SharkFest'26 Europe

Packets don't lie... Until they do!



Sponsored by:

PROFI TAP
DEEP NETWORK OBSERVABILITY

cPacket

TCP/IP training

endace
Record. Respond.

$ whoami

- Relational therapist for computer systems
  - Solve {application,network,performance}-issues by looking at the communication between systems
- Member Wireshark core-team since 2007
  - ... and of the Wireshark Technical Steering Committee
- Started SYN-bit in 2010
  - Application and Network troubleshooting
  - Protocol and packet analysis
  - Training (Wireshark, TCP, TLS)
- Organizer of the Wireshark Users NL meetups
- Wireshark Certified Analyst (WCA) #1 🎉





SYN-bit

deep traffic analysis

Application and network troubleshooting

Protocol and packet analysis

Training (Wireshark, TCP, SSL)

www.SYN-bit.nl

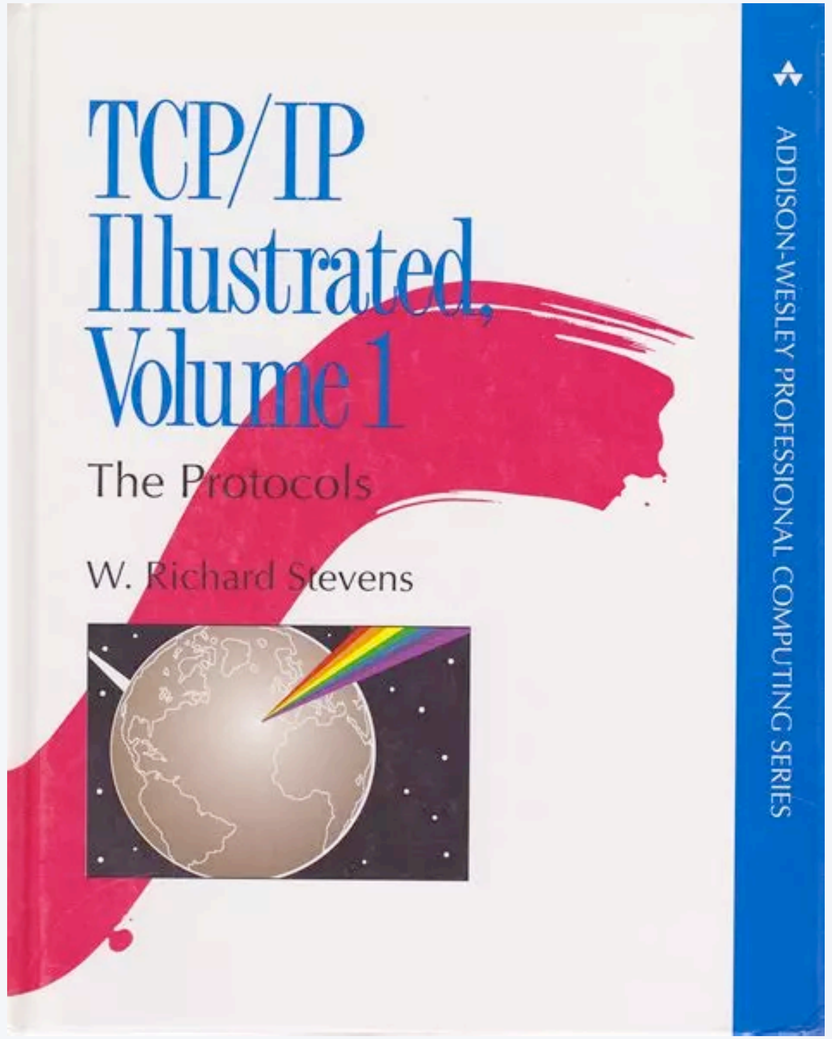
Packet don't lie!




Do you swear to tell the truth, the whole truth, and nothing but the truth?
RECHTZAL
GETUIGENBANK
TCP/IP PACKET
SOURCE: 192.168.1.10
DEST: 10.0.0.5
DATA
ACK
SYN
TCP Illustrated Volume 1
W. RICHARD STEVENS
TCP Illustrated, Volume 1
GETUIGENBANK

But sometimes they do NOT tell the truth, the whole truth and noting but the truth!

- Captured packets might be different than the packets on the network
- Sometimes not all packets are seen
- Sometimes too many packets are seen
- So, packets do lie!

