



EDUCATION AND TRAINING

## CyberSecPro Training

We are creating cutting-edge education and training to advance competencies and professionalism in EU cybersecurity.

OUR VISION

Next level cybersecurity education and training



# Network Protection for Energy Control Systems

## CSP004\_C\_E

PRESENTATION BY:  
DR. STEFAN SCHAUER  
DR. ABDELKADER SHAABAN  
AIT AUSTRIAN INSTITUTE OF TECHNOLOGY



# Network Protection for Energy Control Systems

**These slides outline the essential offensive tools that will be used in this course.**

These tools are intended for use within this course to demonstrate how different tools can be employed for various cyberattack activities and address existing security weaknesses to avoid or mitigate related cyber risks. Therefore, all these practical activities are solely intended for educational purposes ONLY and not for any other malicious or unauthorized activities.

# GNS3 Simulator

# GNS3 Simulator

## Field Devices

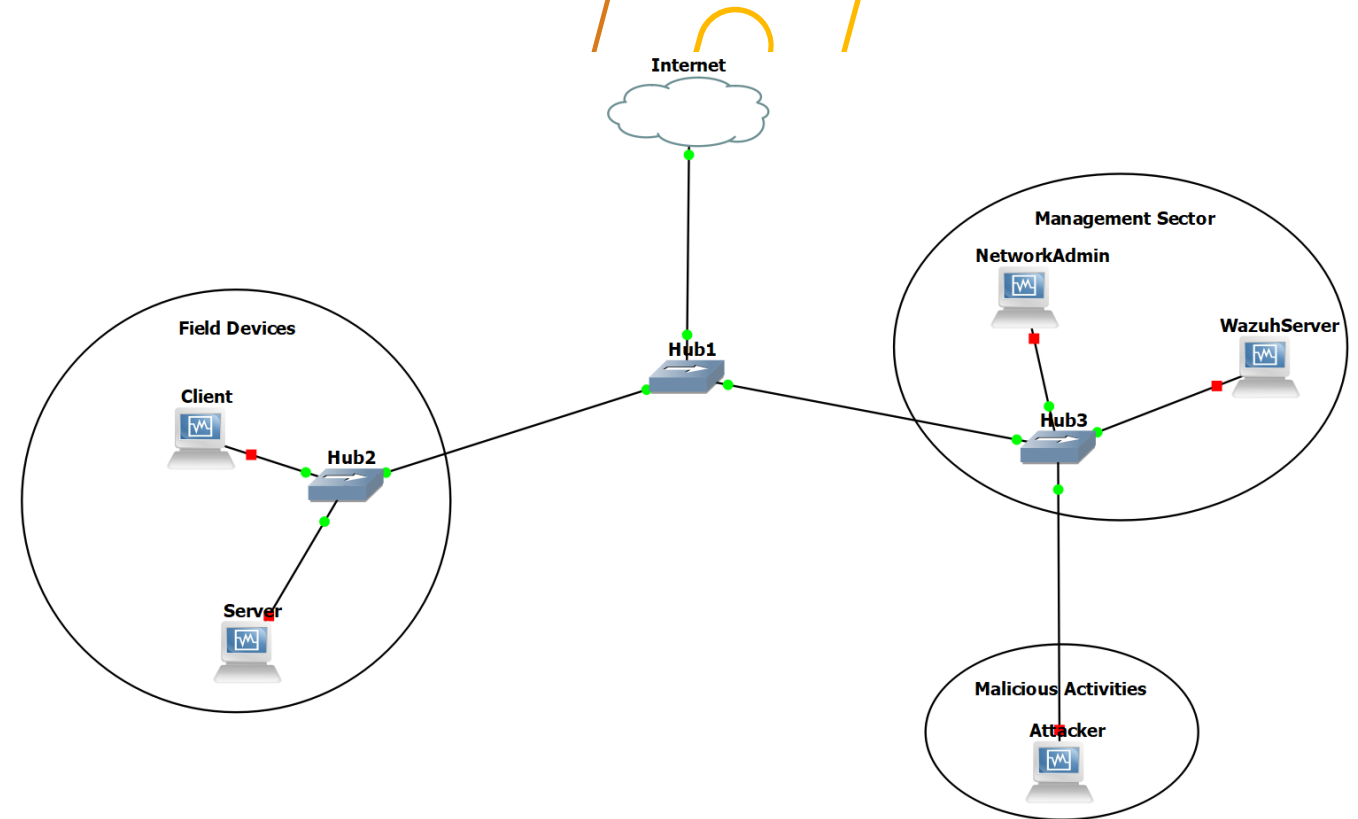
- Raspberry Pi running lightweight Linux version (Client and Server) for transmitting data over ModBus communication protocols using the pyModbusTCP.

