SCAPY – SPAß MIT NETZWERKPAKETEN

Michael Estner

AGENDA

Wer bin ich

Motivation

Scapy

Das erste Paket

Sniffing

Integration

Q&A

MICHAEL ESTNER

Senior Software Entwickler

Embedded Linux & Netzwerk

Bachelor: Elektro- & Informationstechnik

Privat: MMA, Wandern, Kochen

MOTIVATION

Warum scapy

Toollandschaft

SCAPY

Einführung

SCAPY EINFÜHRUNG

Scapy Fakten

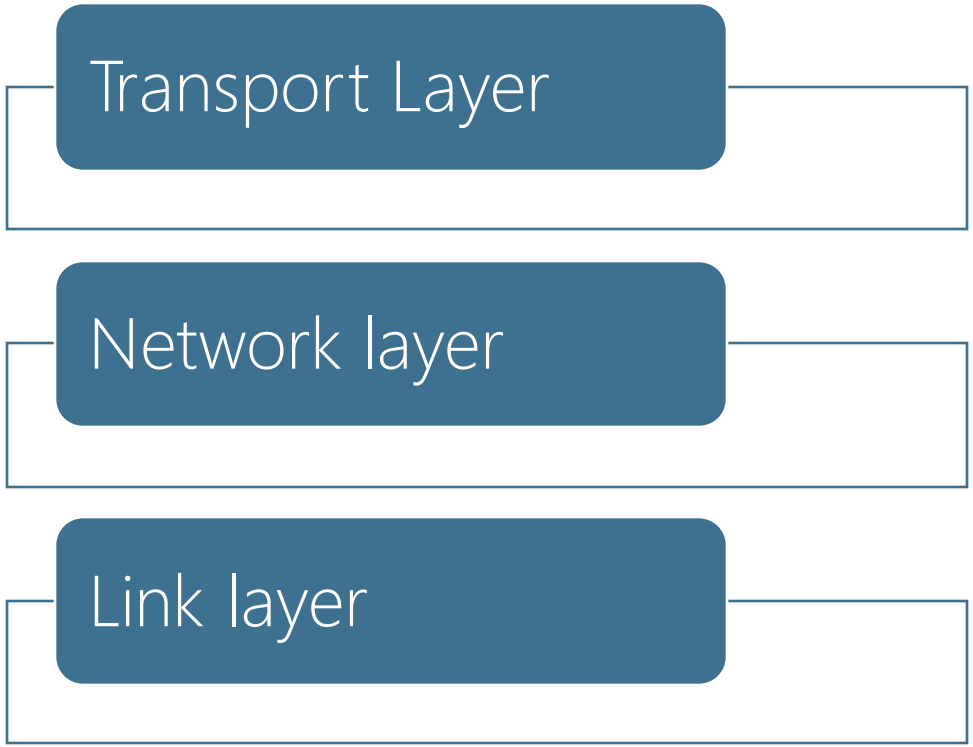
Python basiert

Netzwerkpackete können gebaut, versendet und empfangen werden