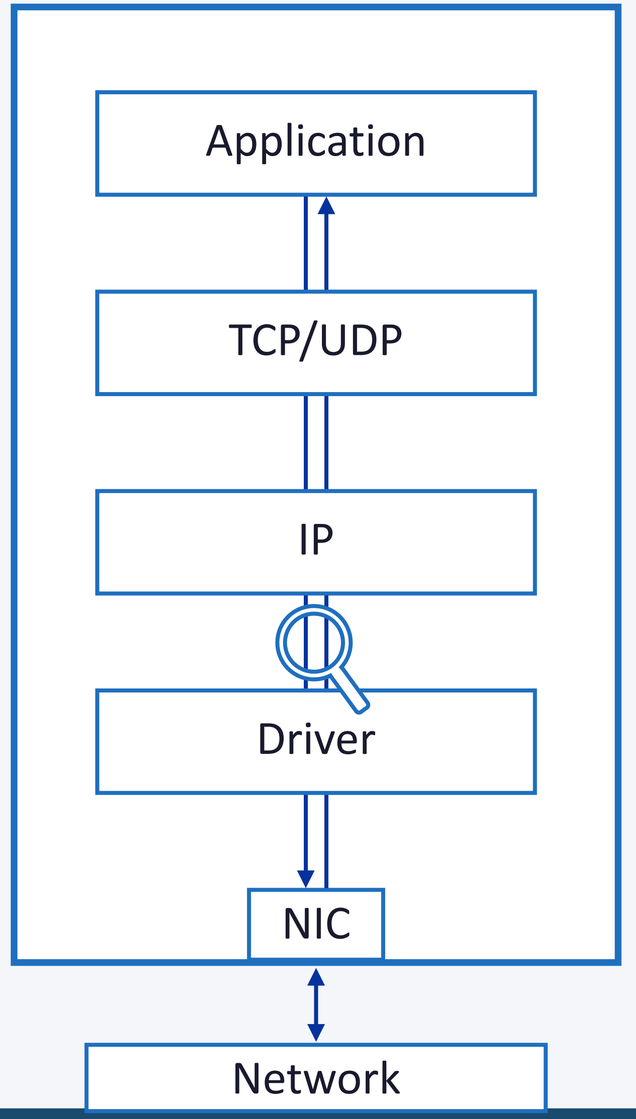


Packet lies due to capturing on an endpoint...

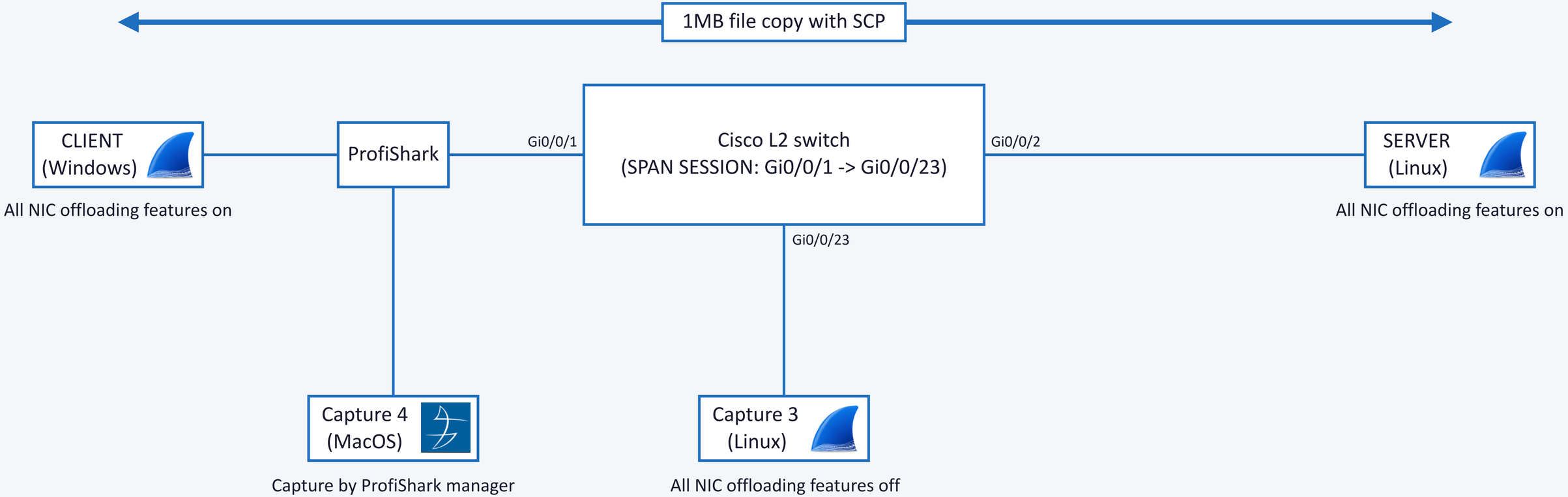


Packet lies when capturing on an endpoint...