## Management Sector

- Network administration device for monitoring network traffic using Kali Linux.
- Wazuh Server: Details to be addressed later.

## Malicious Activities

- Kali Linux will be used to simulate various malicious activities targeting devices within this closed network.



# Offensive Tools

## Wireshark

# Wireshark Sniffing & Packets Analysis

- **Wireshark** is a network protocol analyzer designed to help network administrators keep track of what is **happening** in their network.
- When you **become** an **MITM**, the **Wireshark tool** can be used to **sniff** and **analyze** traffic **sent** and **received** by the targets.

**Green** – TCP packets

**Dark blue** - is DNS packet

**Black** – TCP packets that have an issue

No.	Time	Source	Destination	Protocol	Len	Info
7732	74.186283530	192.168.122.21	20.82.9.214	TCP	60	49852 → 443 [ACK] Seq=587 Ack=6363 Win=64240 Len=0
7736	74.188082949	192.168.122.21	20.82.9.214	TCP	1	49852 → 443 [ACK] Seq=1168 Ack=6363 Win=64240 Len=1460 [TCP s
7737	74.188420888	20.82.9.214	192.168.122.21	TCP	60	443 → 49852 [ACK] Seq=6363 Ack=667 Win=65535 Len=0
7739	74.188833908	20.82.9.214	192.168.122.21	TCP	60	443 → 49852 [ACK] Seq=6363 Ack=759 Win=65535 Len=0
7740	74.188943871	20.82.9.214	192.168.122.21	TCP	60	443 → 49852 [ACK] Seq=6363 Ack=1168 Win=65535 Len=0
7741	74.189329824	20.82.9.214	192.168.122.21	TCP	60	443 → 49852 [ACK] Seq=6363 Ack=2628 Win=65535 Len=0
7742	74.189334380	20.82.9.214	192.168.122.21	TCP	60	443 → 49852 [ACK] Seq=6363 Ack=2893 Win=65535 Len=0
7745	74.228373590	20.82.9.214	192.168.122.21	TCP	60	443 → 49852 [ACK] Seq=7062 Ack=2924 Win=65535 Len=0
7748	74.243534830	44.238.29.244	192.168.122.21	TCP	71	80 → 49851 [PSH, ACK] Seq=6587 Ack=1484 Win=65535 Len=17 [TCP
7749	74.244286484	192.168.122.21	20.82.9.214	TCP	60	49852 → 443 [ACK] Seq=2924 Ack=8222 Win=64240 Len=0
7750	74.255309779	44.238.29.244	192.168.122.21	TCP	2	80 → 49851 [ACK] Seq=6604 Ack=1484 Win=65535 Len=2920 [TCP se
7752	74.256261935	192.168.122.21	44.238.29.244	TCP	60	49851 → 80 [ACK] Seq=1484 Ack=9524 Win=64240 Len=0
7753	74.256657628	192.168.122.21	44.238.29.244	TCP	60	49851 → 80 [ACK] Seq=1484 Ack=9962 Win=63802 Len=0
7765	74.358726695	192.168.122.21	20.74.47.205	TCP	66	49853 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK
7769	74.374809724	192.168.122.21	204.79.197.239	TCP	66	49854 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256 SACK
7772	74.375126719	192.168.122.21	20.82.9.214	TCP	1	[TCP Out-Of-Order] 49852 → 443 [ACK] Seq=3158 Ack=8222 Win=64
7773	74.375754969	20.82.9.214	192.168.122.21	TCP	60	443 → 49852 [ACK] Seq=8222 Ack=3158 Win=65535 Len=0
7774	74.375760572	20.82.9.214	192.168.122.21	TCP	60	443 → 49852 [ACK] Seq=8222 Ack=4618 Win=65535 Len=0
7775	74.375763939	20.82.9.214	192.168.122.21	TCP	60	443 → 49852 [ACK] Seq=8222 Ack=5292 Win=65535 Len=0
7778	74.395137210	204.79.197.239	192.168.122.21	TCP	60	443 → 49854 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
7779	74.395151392	20.74.47.205	192.168.122.21	TCP	60	443 → 49853 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
7780	74.396822080	192.168.122.21	204.79.197.239	TCP	60	49854 → 443 [ACK] Seq=1 Ack=1 Win=64240 Len=0
7781	74.397202928	192.168.122.21	20.74.47.205	TCP	60	49853 → 443 [ACK] Seq=1 Ack=1 Win=64240 Len=0
7784	74.398034395	20.74.47.205	192.168.122.21	TCP	60	443 → 49853 [ACK] Seq=1 Ack=596 Win=65535 Len=0