Packete sind über mehrere Layer modifizierbar

Schnell und leicht eigene Packete erstellen

Cross Plattform



SCAPY EINFÜHRUNG

Besonderheiten

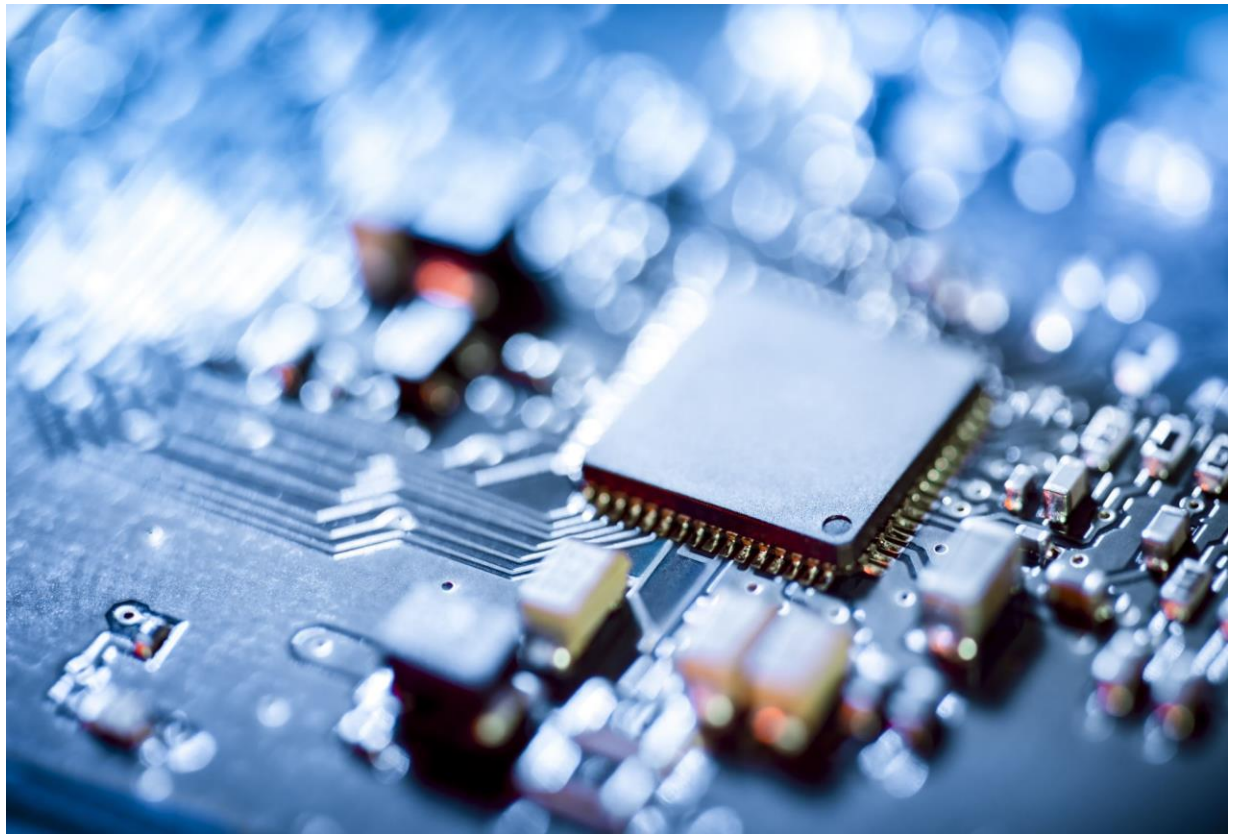
Eigener Netzwerk Stack

-> Interface Liste, routing table, usw.

-> Alles ist konfigurierbar

Interpretiert nicht, es dekodiert

Sudo Rechte werden benötigt



SCAPY EINFÜHRUNG

Installation

```
pip install scapy
```

```
git clone https://github.com/secdev/scapy.git

cd scapy

pip install .
```