- Packets that are too small (less than 64 bytes)
  - ... because of padding and FCS adding on the NIC (egress)
  - ... because the FCS is stripped by the NIC (ingress)
- Packets that are too big (more than 1518/9022 bytes)
  - ... because of TCP segmentation offloading (egress)
  - ... because of Generic Receive offloading (ingress, Linux default!)
- Bad checksums
  - ... because of checksum offloading (egress)
  - ... we disabled checksum checking by default in version 1.2 (IIRC)
- Delta times are unrealistically low
  - ... because time-stamping occurs after buffering (NIC & Kernel)
  - ... on 1Gbit/s, 1518 byte frames should be at least 12,3 µs apart



LAB setup...


1MB file copy with SCP
CLIENT
(Windows)
All NIC offloading features on
ProfiShark
Gi0/0/1
Cisco L2 switch
(SPAN SESSION: Gi0/0/1 -> Gi0/0/23)
Gi0/0/2
SERVER
(Linux)
All NIC offloading features on
Gi0/0/23
Capture 4
(MacOS)
Capture by ProfiShark manager
Capture 3
(Linux)
All NIC offloading features off

Let's look at the packets...



Solutions...

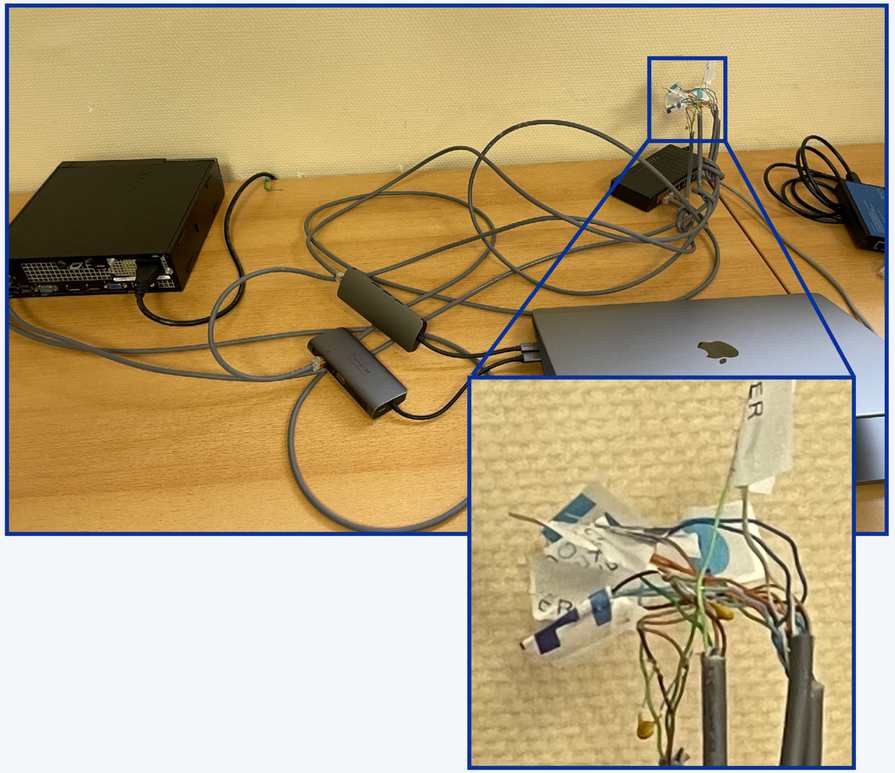
- Good
  - Recognize the lies and adjust your analysis accordingly
- Better
  - Don't capture on an endpoint
  - ... and capture on a SPAN port or TAP
  - ... and disable all offloading features on your capture device
- Best
  - Use a specialised capturing card or device
  - ... connected to a TAP

Packet lies due to the capturing setup...



Packet lies due to the capturing setup...

- Negative delta times and strange packet order
  - ... because capturing on multiple interfaces
    - ... with a passive DIY TAP
    - ... with a non-aggregating TAP
  - ... because of Generic receive offloading
- Many retransmissions and DUP-ACKs
  - ... because of spanning a VLAN in both directions
- Missing packets
  - ... because of disk I/O on the capture device
  - ... because of overloading the NIC buffer
  - ... because of oversubscribing the SPAN port



Let's look at the packets...



Extreme levels of RETRANSMISSIONs and DUP-ACKs...