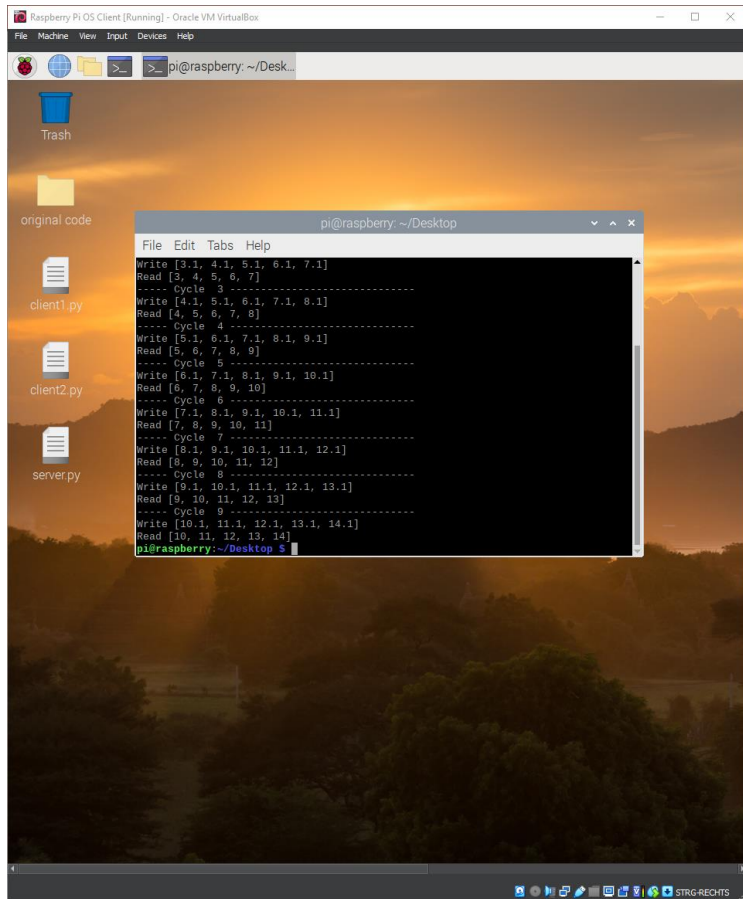
Frame 37: 92 bytes on wire (736 bits), 92 bytes captured (736 bits) on interface eth0, id 0  
 Ethernet II, Src: PcsCompu\_27:29:8c (08:00:27:27:29:8c), Dst: PcsCompu\_ab:32:d8 (08:00:27:ab:32:d8)  
 Internet Protocol Version 4, Src: 192.168.122.27, Dst: 192.168.122.21  
 User Datagram Protocol, Src Port: 44822, Dst Port: 137  
 NetBIOS Name Service

0000 08 00 27 ab 32 d8 08 00 27 27 29 8c 08 00 45 00 ... 2 ... ) ... E  
 0010 00 4e 2c e5 40 00 40 11 98 38 c0 a8 7a 1b c0 a8 .N, @ @ 8 - z ...  
 0020 7a 15 af 16 00 89 00 3a 75 cd 82 28 00 00 00 01 z : : : : u : ( : :  
 0030 00 00 00 00 00 00 20 43 4b 41 41 41 41 41 41 41 : : : : C K A A A A A A  
 0040 41 41 41 41 41 41 41 41 41 41 41 41 41 41 41 A  
 0050 41 41 41 41 41 41 41 00 00 21 00 01 A A A A A A A A : : ! : :