SCAPY EINFÜHRUNG

Unterstützte Protokolle

TCP

Ethernet

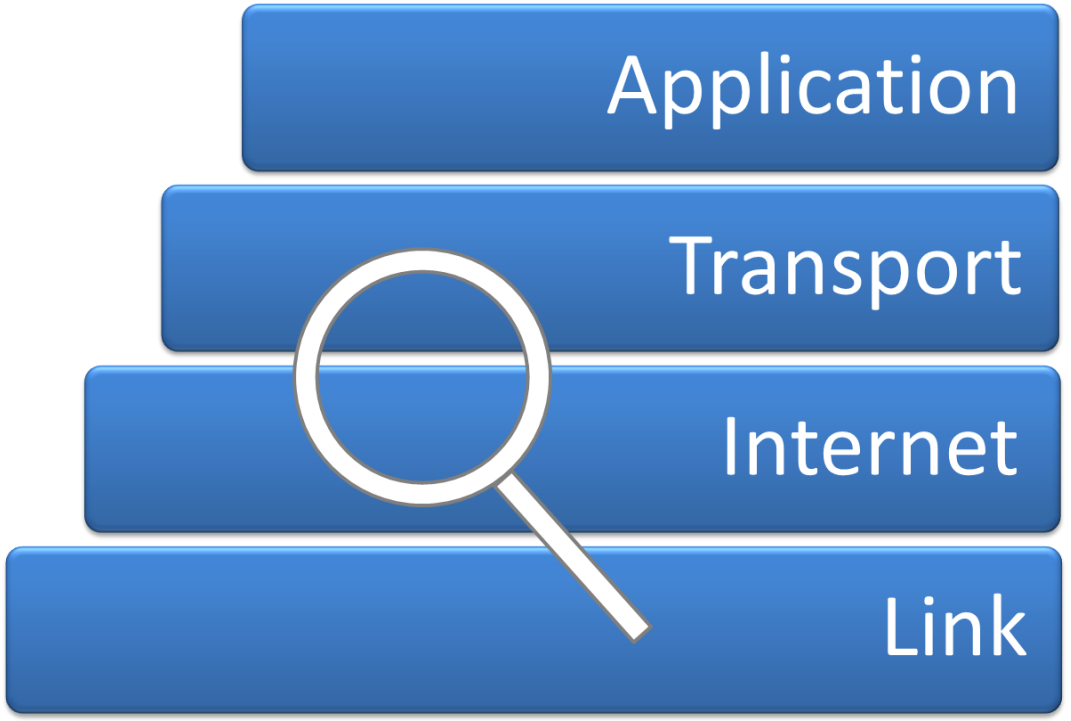
IP

UDP

ICMP

HTTP

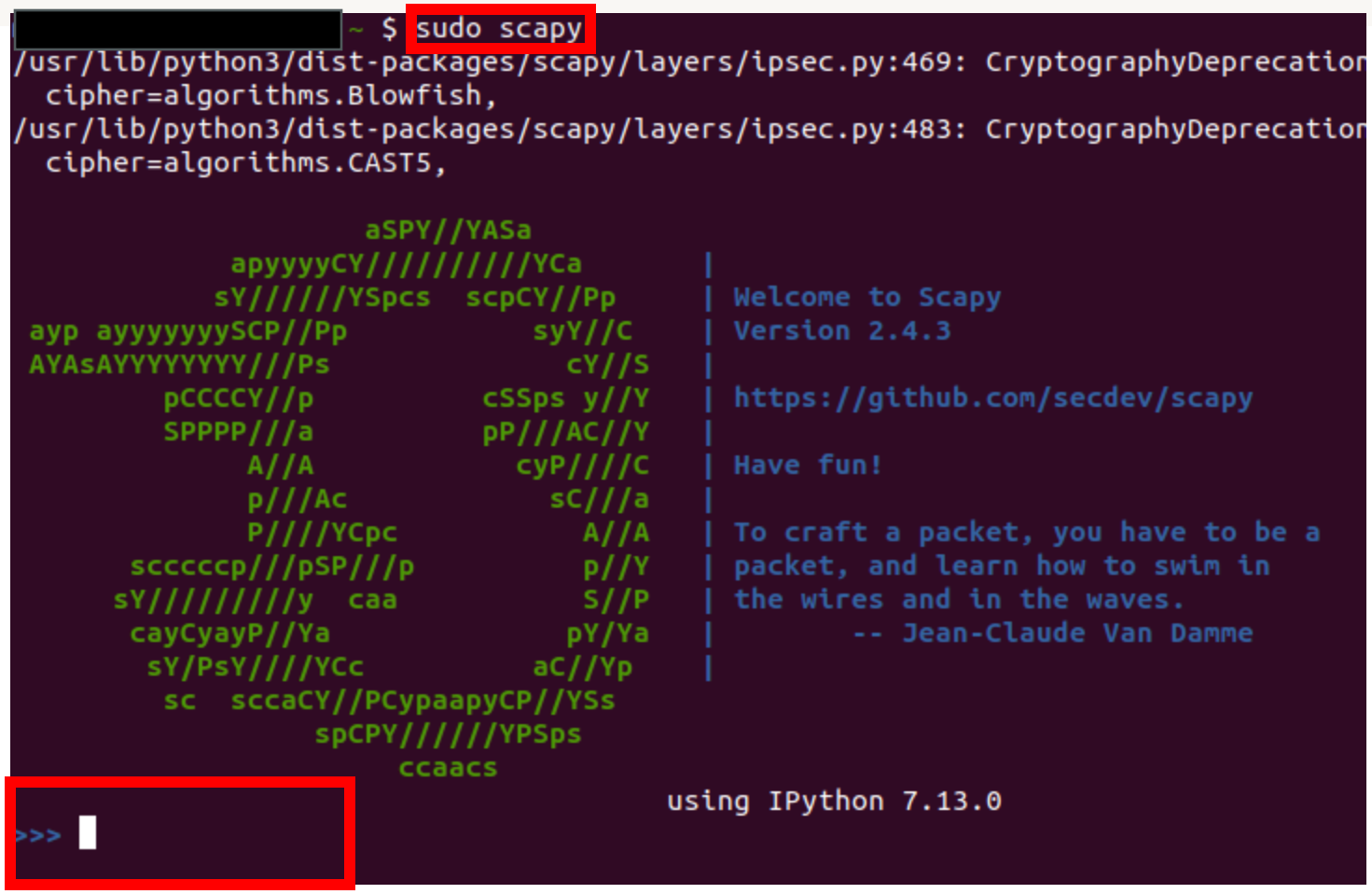
Und viele mehr :)



