10.0.0.2 | 8: eth0@if9: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc
noqueue state UP group default link-netnsid 0
sender-10.0.0.2 |      inet 10.0.0.2/24 brd 10.0.0.255 scope global eth0
sender-10.0.0.2 |      valid_lft forever preferred_lft forever
sender-10.0.0.2 | RTNETLINK answers: File exists
observer-10.0.0.4 | 6: eth0@if7: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdis
c noqueue state UP group default link-netnsid 0
observer-10.0.0.4 |      inet 10.0.0.4/24 brd 10.0.0.255 scope global eth0
observer-10.0.0.4 |      valid_lft forever preferred_lft forever
```

```
SEED-Ubuntu20.04 (Jupyter) - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
seed@VM: ~/.../Assign1_setup
[10/21/25]seed@VM:~/.../Assign1_setup$ dockps
bf53bb957d30 receiver-10.0.0.3
09c9905697df sender-10.0.0.2
4e8a54119ab6 observer-10.0.0.4
[10/21/25]seed@VM:~/.../Assign1_setup$ docksh bf5
root@bf53bb957d30:/# nano reciever.py
root@bf53bb957d30:/# nano receiver.py
root@bf53bb957d30:/# nano receiver.py
root@bf53bb957d30:/# nano custom_proto.py
root@bf53bb957d30:/#
```



```
GNU nano 4.8 custom_proto.py
from scapy.all import *
from Crypto.Cipher import AES
from Crypto.Util.Padding import pad, unpad

SECRET_KEY = b'1234567890abcdef'

def encrypt_data(data: bytes) -> bytes:
    cipher = AES.new(SECRET_KEY, AES.MODE_ECB)
    return cipher.encrypt(pad(data, AES.block_size))

def decrypt_data(data: bytes) -> bytes:
    cipher = AES.new(SECRET_KEY, AES.MODE_ECB)
    return unpad(cipher.decrypt(data), AES.block_size)

class CustomProto(Packet):
    name = "CustomProto"
    fields_desc = [
        ShortField("id", 0),
        ByteField("msg_type", 0),
        FieldLenField("len", None, length_of="data", fmt="N"),
        StrLenField("data", "", length_from=lambda pkt: pkt.len)
    ]

bind_layers(Ether, CustomProto, type=0x1234)
```

Wrote python file for sender

```
GNU nano 4.8 sender.py
from scapy.all import *
from custom_proto import CustomProto, encrypt_data
import time

INTERFACE = "eth0"
DEST_MAC = "02:42:ac:11:00:03"
TIMEOUT = 3

def send_frame(seq, message):
    encrypted_data = encrypt_data(message.encode())
    frame = Ether(dst=DEST_MAC, type=0x1234) / CustomProto(
        id=seq,
        msg_type=0,
        data=encrypted_data
    )
    sendp(frame, iface=INTERFACE, verbose=False)
    print(f"[SENDER] Sent frame #{seq}")

def main():
    print("[SENDER] Starting sender...")
    seq = 1
    while True:
        message = input("\nEnter message to send (or 'exit' to quit): ")
        if message.lower() == 'exit':
            print("[SENDER] Exiting...")
            break
        send_frame(seq, message)
        seq += 1
```

```
GNU nano 4.8
from scapy.all import *
from custom_proto import CustomProto, decrypt_data

INTERFACE = "eth0"

def send_ack(seq, sender_mac):
    try:
        ack_frame = Ether(dst=sender_mac, type=0x1234) / CustomProto(
            id=seq,
            msg_type=1,
            data=b'ACK'
        )
        sendp(ack_frame, iface=INTERFACE, verbose=False)
        print(f"[RECEIVER] Sent ACK for frame #{seq}")
    except Exception as e:
        print(f"[RECEIVER] Error sending ACK: {e}")

def handle_frame(pkt):
    try:
        if CustomProto not in pkt:
            return

        if pkt[CustomProto].msg_type != 0:
            return

        seq = pkt[CustomProto].id
        encrypted_data = pkt[CustomProto].data
```

Wrote observer.py code