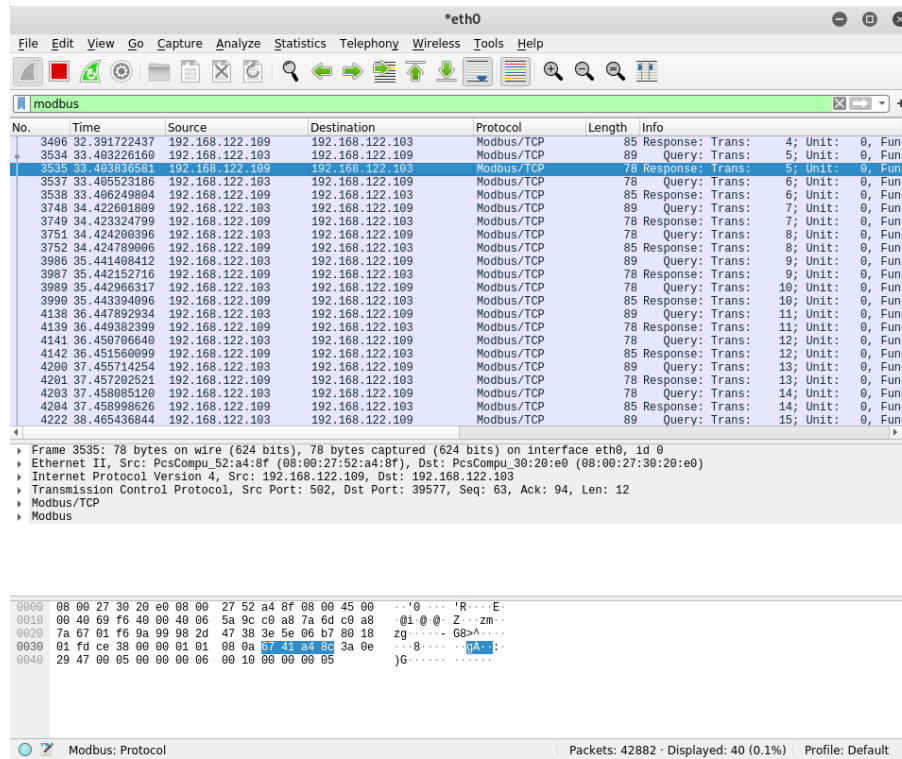
# Wireshark – Client/Server Example

- In the Server/Client example, we have two Linux machines: the client sending data to the server through the **ModbusTCP** protocol.
- When the attacker is a **Man-in-the-Middle**, Wireshark tool can be used for capturing all data transmitting, and other actions could be applied, such as:
  - Filtering collected data (**Modbus**)
  - Analysis traffic