SCAPY EINFÜHRUNG

Scapy Konsole

Scapy



Scapy stack layers

```
#Layer Prinzip

Ether()/IP()/TCP()



Ether(type=0x0800, dst="ff:ff:ff:ff:fe:ee")

IP(src="192.168.178.2", dst="192.168.178.6")

TCP(dport=80, flags="S")
```

SCAPY EINFÜHRUNG

Scapy stack layer

```
package=Ether(type=0x0800,
dst="ff:ff:ff:ff:fe:ee")/IP(src="192.168.178.2",
dst="192.168.178.6")/TCP(dport=80, flags="S")


package.show()
```

```
###[ Ethernet ]###
  dst= ff:ff:ff:ff:fe:ee
  src= 38:00:25:6d:66:13
  type= IPv4
###[ IP ]###
     version= 4
     ihl= None
     tos= 0x0
     len= None
     id= 1
     flags=
     frag= 0
     ttl= 64
     proto= tcp
     chksum= None
     src= 192.168.178.2
     dst= 192.168.178.6
     \options\
###[ TCP ]###
        sport= ftp_data
        dport= http
        seq= 0
        ack= 0
        dataofs= None
        reserved= 0
        flags= S
        window= 8192
        chksum= None
        urgptr= 0
        options= []
```

DAS ERSTE PAKET

DAS ERSTE PAKET

Scapy TCP

```
package=IP(src="192.168.178.1", dst="192.168.178.5")/TCP(dport=80)

send(package, iface="wlan0")
```

```
>>> send(package, iface="wlan0")
.
Sent 1 packets.
>>>
```

DAS ERSTE PAKET

Scapy ICMP

IPv6()/ICMP(type=8, code=0)

```
###[ IPv6 ]###
  version= 6
  tc= 0
  fl= 0
  plen= None
  nh= No Next Header
  hlim= 64
  src= ::1
  dst= ::1
###[ ICMP ]###
     type= echo-request
     code= 0
     chksum= None
     id= 0x0
     seq= 0x0
```

