Network traffic analysis with Python

Nadja Deininger

About me & this talk

- I'm a Python developer with an interest in networking and security
- I’m not an expert (yet)
- This is an introduction to the topic, suitable for complete beginners

Contact:

Twitter: @machine_person

Mail: nadja@ef.gy

Web: machineperson.github.io

How does it work?

Capture network packets using software

- Wireshark
- `tshark` or `tcpdump`

To get all packets on the network you might need to use **port mirroring**.

What is a network packet?

When you send data over a network it will be sent in one or more units called packets.

Each packet contains control information (e.g. source, destination) together with the data you are sending.

Long messages may be split across multiple packets:

- Routers and switches have limited buffer sizes
- Transfer is not 100% reliable, some packets may be dropped

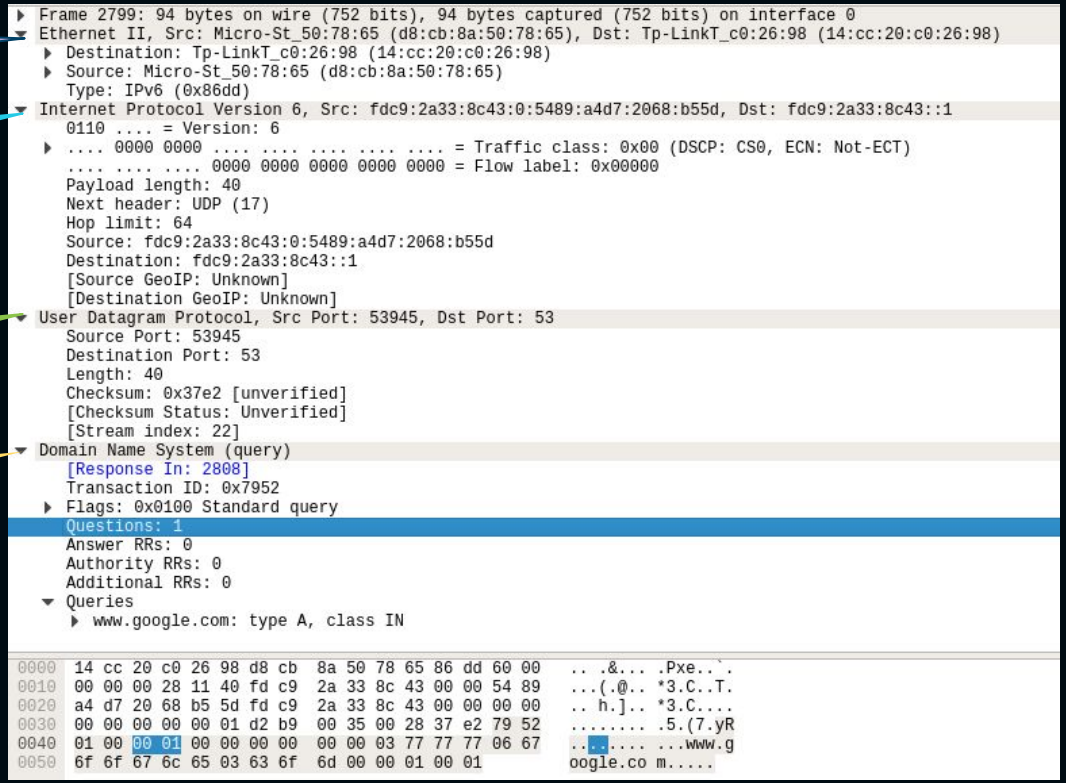
Example packet: DNS request

Ethernet

IPv6

UDP

DNS



How can we do this in Python?

e.g. pyshark library

- Python wrapper for tshark, so tshark must be installed
- Python 3 only
- Uses tshark's parsing capabilities

Other libraries:

- pypcapfile - for analysing capture files
- pypcap - live packet capture, based on libpcap

pyshark example - live capture

```
import pyshark

cap = pyshark.LiveCapture(interface="eth0")

for packet in capture.sniff_continuously(packet_count=5):
    print(packet)
```