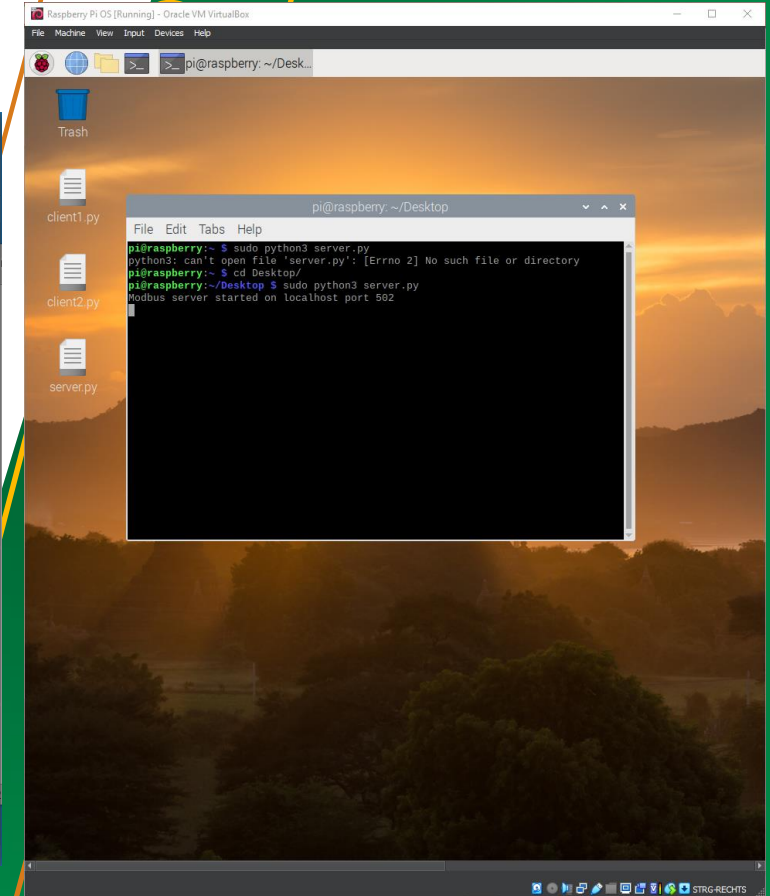
# Wireshark - Client/Server Example



Client



Kali



Server

# Wireshark - Client/Server Example

Wireshark · Packet 276 · eth0

```
▶ Frame 276: 89 bytes on wire (712 bits), 89 bytes captured (712 bits) on interface eth0, id 0
▶ Ethernet II, Src: PcsCompu_30:20:e0 (08:00:27:30:20:e0), Dst: PcsCompu_52:a4:8f (08:00:27:52:a4:8f)
▶ Internet Protocol Version 4, Src: 192.168.122.103, Dst: 192.168.122.109
▶ Transmission Control Protocol, Src Port: 39259, Dst Port: 502, Seq: 1, Ack: 1, Len: 23
  Source Port: 39259
  Destination Port: 502
  [Stream index: 0]
  [TCP Segment Len: 23]
  Sequence number: 1 (relative sequence number)
  Sequence number (raw): 2456416165
  [Next sequence number: 24 (relative sequence number)]
  Acknowledgment number: 1 (relative ack number)
  Acknowledgment number (raw): 4188012495
  1000 .... = Header Length: 32 bytes (8)
  ▾ Flags: 0x018 (PSH, ACK)
    000. .... = Reserved: Not set
    ....0 ... = Nonce: Not set
    .... 0... = Congestion Window Reduced (CWR): Not set
    .... .0.. = ECN-Echo: Not set
    .... ..0. = Urgent: Not set
    .... ...1 = Acknowledgment: Set
    .... .... 1... = Push: Set
    .... .... .0.. = Reset: Not set
    .... .... ..0. = Syn: Not set
    .... .... ...0 = Fin: Not set
    [TCP Flags: .....AP...]
  Window size value: 502
  [Calculated window size: 64256]
  [Window size scaling factor: 128]
  Checksum: 0x1a4f [unverified]
  [Checksum Status: Unverified]
  Urgent pointer: 0
  ▾ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▾ TCP Option - No-Operation (NOP)
  0000 08 00 27 52 a4 8f 08 00 27 30 20 e0 08 00 45 00  ..'R... '0...E.
  0010 00 4b e2 05 40 00 40 06 e2 81 c0 a8 7a 67 c0 a8  ..K..@.. ..zg..
  0020 7a 6d 99 5b 01 f6 92 69 ef a5 f9 9f ff cf 80 18  zm.[...i .....
  0030 01 f6 1a 4f 00 00 01 01 08 0a 3a 1e 4b c4 67 51  ...0..... :K.gQ
  0040 c7 07 00 01 00 00 00 11 00 10 00 00 00 05 0a 00  .....
  0050 01 00 02 00 03 00 04 00 05  .....
```

Source and Destination IP

Source and Destination Ports

# Wireshark - Client/Server Example

Wireshark · Packet 276 · eth0

[iRTT: 0.001079015 seconds]  
[Bytes in flight: 23]  
[Bytes sent since last PSH flag: 23]

- Timestamps
  - Time since first frame in this TCP stream: 0.001085504 seconds
  - Time since previous frame in this TCP stream: 0.000006489 seconds
- TCP payload (23 bytes)
  - PDU Size: 23
- Modbus/TCP
  - Transaction Identifier: 1
  - Protocol Identifier: 0
  - Length: 17
  - Unit Identifier: 0
- Modbus
  - .001 0000 = Function Code: Write Multiple Registers (16)
  - Reference Number: 0
  - Word Count: 5
  - Byte Count: 10
  - Register 0 (UINT16): 1
    - Register Number: 0
    - Register Value (UINT16): 1
  - Register 1 (UINT16): 2
    - Register Number: 1
    - Register Value (UINT16): 2
  - Register 2 (UINT16): 3
    - Register Number: 2
    - Register Value (UINT16): 3
  - Register 3 (UINT16): 4
    - Register Number: 3
    - Register Value (UINT16): 4
  - Register 4 (UINT16): 5
    - Register Number: 4
    - Register Value (UINT16): 5

0000 08 00 27 52 a4 8f 08 00 27 30 20 e0 08 00 45 00 ..'R...'.0..E.  
0010 00 4b e2 05 40 00 40 06 e2 81 c0 a8 7a 67 c0 a8 .K..@.@...zg..  
0020 7a 6d 99 5b 01 f6 92 69 ef a5 f9 9f ff cf 80 18 zm[...]i.....  
0030 01 f6 1a 4f 00 00 01 01 08 0a 3a 1e 4b c4 67 51 ...0...:K.gQ  
0040 c7 07 00 01 00 00 00 11 00 10 00 00 00 05 0a 00 .....  
0050 01 00 02 00 03 00 04 00 05 .....