IPv6()/ICMPv6EchoRequest()

```
###[ IPv6 ]###
  version= 6
  tc= 0
  fl= 0
  plen= None
  nh= ICMPv6
  hlim= 64
  src= ::1
  dst= ::1
###[ ICMPv6 Echo Request ]###
     type= Echo Request
     code= 0
     cksum= None
     id= 0x0
     seq= 0x0
     data= ''
```

Scapy Packete versenden

```
package=IP(src="192.168.178.1", dst="192.168.178.5")/TCP(dport=80)

#Layer3

send(package, iface="eth0")

#Layer2

sendp(package, iface="eth0")

#Send with answer

sr(package, iface="eth0")
```

SNIFFING

Sniffing und Wireshark

SNIFFING

Sniffing

```
send(IP(src="192.168.178.1")/ICMP(), count=4, iface="lo")

#Sniffing mit filter

sniff(filter="icmp and src 192.168.178.1", count=4, iface="lo")
```

```
>>> sniff(filter="icmp and src 192.168.178.1", count=4, iface="lo").show()
0000 Ether / IP / ICMP 192.168.178.1 > 127.0.0.1 echo-request 0
0001 Ether / IP / ICMP 192.168.178.1 > 127.0.0.1 echo-request 0
0002 Ether / IP / ICMP 192.168.178.1 > 127.0.0.1 echo-request 0
0003 Ether / IP / ICMP 192.168.178.1 > 127.0.0.1 echo-request 0
>>>
```

SNIFFING

Sniffing und Code ausführen

```
sniff(iface="eth0",  filter="tcp and port 80",
count="1", prn=test_function)
```

```
def test_function(x)
    if x[TCP].dport == 80:
        print("Port is 80")
    else:
        print("Port is wrong")
```

Daten exportieren

```
# Erstmal ein Packet erstellen ...

packets = IP(src="192.0.2.9", dst=Net("192.0.2.10/30"))/ICMP()

# Öffnen mit Wireshark

wireshark(packets)
```

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.0.2.9	192.0.2.8	ICMP	28	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response
2	0.000000	192.0.2.9	192.0.2.9	ICMP	28	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response
3	0.000000	192.0.2.9	192.0.2.10	ICMP	28	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response
4	0.000000	192.0.2.9	192.0.2.11	ICMP	28	Echo (ping) request id=0x0000, seq=0/0, ttl=64 (no response

```
wrpcap("temp.cap",pkts)
```

SNIFFING

Interfaces

```
#Interface Konfiguration

-> conf.ifaces

#List interfaces

-> get_if_list()

#Routing Tabelle

-> conf.route
```

ADVANCED

Fuzzing & Byte injection

Fuzzing

#Fuzz testing mit Paketen

send(IP(src="192.168.178.1")/fuzz(ICMP()), count=40, iface="lo")

```
>>> sniff(filter="icmp and src 192.168.178.1", count=40, iface="lo").show()
0000 Ether / IP / ICMP 192.168.178.1 > 127.0.0.1 150 12
0001 Ether / IP / ICMP 192.168.178.1 > 127.0.0.1 150 12
0002 Ether / IP / ICMP 192.168.178.1 > 127.0.0.1 echo-reply 172
0003 Ether / IP / ICMP 192.168.178.1 > 127.0.0.1 echo-reply 172
0004 Ether / IP / ICMP 192.168.178.1 > 127.0.0.1 46 247
0005 Ether / IP / ICMP 192.168.178.1 > 127.0.0.1 46 247
0006 Ether / IP / ICMP 192.168.178.1 > 127.0.0.1 29 103
0007 Ether / IP / ICMP 192.168.178.1 > 127.0.0.1 29 103
0008 Ether / IP / ICMP 192.168.178.1 > 127.0.0.1 226 198
0009 Ether / IP / ICMP 192.168.178.1 > 127.0.0.1 226 198
0010 Ether / IP / ICMP 192.168.178.1 > 127.0.0.1 177 134
0011 Ether / IP / ICMP 192.168.178.1 > 127.0.0.1 177 134
0012 Ether / IP / ICMP 192.168.178.1 > 127.0.0.1 64 43
0013 Ether / IP / ICMP 192.168.178.1 > 127.0.0.1 64 43
0014 Ether / IP / ICMP 192.168.178.1 > 127.0.0.1 107 113
0015 Ether / IP / ICMP 192.168.178.1 > 127.0.0.1 107 113
```