pyshark example - existing capture

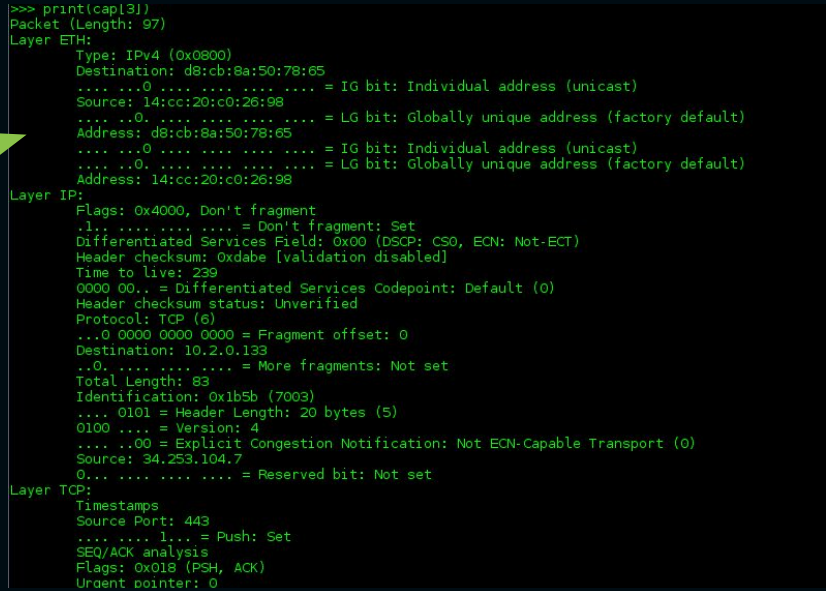
```
import pyshark

cap = pyshark.FileCapture(filename)

packet = cap[0]

print(packet)

# Layer objects with control
information
link_layer = packet.layers[0]
```



pyshark example - finding protocols

```
protonums = {1: "ICMP",
             6: "TCP",
             17: "UDP",
             58: "IPv6-ICMP"}

ip_layer = packet.layers[1]
protocol = None

src_addr = ip_layer.src
dst_addr = ip_layer.dst

if ip_layer.version == "4":
    protocol = ip_layer.proto
elif ip_layer.version == "6":
    protocol = ip_layer.nxt

return {"src_addr": src_addr,
        "dst_addr": dst_addr,
        "protocol": protonums.get(int(protocol), protocol)}
```

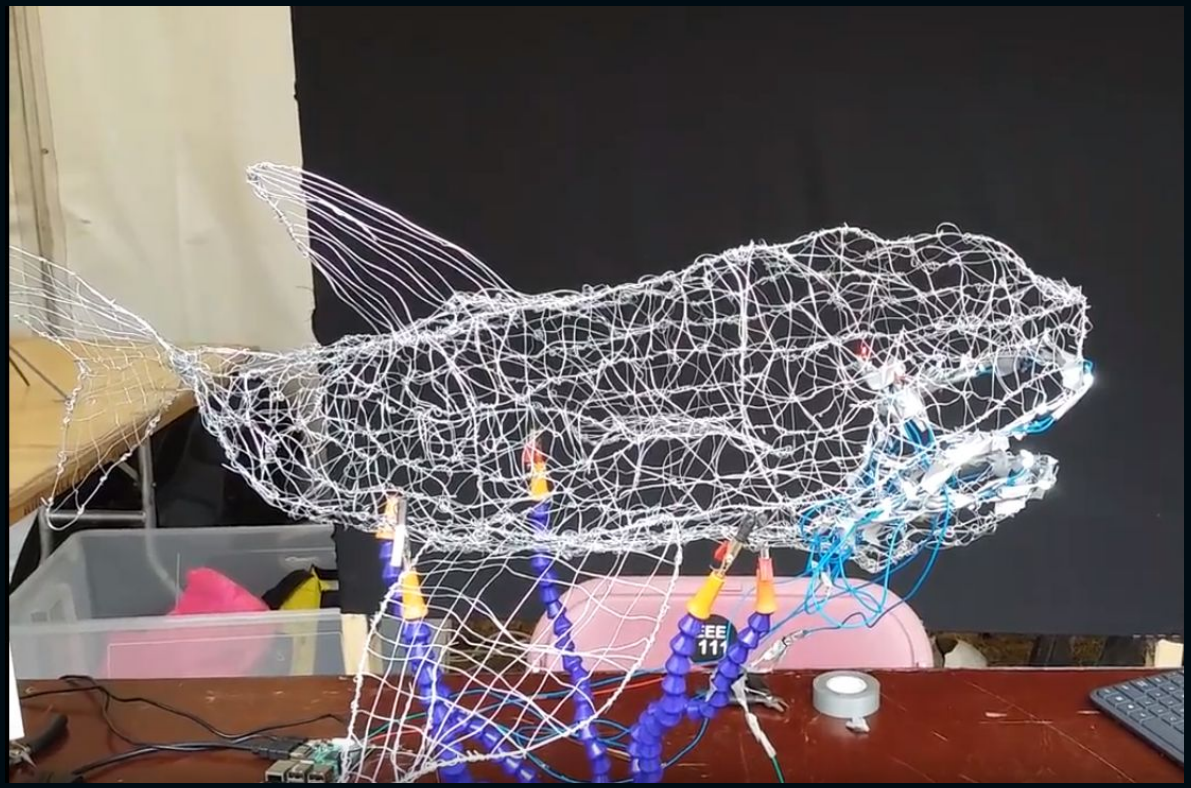
Potential applications

- Live capture with monitoring instrumentation
- Statistics on capture files
- Data visualisation
- ...or just light up some LEDs because I can...

Meet Sharky...

- Wire shark model
- pre-recorded packet capture
- Python program on a Raspberry Pi that interprets the packets
- LEDs

→ Network-based blinkenlights!



Thank you!