Close Help

# Offensive Tools

## Scapy

# SCAPY: Create/Send a Packet

## Create packets on Scapy

- Start Scapy, and a Python command line within Scapy will open

```

Scapy v2.4.3
Scapy v2.4.3 83x25
root@kali:~# scapy
INFO: Can't import PyX. Won't be able to use psdump() or pdfdump().
WARNING: No route found for IPv6 destination :: (no default route?)

      aSPY//YASa
      apyyyyCY////////YCa
      sY////////YSpcs  scpCY//Pp
ayp ayyyyyySCP//Pp      syY//C
AYAsAYYYYYYYY//Ps      cY//S
pCCCCY//p      cSSps y//Y
SPPPP//a      pP//AC//Y
A//A      cyP////C
p//Ac      sC//a
P////YCpc      A//A
scccccp///pSP///p      p//Y
sY/////////y caa      S//P
cayCyayP//Ya      pY/Ya
sY/PsY////YCc      aC//Yp
sc sccaCY//PCyPaapyCP//YSs
spCPY////////YPSps
ccaacs

Welcome to Scapy
Version 2.4.3
https://github.com/secdev/scapy
Have fun!
To craft a packet, you have to be a
packet, and learn how to swim in
the wires and in the waves.
-- Jean-Claude Van Damme

using IPython 7.12.0
>>>

```

## Use Wireshark to catch the ICMP message

- Assign the packet to a variable, say P.
  - >> **P = IP()**
- Define the IP Source (make it a fake IP, for example, a gateway IP address):
  - >> **P.src = "192.168.122.1"**
- Define the IP Target (victim machine):
  - >> **P.dst = "192.168.122.103"**
- Send the packet
  - send(P)**

```

Scapy v2.4.3
Scapy v2.4.3 83x25
      aSPY//YASa
      apyyyyCY////////YCa
      sY////////YSpcs  scpCY//Pp
ayp ayyyyyySCP//Pp      syY//C
AYAsAYYYYYYYY//Ps      cY//S
pCCCCY//p      cSSps y//Y
SPPPP//a      pP//AC//Y
A//A      cyP////C
p//Ac      sC//a
P////YCpc      A//A
scccccp///pSP///p      p//Y
sY/////////y caa      S//P
cayCyayP//Ya      pY/Ya
sY/PsY////YCc      aC//Yp
sc sccaCY//PCyPaapyCP//YSs
spCPY////////YPSps
ccaacs

Welcome to Scapy
Version 2.4.3
https://github.com/secdev/scapy
Have fun!
We are in France, we say Skappee.
OK? Merci.
-- Sebastien Chabal

using IPython 7.12.0
>>> p = IP()
>>> p.src="192.168.122.1"
>>> p.dst="192.168.122.103"
>>> send(p)
.
Sent 1 packets.
>>>

```

# SCAPY: Create/Send a Packet

## Multiple Packet Creation

- **Create multi-layer packets** (e.g., combining IP with TCP or UDP layers) to simulate different network protocols.
  - TCP Packet
    - `>> P = IP(dst="192.168.122.103") / TCP(dport=80)`
  - Send the packet
    - `send(P)`
- UDP Packet
  - `>> P = IP(dst="192.168.122.103") / UDP(dport=80)`
  - Send the packet
    - `send(P)`

```

Scapy v2.4.3
Scapy v2.4.3 83x25
SPPPP//a      pP//AC//Y      Have fun!
A//A          cyP///C
p///Ac        sC///a
P///YCpc     A//A
scccccp///pSP//p  p//Y      Wanna support scapy? Rate it on
sY////////y caa      S//P      sectools!
cayCyayP//Ya    pY/Ya      http://sectools.org/tool/scapy/
sY/PsY////YCc   aC//Yp      -- Satoshi Nakamoto
sc sccaCY//PCyapaapyCP//YSs
spCPY////////YPSps
ccaacs

using IPython 7.12.0
>>> P = IP(dst="192.168.122.103") / TCP(dport=80)
>>> P = IP(dst="192.168.122.103") / TCP(dport=80)
>>> send(P)
.
Sent 1 packets.
>>> send(P)
.
Sent 1 packets.
>>> P = IP(dst="192.168.122.103") / TCP(dport=80)
>>> send(P)
.
Sent 1 packets.
>>>