```
GNU nano 4.8
from scapy.all import *
from custom_proto import CustomProto

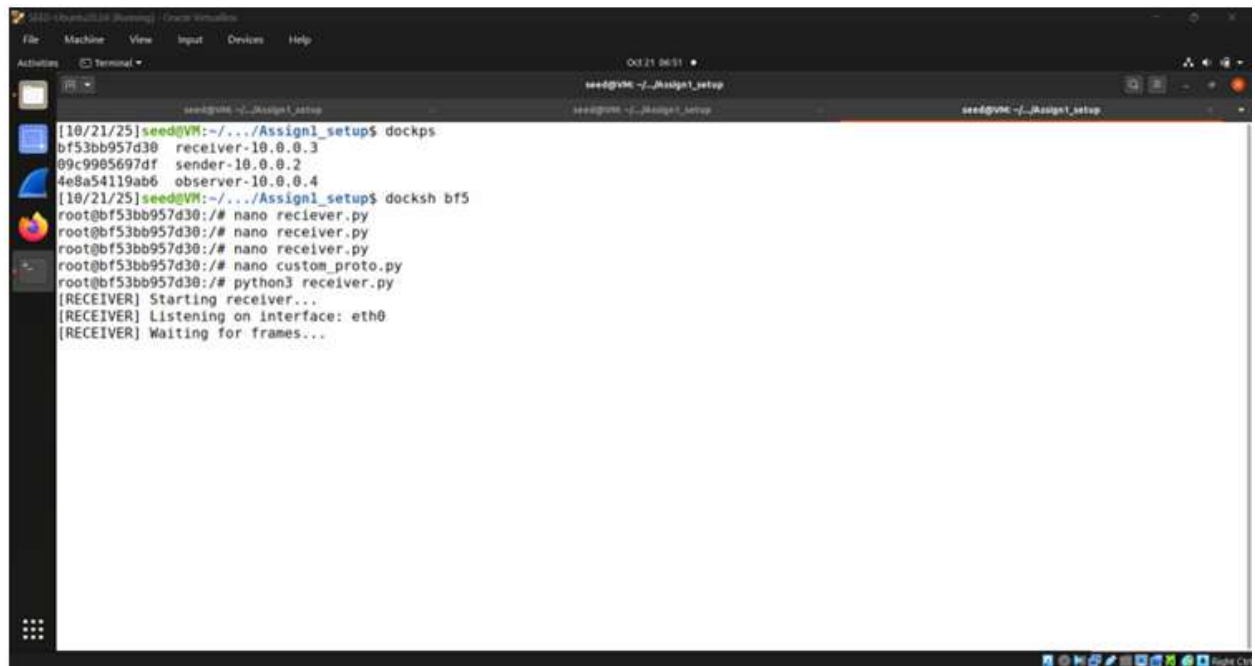
INTERFACE = "eth0"

def handle_packet(pkt):
    if CustomProto not in pkt:
        return

    seq = pkt[CustomProto].id
    msg_type = pkt[CustomProto].msg_type
    src_mac = pkt.src
    dst_mac = pkt.dst

    if msg_type == 0:
        print(f"DATA frame #{seq}: {src_mac} -> {dst_mac}")
    else:
        print(f"ACK frame #{seq}: {src_mac} -> {dst_mac}")