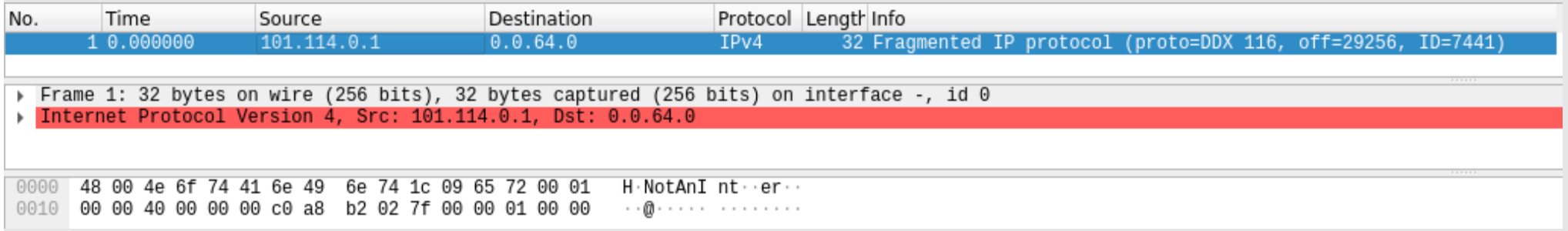
Byte Injection

```
package = IP(len=RawVal(b"NotAnInteger"), src="127.0.0.1")

bytes(package)

b'H\x00NotAnInt\x0f\xb3er\x00\x01\x00\x00@\x00\x00\x00\x7f\x00\x00\x01\x7f\x00\x00\x01\x00\x00'

Wireshark(package)
```



INTEGRATION

Scapy in Pyhton Skript verwenden

```
from scapy.all import *

# Erstelle ein Paket

packet = IP(src="192.168.1.100", dst="192.168.1.101")/TCP(sport=12345, dport=80, flags="S")

# Sende das Paket

send(packet, iface="eth0")
```