```

```

Scapy v2.4.3
Scapy v2.4.3 83x25
scccccp///pSP//p  p//Y      sectools!
sY////////y caa      S//P      http://sectools.org/tool/scapy/
cayCyayP//Ya    pY/Ya      -- Satoshi Nakamoto
sY/PsY////YCc   aC//Yp
sc sccaCY//PCyapaapyCP//YSs
spCPY////////YPSps
ccaacs

using IPython 7.12.0
>>> P = IP(dst="192.168.122.103") / TCP(dport=80)
>>> P = IP(dst="192.168.122.103") / TCP(dport=80)
>>> send(P)
.
Sent 1 packets.
>>> send(P)
.
Sent 1 packets.
>>> P = IP(dst="192.168.122.103") / TCP(dport=80)
>>> send(P)
.
Sent 1 packets.
>>> P = IP(dst="192.168.122.103") / UDP(dport=80)
>>> send(P)
.
Sent 1 packets.
>>>

```

# SCAPY: Create/Send a Packet

## Multiple Packet Creation

- Create multi-layer packet
  - TCP Packet
    - `>> P = IP(dst="192.168.122.103") / TCP(dport=80)`
  - Send the packet
    - `send(P)`

```

SPPPP//a      pP//AC//
A//A          cyP//
p//Ac         sC//
P//Ycpc      A//
scccccp//pSP//p  p//
sY////////y caa  S//
cayCyayP//Ya    pY//
sY/PsY////Ycc   aC//Yf
sc  sccaCY//PCyPaapyCP//YSs
      spCPY////////YPSps
      ccaacs

>>> P = IP(dst="192.168.122.103") / TCP(dport=80)
>>> send(P)
Sent 1 packets.
>>> send(P)
Sent 1 packets.
>>> P = IP(dst="192.168.122.103") / TCP(dport=80)
>>> send(P)
Sent 1 packets.
>>>

```

eth0

File Edit View Go Capture Analyze Statistics Telephony Wireless Tools Help

TCP

No.	Time	Source	Destination	Protocol	Length	Info
4290	2546.0550067...	34.104.35.123	192.168.122.103	TCP	60	[TCP Dup ACK
4306	2558.3347343...	192.168.122.103	142.251.39.42	TCP	60	[TCP Dup ACK
4307	2558.3353314...	142.251.39.42	192.168.122.103	TCP	60	[TCP Dup ACK
4343	2591.0838667...	192.168.122.103	34.104.35.123	TCP	60	[TCP Dup ACK
4344	2591.0852101...	34.104.35.123	192.168.122.103	TCP	60	[TCP Dup ACK
4361	2603.3693505...	192.168.122.103	142.251.39.42	TCP	60	[TCP Dup ACK
4362	2603.3712799...	142.251.39.42	192.168.122.103	TCP	60	[TCP Dup ACK
4411	2636.1552966...	192.168.122.103	34.104.35.123	TCP	60	[TCP Dup ACK
4412	2636.1570713...	34.104.35.123	192.168.122.103	TCP	60	[TCP Dup ACK
4431	2648.3992567...	192.168.122.103	142.251.39.42	TCP	60	[TCP Dup ACK
4432	2648.3992571...	142.251.39.42	192.168.122.103	TCP	60	[TCP Dup ACK
4456	2659.3614021...	192.168.122.27	192.168.122.103	TCP	60	[TCP Port num
4457	2659.3635469...	192.168.122.103	192.168.122.27	TCP	60	80 → 20 [RST,

▼ Frame 3660: 60 bytes on wire (480 bits), 60 bytes captured (480 bits) on interface 0 (eth0)
   
Section number: 1
   
▼ Interface id: 0 (eth0)
   
Interface name: eth0
   
Encapsulation type: Ethernet (1)
   
Arrival Time: Sep 18, 2024 13:10:52.199897292
   
UTC Arrival Time: Sep 18, 2024 17:10:52.199897292
   
Epoch Arrival Time: 1726679452.199897292
   
[Time shift for this packet: 0.000000000]
   
[Time delta from previous captured frame: 0.000000000]
   
[Time delta from previous displayed frame: 0.000000000]
   
[Time since reference or first frame: 22.000000000]
   
Frame Number: 3660
   
Frame Length: 60 bytes (480 bits)
   
Capture Length: 60 bytes (480 bits)

0000 08 00 27 27 29 8c 08 00 27 30 20 e0 08
   
0010 00 28 00 00 40 00 40 06 c4 fc c0 a8 7a
   
0020 7a 1b 00 50 00 14 00 00 00 00 00 00 00
   
0030 00 00 39 98 00 00 00 00 00 00 00 00 00

TCP is neither a field nor a protocol name. Packets: 4474 · Displayed: 870 (19.4%) Profile: Default

protocols.