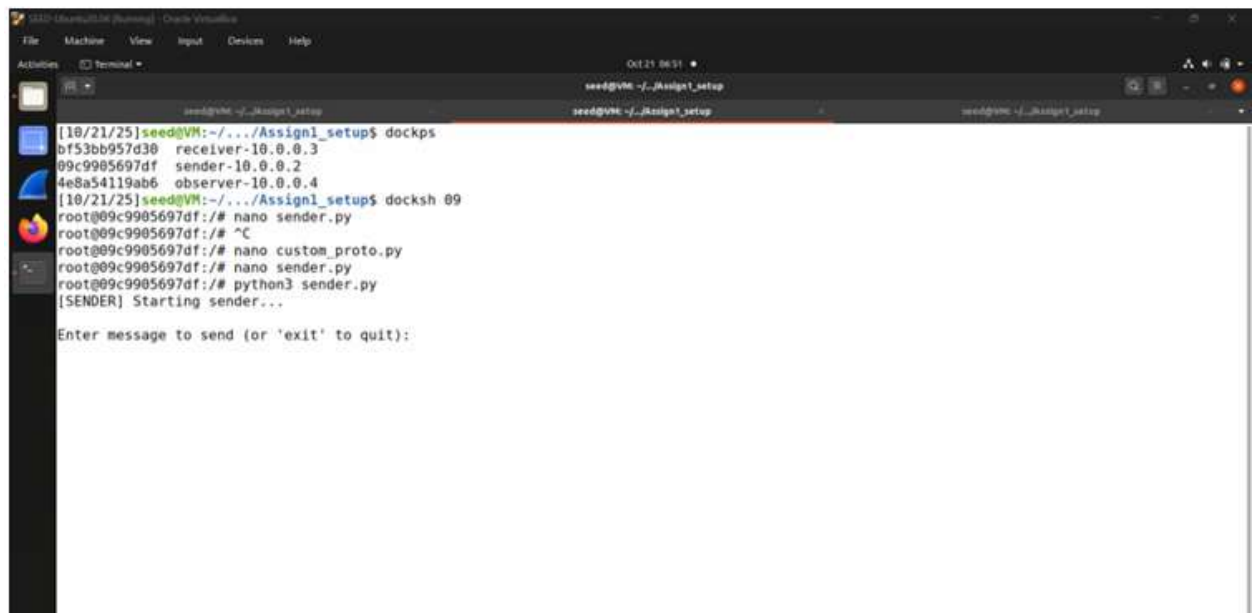
def main():
    print("[OBSERVER] Listening...")
    try:
        sniff(
            iface=INTERFACE,
            filter="ether proto 0x1234",
            prn=handle_packet,
```



The screenshot shows a terminal window titled "seed@vm: ~/Assign1_setup". The user has run the following commands:

```
[10/21/25]seed@VM:~/Assign1_setup$ dockps
bf53bb957d30  receiver-10.0.0.3
09c9905697df  sender-10.0.0.2
4e8a54119ab6  observer-10.0.0.4
[10/21/25]seed@VM:~/Assign1_setup$ docksh bf5
root@bf53bb957d30:/# nano receiver.py
root@bf53bb957d30:/# nano receiver.py
root@bf53bb957d30:/# nano receiver.py
root@bf53bb957d30:/# nano custom_proto.py
root@bf53bb957d30:/# python3 receiver.py
[RECEIVER] Starting receiver...
[RECEIVER] Listening on interface: eth0
[RECEIVER] Waiting for frames...
```

Started sender



The screenshot shows a terminal window titled "seed@vm: ~/Assign1_setup". The user has run the following commands:

```
[10/21/25]seed@VM:~/Assign1_setup$ dockps
bf53bb957d30  receiver-10.0.0.3
09c9905697df  sender-10.0.0.2
4e8a54119ab6  observer-10.0.0.4
[10/21/25]seed@VM:~/Assign1_setup$ docksh 09
root@09c9905697df:/# nano sender.py
root@09c9905697df:/# ^C
root@09c9905697df:/# nano custom_proto.py
root@09c9905697df:/# nano sender.py
root@09c9905697df:/# python3 sender.py
[SENDER] Starting sender...

Enter message to send (or 'exit' to quit):
```

```
seed@VM: ~/Assign1_setup
[10/21/25]seed@VM:~/Assign1_setup$ dockps
bf53bb957d30 receiver-10.0.0.3
09c9905697df sender-10.0.0.2
4e8a54119ab6 observer-10.0.0.4
[10/21/25]seed@VM:~/Assign1_setup$ docksh 09
root@09c9905697df:/# nano sender.py
root@09c9905697df:/# ^C
root@09c9905697df:/# nano custom_proto.py
root@09c9905697df:/# nano sender.py
root@09c9905697df:/# python3 sender.py
[SENDER] Starting sender...