BEISPIELE

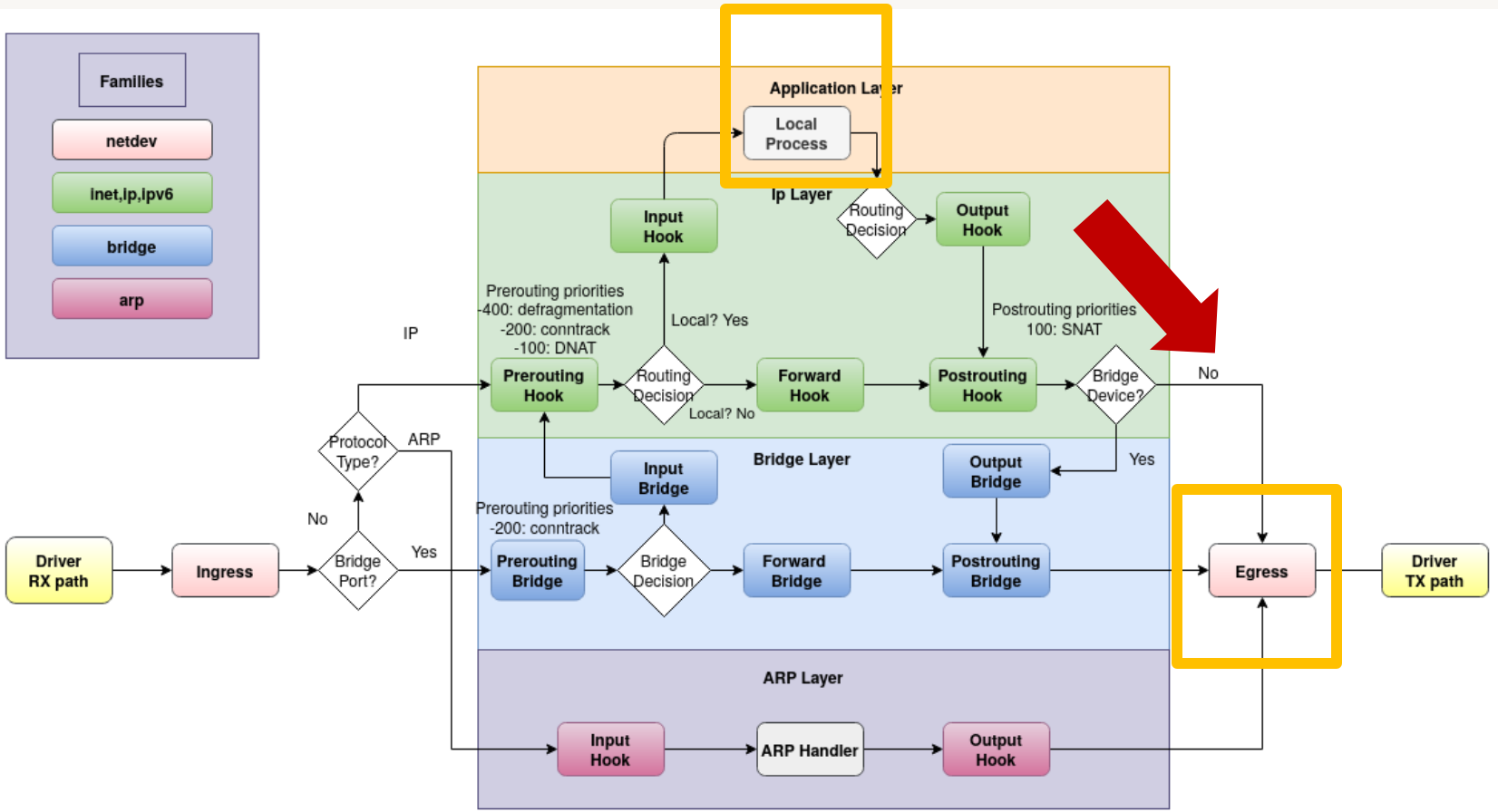
BEISPIELE

Anwendungen

- Testen von Firewall Regeln
- Testen der Traffic Control
- Applikationen testen

BEISPIELE

- Egressing



https://wiki.nftables.org/wiki-nftables/index.php/Netfilter_hooks 19.12.2023 15:18

BEISPIELE

Iptables –t nat –A OUTPUT –d 192.168.7.1/32 -o eth0 –p tcp --dport 100 –j REDIRECT --to-ports 60001

Craft a fitting package:

```
pkt = TCP(dport=100)
```

Send it:

