NODIG:
1. Ubuntu 22.04 LTS: http://ftp.belnet.be/ubuntu-releases/22.04/,
sudo apt-get install nmap wireshark tcpdump python3-scapy python3-pycryptodome ipython3 python3-graphviz imagemagick. Voorzie een Host-Only adapter en een NAT adapter!
2. Slitaz: http://mirror1.slitaz.org/iso/4.0/slitaz-4.0.iso (of neem een extra ubuntu)

REF1: https://readthedocs.org/projects/scapy/downloads/pdf/latest/
REF2: https://theitgeekchronicles.files.wordpress.com/2012/05/scapyguide1.pdf

OPGAVE:
Opgelet! Alle oefeningen moeten werken in python v3

1. SNIFFING

Open een editor en schrijf script 01_sniff.py3:

```
#!/usr/bin/env python3
#############################################################################
__author__      = "jan.celis@kdg.be"
__description__ = "scapy sniffer, sniffs 10 packets"
__license__     = "GPLv3 https://www.gnu.org/licenses/gpl-3.0.nl.html"
__arguments__   = "none"
__version__     = "03-12-2019"
__requires__    = "sudo apt install tcpdump python3-scapy"
#############################################################################

from scapy.all import sniff
about=     "##### "+  __description__ + " #####"
print(about)
packets=sniff(count=10)
packets.summary()
```

Maak het bestand executable en start het op als root gebruiker:

```
chmod +x 01_sniff.py3
sudo ./01_sniff.py3
```

Als er wat netwerkcommunicatie is, zie je een samenvatting van de 10 volgende pakketten die aankomen of vertrekken.

Ideaal kijkt jouw script na of het als rootgebruiker opgestart is.
Verder kan je fouten opvangen met een try/except structuur. Probeer script 01b_sniff.py3 uit:

```
#!/usr/bin/env python3
from scapy.all import sniff
from os import getuid
if getuid() != 0 :
   print("Warning: Starting as non root user!")
try:
   packets=sniff(count=10)
   packets.summary()
except:
   print("Error: Unable to sniff packets, try using sudo.")
   exit(1)     # exit code 1 is gefaald
```

```
./01_sniff.py3
Warning: Starting as non root user!

Error: Unable to sniff packets, try using sudo.
```

2. VERZENDEN PAKKETTEN

Pakketten kan je versturen met de functies send() en sendp(). Send verstuurt pakketten op laag 3, Sendp verstuurt pakketten op laag 2. Schrijf het programma 02_send.py3:

```
#!/usr/bin/env python3
from scapy.all import *
send(IP(dst="127.0.0.1")/ICMP())
sendp(Ether()/IP(dst="127.0.0.1",ttl=(1,4)))
```

Het programma stuurt een ICMP pakket op laag 4 (ICMP) en laag 3 (IP).
Daarna stuurt het programma 4 pakketten op laag 3 (IP) en laag 2 (Ether).

Je krijgt bij uitvoer enkel te zien dat er eerst 1 en daarna 4 pakketten verzonden worden.

3. ZENDEN EN ONTVANGEN

Buiten het verzenden van pakketten, weten we graag welke reactie er komt van het andere systeem. Start hiervoor in Virtualbox of VMware een Live versie op van het mini Linux systeem Slitaz. Kies een Host-Only Adapter als eerste netwerkverbinding. In Slitaz open je een terminal en geef je het commando "ip address" om je adres te zien. In het voorbeeld gebruiken we 192.168.56.101 als adres.

```
tux@slitaz:~$
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 16436 qdisc noqueue state UNKNOWN
   link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
   inet 127.0.0.1/8 scope host lo
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP
   link/ether 08:a2:d6:42:39:df brd ff:ff:ff:ff:ff:ff
   inet 192.168.56,101/24 brd 192.168.1.255 scope global eth0
```

We kunnen met de functie sr1() een antwoord ontvangen of met sr() meerdere antwoorden. Schrijf het script 03_sendreceive.py3:

```
#!/usr/bin/env python3
from scapy.all import *
ans,unans=sr(IP(dst="192.168.56.101",ttl=5)/ICMP())
ans.nsummary()
unans.nsummary()
p=sr1(IP(dst="192.168.56.101")/ICMP()/"XXXXXX")
p.show()
```

Opgelet sr() en sr1() verzenden enkel pakketten op laag 3. Als je pakketten op laag 2 wil verzenden, kan dat met srp() en srp1().

4. SCRIPT MET ARGUMENTEN

Handiger is als je het ip adres als argument kan meegeven met je script. Schrijf 04_sr_arg.py3 met volgende inhoud:

```
#!/usr/bin/env python3
import sys
from scapy.all import sr1,IP,ICMP
p=sr1(IP(dst=sys.argv[1])/ICMP(),timeout=1)
if p:
    p.show()
```

Start het programma als volgt:

```
sudo ./04_sr_arg.py3 192.168.56.101
```

5. ARP PAKKET

Je snift 1 ARP pakket met de functie sniff(). Om enkel een ARP pakket te krijgen gebruik je de optie filter=arp. Verder gebruik je de optie prn=analyze. De optie prn start een functie op, telkens dat hij een pakket vindt dat overeenkomt.

```
#!/usr/bin/env python3
from scapy.all import *
def analyze(p):
    p.show()
packet=sniff(filter="arp",prn=analyze,count=1)
print(packet)
```

```
sudo ./05_sniff_arp.py3
```