Enter message to send (or 'exit' to quit): ahmed
[SENDER] Sent frame #1
[SENDER] Waiting for ACK #1...
[SENDER] ACK received for frame #1

Enter message to send (or 'exit' to quit):
```

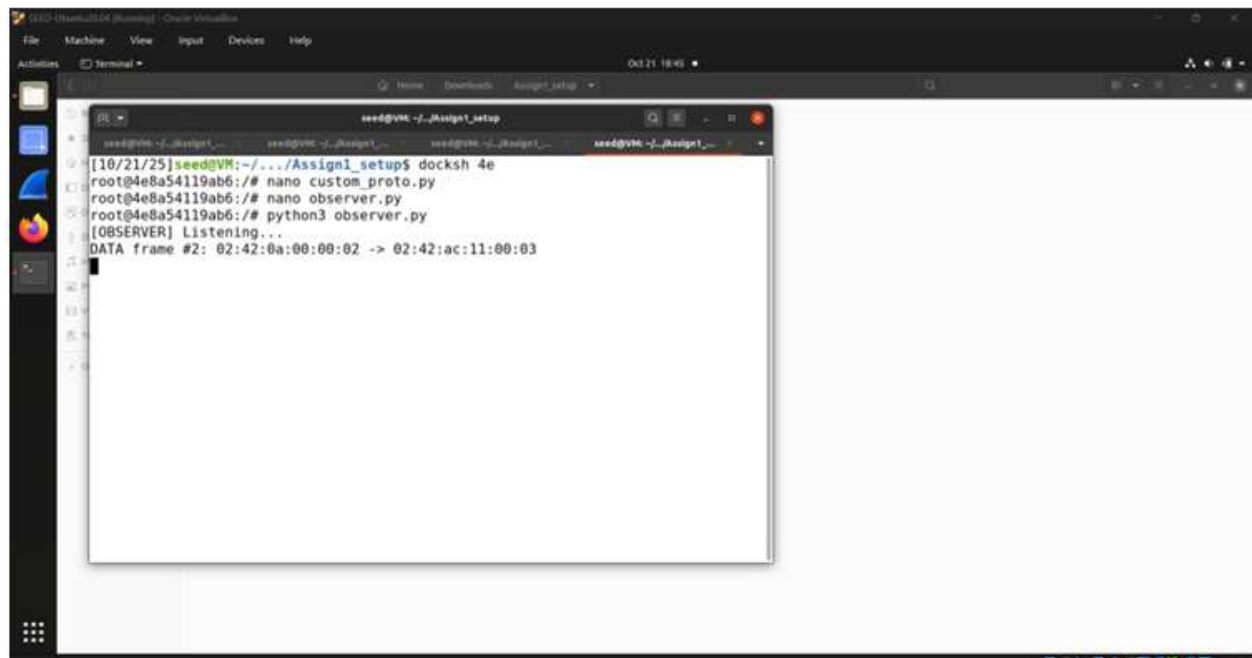
Message received. ack sent from receiver and received at sender

```
Oct 21 00:51
File Machine View Input Devices Help
Activities Terminal
seed@VM: ~/Assign1_setup
[10/21/25]seed@VM:~/Assign1_setup$ dockps
bf53bb957d30 receiver-10.0.0.3
09c9905697df sender-10.0.0.2
4e8a54119ab6 observer-10.0.0.4
[10/21/25]seed@VM:~/Assign1_setup$ docksh bf5
root@bf53bb957d30:/# nano reciever.py
root@bf53bb957d30:/# nano receiver.py
root@bf53bb957d30:/# nano receiver.py
root@bf53bb957d30:/# nano custom_proto.py
root@bf53bb957d30:/# python3 receiver.py
[RECEIVER] Starting receiver...
[RECEIVER] Listening on interface: eth0
[RECEIVER] Waiting for frames...

[RECEIVER] Received frame #1 from 02:42:0a:00:00:02
=====
Frame #1
From: 02:42:0a:00:00:02
Message: ahmed
=====

[RECEIVER] Sent ACK for frame #1
```


Observer saw frame



The screenshot shows a Kali Linux virtual machine interface. A terminal window is open, displaying the following commands and output:

```
seed@VM: ~/Assign1_setup$ docksh 4e
root@4e8a54119ab6:/# nano custom_proto.py
root@4e8a54119ab6:/# nano observer.py
root@4e8a54119ab6:/# python3 observer.py
[OBSERVER] Listening...
DATA frame #2: 02:42:0a:00:00:02 -> 02:42:ac:11:00:03
```

The terminal output indicates that a Docker container named '4e' was successfully started, and the observer.py script is running and listening for network traffic. The captured data frame shows a packet from IP 02:42:0a:00:00:02 to IP 02:42:ac:11:00:03.