03") / UDP(dport=80)

```

.org/tool/scapy/
Satoshi Nakamoto

```

# SCAPY: Create/Send a Packet

## Packet Sniffing

Scapy can be used to perform sniffing capabilities and monitor network traffic.

- TCP Packet
  - `packets = sniff(filter="ip and host 192.168.122.103", count=10)`
- Show all collected packets
  - `packets.show()`
- The victim machine doing anything; for example, ping to google

```

Scapy v2.4.3
Scapy v2.4.3 83x25
      A//A      cyP///C | Have fun!
      p///Ac      sC///a
      P///YCpc      A//A | Craft packets before they craft
      sccccp///pSP///p      p//Y | you.
      sY/////////y caa      S//P      -- Socrate
      cayCyayP//Ya      pY/Ya
      sY/PsY///YCc      aC//Yp
      sc sccaCY//PCyPaapyCP//YSs
      spCPY/////////YPSps
      ccaacs
      using IPython 7.12.0
>>> packets = sniff(filter="ip and host 192.168.122.103", count=10)
>>> packets = sniff(filter="ip and host 192.168.122.103", count=10)
>>> packets.show()
0000 Ether / IP / UDP / DNS Qry "b'google.com.'"
0001 Ether / IP / UDP / DNS Qry "b'google.com.'"
0002 Ether / IP / UDP / DNS Ans "142.251.208.174"
0003 Ether / IP / UDP / DNS Ans "2a00:1450:400d:808::200e"
0004 Ether / IP / ICMP 192.168.122.103 > 142.251.208.174 echo-request 0 / Raw
0005 Ether / IP / ICMP 142.251.208.174 > 192.168.122.103 echo-reply 0 / Raw
0006 Ether / IP / UDP / DNS Qry "b'174.208.251.142.in-addr.arpa.'"
0007 Ether / IP / UDP / DNS Ans "b'bud02s43-in-f14.1e100.net.'"
0008 Ether / IP / ICMP 192.168.122.103 > 142.251.208.174 echo-request 0 / Raw
0009 Ether / IP / ICMP 142.251.208.174 > 192.168.122.103 echo-reply 0 / Raw
>>>

```

```

Raspberry Pi OS Client [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
pi@raspberrypi: ~/Desktop
pi@raspberrypi:~/Desktop $ ping google.com
PING google.com (142.251.208.174) 56(84) bytes of data:
64 bytes from bud02s43-in-f14.1e100.net (142.251.208.174): icmp_seq=1 ttl=111 time=39.9 ms
64 bytes from bud02s43-in-f14.1e100.net (142.251.208.174): icmp_seq=2 ttl=111 time=36.7 ms
64 bytes from bud02s43-in-f14.1e100.net (142.251.208.174): icmp_seq=3 ttl=111 time=38.2 ms
^[[A^[[A64 bytes from bud02s43-in-f14.1e100.net (142.251.208.174): icmp_seq=4 ttl=111 time=40.1 ms
64 bytes from bud02s43-in-f14.1e100.net (142.251.208.174): icmp_seq=5 ttl=111 time=42.2 ms
64 bytes from bud02s43-in-f14.1e100.net (142.251.208.174): icmp_seq=6 ttl=111 time=40.7 ms
^C
--- google.com ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 75ms
rtt min/avg/max/mdev = 36.679/39.625/42.209/1.776 ms
pi@raspberrypi:~/Desktop $ ping google.com
PING google.com (142.251.208.174) 56(84) bytes of data:
64 bytes from bud02s43-in-f14.1e100.net (142.251.208.174): icmp_seq=1 ttl=111 time=37.1 ms
64 bytes from bud02s43-in-f14.1e100.net (142.251.208.174): icmp_seq=2 ttl=111 time=40.1 ms
64 bytes from bud02s43-in-f14.1e100.net (142.251.208.174): icmp_seq=3 ttl=111 time=37.2 ms
^C
--- google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 390ms
rtt min/avg/max/mdev = 37.059/38.147/40.144/1.422 ms
pi@raspberrypi:~/Desktop $

```

# Thank you

Please send all questions to:  
Abdelkader Shaaban,  
[abdelkader.Shaaban@ait.ac.at](mailto:abdelkader.Shaaban@ait.ac.at)  
Stefan Schauer  
[Stefan.Schauer@ait.ac.at](mailto:Stefan.Schauer@ait.ac.at)