Tested with Windows10

1. Install python (windows automatically launches the installer if you type „python“ into the search field of the task bar)
2. [~~https://pypi.org/project/libpcap/~~](https://pypi.org/project/libpcap/) ~~explains how to install the libpcap for python:   
   python -m pip install --upgrade libpcap~~
3. Wireshark is already installed, this includes the pcap driver, which is necessary for low-level-network-interaction

Vorsicht: Es gibt (mindestens) drei verschiedene Python-Libs für pcap:

* Libpcap
* Pylibpcap (But: only Python2)
* Pypcap (empfohlen von <https://stackoverflow.com/questions/63941109/pcap-open-live-issue>)
* Pcap-ct (<https://pypi.org/project/pcap-ct/>)

python -m pip install --upgrade pcap-ct

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Dieses beißt sich mit der Libpcap-installation, also letztere wieder deinstallieren (python -m pip uninstall libpcap), dann nochmal die pcap-ct installieren, und schießlich noch in libpcap\\_platform\\_\_init\_\_py das fehlende is\_osx = False hinzufügen. (in the meanwhile fixed with <https://github.com/karpierz/pcap-ct/issues/10>)

Path on Windows10 is C:\Users\uwemi\AppData\Local\Packages\PythonSoftwareFoundation.Python.3.10\_qbz5n2kfra8p0\LocalCache\local-packages\Python310\site-packages\pcap

Path on Raspbian is /usr/local/lib/python3.8/site-packages/pcap

Nun funktioniert der Import:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

import pcap

sniffer = pcap.pcap(name=None, promisc=True, immediate=True, timeout\_ms=50)

addr = lambda pkt, offset: '.'.join(str(pkt[i]) for i in range(offset, offset + 4))

for ts, pkt in sniffer:

print('%d\tSRC %-16s\tDST %-16s' % (ts, addr(pkt, sniffer.dloff + 12), addr(pkt, sniffer.dloff + 16)))

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

(control-C to stop)

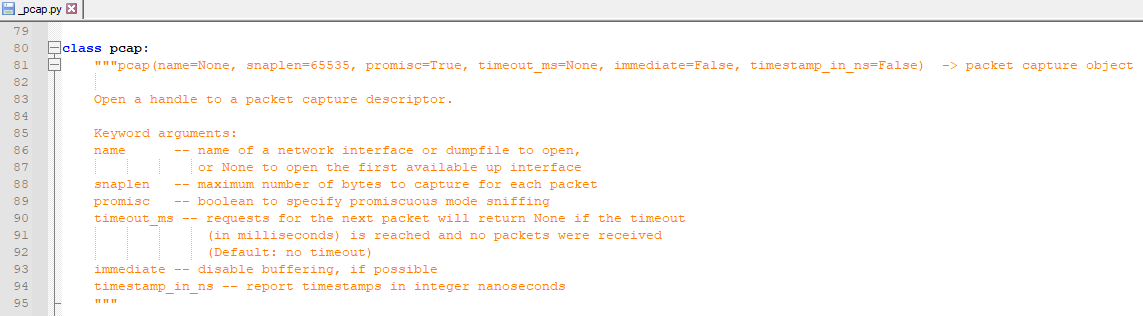
sniffer.close()

ipconfig /all

Ethernet-Adapter Ethernet:

Beschreibung. . . . . . . . . . . : Broadcom NetLink (TM) Gigabit Ethernet

Physische Adresse . . . . . . . . : DC-0E-A1-11-67-08



Aber: Die Funktion pcap.pcap(name="eth0"…) ist nicht in der Lage, das gewünschte Interface zu selektieren. Aus dem eth0 wird einfach die 0 extrahiert, und diese 0 als Index verwendet.

Ein extra instrumentiertes pcap\_ex. Findalldevs findet korrekt die Interfaces:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Die Funktion sortiert die Reihenfolge um, macht aus der vorwärts verketteten Liste eine rückwärts verkettete Liste. (Warum auch immer)

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Wenn wir eth3 als Parameter angeben, wird der dritte Eintrag von hinten genommen, unser gewünschtes Ethernet. Das ist sehr umständlich zu ermitteln…

Zur Verbesserung in die Funktion pcap\_ex.name folgenden Code eingefügt:

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

print("index match at " + str(i) + " dev name=" + str(dev.name) + " dev.description=" + str(dev.description))

Similar to https://github.com/karpierz/pcap-ct/issues/9

In def \_\_init\_\_, add a new attribute self.blNonBlock=False. In def setnonblock set this new attribute according to the argument: self.blNonBlock=nonblock. And in the def \_\_next\_\_, use this new attribute to decide, whether in case of timeout we should just continue (as currently implemented) or we „raise StopIteration“:

If (self.blNonBlock):

Raise StopIteration

Else:

Continue

# Backup

# https://www.tcpdump.org/manpages/pcap\_next\_ex.3pcap.html

# https://pythonhosted.org/pcap\_ylg/pcap.pcap-class.html#next\_ex

# return codes: 1 = pkt, 0 = timeout, -1 = error, -2 = EOF

r, hdr, pkt = self.sniffer.next\_ex()