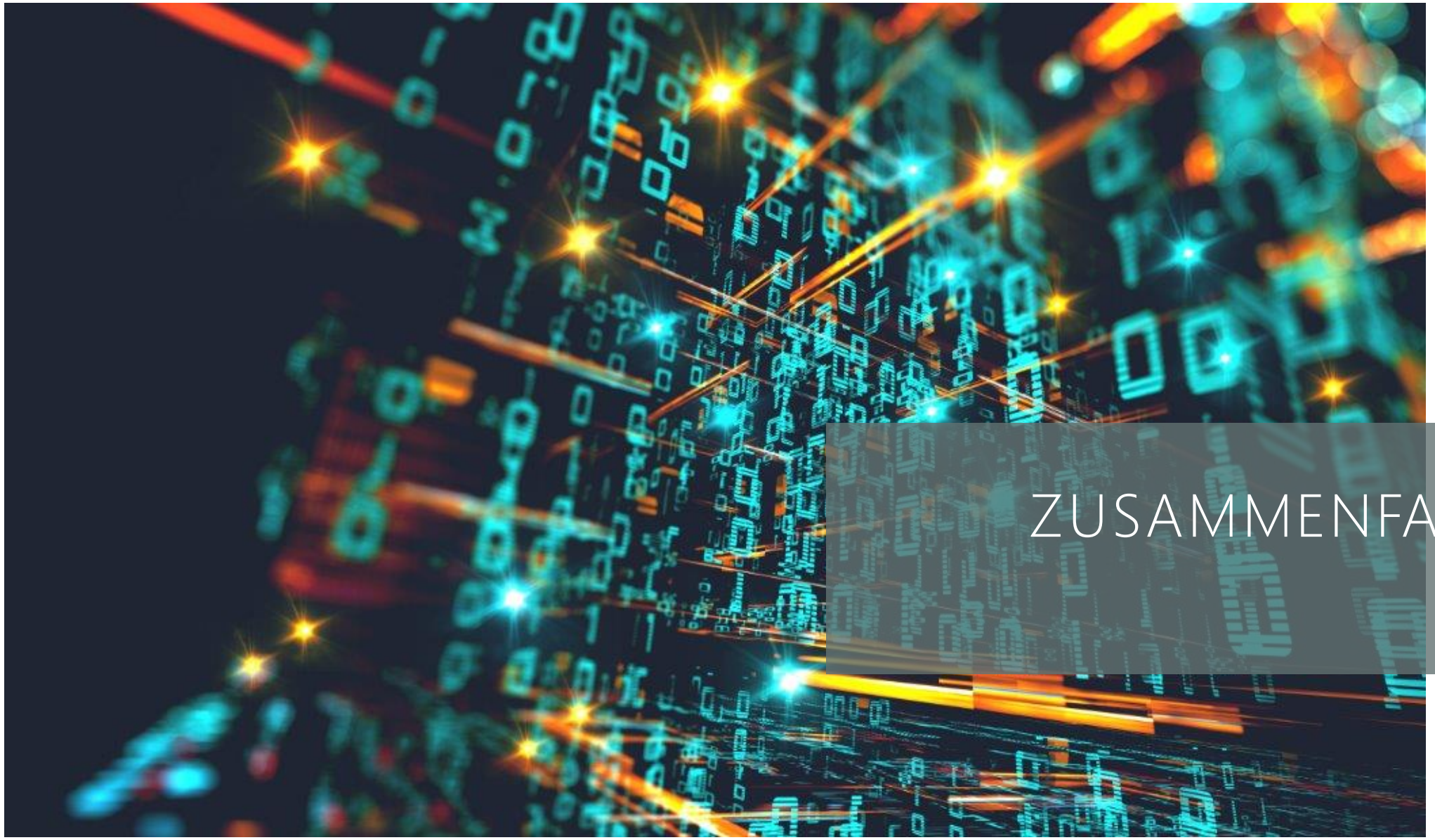
```
s.setsockopt(socket.SOL_SOCKET, 25, str("eth0"))
s.bind(('192.168.7.2', 0))
s.sendto(bytes(pkt), ("192.168.7.1", 0))
```

Sniff for it:

```
sniff(iface="tap0", filter="tcp and port 60001",
count="1", prn=packet3_check)
```

```
def packet3_check(x)
    if x[TCP].dport == 60001:
        print("accepted by FW")
    else:
        print("rejected by FW")
```

FOSDEM Talk



ZUSAMMENFASSUNG

ZUSAMMENFASSUNG

SCAPY EINSTIEG

PAKETE BAUEN

SNIFFING

FUZZING

DATEN HANDLING

INTEGRATION

BEISPIELE

Was noch möglich ist

- Neue/Eigene Protokolle hinzufügen
- Eigene Tools entwickeln
- Scapy mit Addons erweitern
- Und noch vieles mehr :)

WEITERFÜHRENDE LINKS

- Webseite
- Usage
- Oneliners
- Eigenes Protokoll hinzufügen
- Scapy Entwicklung

FRAGEN?

Michael Estner

michaelestner@web.de