Start nu vanuit slitaz een ping op naar een adres in dezelfde range. Hierdoor zal slitaz een ARP pakket moeten sturen om te vragen wat het hardware adres (MAC) is van het ip adres.

```
tux@slitaz:~$ ping -c 1 192.168.56.200
```

De sniffer stopt en geeft het ARP pakket weer:

```
###[ Ethernet ]###
 dst       = ff:ff:ff:ff:ff:ff
 src       = 08:00:27:37:94:a2
 type      = ARP
###[ ARP ]###
    hwtype    = 0x1
    ptype     = IPv4
    hwlen     = 6
    plen      = 4
    op        = who-has
    hwsrc     = 08:00:27:37:94:a2
    psrc      = 192.168.56.101
    hwdst     = 00:00:00:00:00:00
    pdst      = 192.168.56.200
<Sniffed: TCP:0 UDP:0 ICMP:0 Other:1>
```

Wil je de output nog meer filteren bijvoorbeeld enkel de velden psrc en pdst van het ARP pakket, dan kan dat met de sprintf functie (familie van sprintf in de taal C).

```
print(p.sprintf("ARP SRC adres %ARP.psrc% DST adres %ARP.pdst%"))
```

Probeer dit uit!

6. PING RANGE

Voorbeeld met een argument. We sturen een ICMP pakket (default "echo") en bekijken enkel wie antwoord op de ping met een echo-reply. De opmaak van de output doen we in HTML formaat:

```
#!/usr/bin/env python3
from scapy.all import sniff
import sys
import logging                      # disable Warnings
logging.getLogger("scapy.runtime").setLevel(logging.ERROR)


if len(sys.argv) != 2:
    print("Usage: " + sys.argv[0] + " <network>")
    print("       eg:"+ sys.argv[0] + " 192.168.56.0/24 \n")
    print("       eg:"+ sys.argv[0] + " 192.168.56.101-103")
    sys.exit(1)

ans,unans=sr(IP(dst=sys.argv[1])/ICMP(),timeout=1)
print("<html><ol>")
for s,r in ans:
    #r.show()
    print(r.sprintf("<li>received %IP.src% %ICMP.type%</li>\n"))
print("</ol></html>")
```

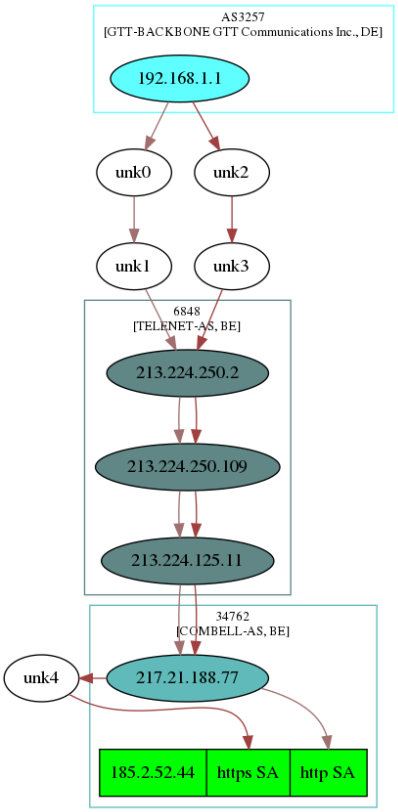
7. GRAFISCHE TRACEROUTE

Wanneer je graphviz installeert, kan je de functie graph() gebruiken voor de functie traceroute. Deze maakt van opgenomen data een grafiek.

```
#!/usr/bin/env python3
#############################################################################
__author__       = "jan.celis@kdg.be"
__description__  = "traceroute with graph output"
__license__      = "GPLv3 https://www.gnu.org/licenses/gpl-3.0.nl.html"
__arguments__    = "None"
__version__      = "05-12-2019"
__output__       = "tracert" + __version__ + ".png"
__requires__     = "sudo apt install nmap python3-crypto ipython3"
__requires__     = "sudo apt install python3-graphviz imagemagick"
#############################################################################

print("##### "+__description__+"#####")

from scapy.all import *
res,unans =traceroute(["www.kdg.be"],dport=[80,443],maxttl=20,retry=-2)
res.graph(target=__output__)
```



8. DHCP REQUEST

Gebruik de referenties om een script te schrijven dat op zoek gaat naar alle DHCP servers in het netwerk door een DHCP request te sturen en te kijken wie daar op antwoordt.
Je kan dit ook testen door een DHCP request te sturen op je Host Only interface. Hierop moet dan wel een actieve DHCP server staan.

9. SCANS

a) XMAS SCAN
Schrijf in python v3 een XMAS scan. Bekijk de scan met wireshark.
Zorg ervoor dat je juist **dezelfde** output krijgt als bij een **nmap** XMAS scan.

-Datum/tijd
-Elapsed time
-MAC Address

b) SYN SCAN
Schrijf in python v3 een SYN scan. Bekijk de scan met wireshark.
Zorg ervoor dat je juist dezelfde output krijgt als bij een nmap SYN scan.

10. SCHRIJVEN VAN PCAP

Schrijf een scapy script dat in python3:
- het eerste argument gebruikt als filename voor de log
- 10 pakketten snift
- deze wegschrijft naar een pcap log
- deze log opent in wireshark