```python
#Robert Timmons // Pycharm Testing
#Done with research online and through ChatGPT

from tqdm import tqdm
from scapy.layers.l2 import ARP, Ether, srp
import ipaddress

# Camera OUIS categorized by manufacturer
camera_ouis = {
    "Axis": [
        "00:40:8c",  # AXIS Communications
        "ac:cc:8e",  # AXIS Communications
        "e8:27:25",  # AXIS Communications
        "b8:a4:4f",  # AXIS Communications
    ],
    "Vivotek": [
        "00:02:d1",  # Vivotek
    ],
    "Hanwha": [
        "e4:30:22",  # Hanwha WISENET
        "00:09:18",  # Hanwha WISENET
    ],
    "I-Pro TIGER": [
        "d4:2d:c5",  # I-Pro TIGER
    ],
    "Pelco": [
        "00:04:7d",  # Pelco
    ],
    "Bosch": [
        "00:07:5f",  # Bosch
    ],
    "Avigilon": [
        "00:1f:92",  # Avigilon
        "00:18:85",  # Avigilon
    ],
    "IC Realtime": [
        "00:26:e6",  # IC Realtime
        "00:02:d1",  # IC Realtime (duplicate entry in the original
list)
    ]
}
```

```python
# This should check the camera based on the OUI and categorize it
def mac_check(mac):
    mac_normalized = mac.lower()
    for manufacturer, ouis in camera_ouis.items():
        for camera_oui in ouis:
            if mac_normalized.startswith(camera_oui):
                return manufacturer
    return None


# This should scan for the mac address and categorize the result
def scan_network(target_ip):
    arp_request = ARP(pdst=target_ip)
    broadcast = Ether(dst="ff:ff:ff:ff:ff:ff")
    arp_request_broadcast = broadcast / arp_request

    try:
        answered_list = srp(arp_request_broadcast, timeout=1, verbose=False)[0]  # Reduced timeout to speed things up
    except Exception as e:
        print(f"Error scanning {target_ip}: {e}")
        return []

    devices = []
    for element in answered_list:
        mac = element[1].hwsrc  # Grabs mac address from response
        manufacturer = mac_check(mac)  # Find out which manufacturer the MAC belongs to
        if manufacturer:
            devices.append((element[1].psrc, mac, manufacturer))  # Store IP, MAC, and Manufacturer
    return devices


# The network range it should scan
network_ranges = [
    "192.168.202.1/24",
    "192.168.203.1/24",
    "192.168.204.1/24",
    "192.168.205.1/24"
]

camera_ips_and_macs = {
    "Axis": [],
    "Vivotek": [],
    "Hanwha": [],
    "I-Pro TIGER": [],
    "Pelco": [],
    "Bosch": [],
    "Avigilon": [],
    "IC Realtime": []
}

# Scan each network range
for network in network_ranges:
    print(f"Scanning network range: {network}")  # Debugging
message to track progress
    ip_range = ipaddress.IPv4Network(network, strict=False)

    # Iterate through each IP in the current network range
    for ip in tqdm(ip_range.hosts(), desc=f"Scanning
{network}", unit="IP"):
        devices = scan_network(str(ip))
        for device_ip, device_mac, manufacturer in devices:
            if manufacturer:

camera_ips_and_macs[manufacturer].append((device_ip,
device_mac))

# Write found IPs to respective files
for manufacturer, devices in camera_ips_and_macs.items():
    if devices:
        with open(f"{manufacturer}Cameraips.txt", "w") as file:
            file.write("IP ADDR\t\t\tMAC ADDR\n")
            for ip, mac in devices:
                file.write(f"{ip}\t{mac}\n")
            print(f"{manufacturer} cameras found and saved to
{manufacturer}Cameraips.txt")
    else:
        print(f"No {manufacturer} cameras found...")
```