No.	Time	Relative	Source	Destination	TTL	Identification	Protocol	Length	Info
1	14:00:32.86791…	0.000000…	10.0.0.254	224.0.0.5	1	0x1ace	OSPF	78	Hello Packet
2	14:00:32.86791…	0.000000…	10.0.0.254	224.0.0.5	1	0x1ace	OSPF	78	Hello Packet
3	14:00:32.86791…	0.000000…	10.0.0.254	224.0.0.5	1	0x1ace	OSPF	78	Hello Packet
4	14:00:37.93490…	5.066983…	ASUSTekCOMPU_1…	Broadcast			ARP	60	Who has 10.0.0.109? Tell 10.0.0.106
5	14:00:37.93490…	5.066983…	ASUSTekCOMPU_1…	Broadcast			ARP	60	Who has 10.0.0.109? Tell 10.0.0.106
6	14:00:37.93490…	5.066984…	ASUSTekCOMPU_1…	Broadcast			ARP	60	Who has 10.0.0.109? Tell 10.0.0.106
7	14:00:37.93490…	5.066984…	WistronInfoc_5…	ASUSTekCOMPU_1…			ARP	60	10.0.0.109 is at f0:de:f1:58:72:b5
8	14:00:37.93490…	5.066984…	WistronInfoc_5…	ASUSTekCOMPU_1…			ARP	60	10.0.0.109 is at f0:de:f1:58:72:b5
9	14:00:37.93498…	5.067065…	10.0.0.106	10.0.0.109	128	0x692b	TCP	66	49928 → 22 [SYN] Seq=0 Win=65535 Len=0 MSS=1464 WS=4 SACK_PERM
10	14:00:37.93498…	5.067065…	10.0.0.106	10.0.0.109	128	0x692b	TCP	66	[TCP Retransmission] 49928 → 22 [SYN] Seq=0 Win=65535 Len=0 MSS=1464 WS=4 SACK_PERM
11	14:00:37.93518…	5.067267…	10.0.0.109	10.0.0.106	64	0x0000	TCP	66	22 → 49928 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
12	14:00:37.93518…	5.067267…	10.0.0.109	10.0.0.106	64	0x0000	TCP	66	[TCP Retransmission] 22 → 49928 [SYN, ACK] Seq=0 Ack=1 Win=64240 Len=0 MSS=1460 SACK_PERM WS=128
13	14:00:37.93559…	5.067679…	10.0.0.106	10.0.0.109	128	0x692c	TCP	60	49928 → 22 [ACK] Seq=1 Ack=1 Win=65532 Len=0
14	14:00:37.93559…	5.067679…	10.0.0.106	10.0.0.109	128	0x692c	TCP	60	[TCP Dup ACK 13#1] 49928 → 22 [ACK] Seq=1 Ack=1 Win=65532 Len=0
15	14:00:37.95230…	5.084381…	10.0.0.109	10.0.0.106	64	0x5e42	SSHv2	96	Server: Protocol (SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.13)
16	14:00:37.95230…	5.084382…	10.0.0.109	10.0.0.106	64	0x5e42	TCP	96	[TCP Retransmission] 22 → 49928 [PSH, ACK] Seq=1 Ack=1 Win=64256 Len=42
17	14:00:37.95289…	5.084971…	10.0.0.106	10.0.0.109	128	0x692d	SSHv2	75	Client: Protocol (SSH-2.0-OpenSSH_5.8)
18	14:00:37.95289…	5.084971…	10.0.0.106	10.0.0.109	128	0x692d	TCP	75	[TCP Retransmission] 49928 → 22 [PSH, ACK] Seq=1 Ack=43 Win=65492 Len=21
19	14:00:37.95289…	5.084971…	10.0.0.109	10.0.0.106	64	0x5e43	TCP	60	22 → 49928 [ACK] Seq=43 Ack=22 Win=64256 Len=0
20	14:00:37.95289…	5.084971…	10.0.0.109	10.0.0.106	64	0x5e43	TCP	60	[TCP Dup ACK 19#1] 22 → 49928 [ACK] Seq=43 Ack=22 Win=64256 Len=0
21	14:00:37.95515…	5.087238…	10.0.0.109	10.0.0.106	64	0x5e44	SSHv2	1134	Server: Key Exchange Init
22	14:00:37.95515…	5.087239…	10.0.0.109	10.0.0.106	64	0x5e44	TCP	1134	[TCP Retransmission] 22 → 49928 [PSH, ACK] Seq=43 Ack=22 Win=64256 Len=1080
23	14:00:37.95515…	5.087239…	10.0.0.106	10.0.0.109	128	0x692e	SSHv2	1198	Client: Key Exchange Init
24	14:00:37.95515…	5.087239…	10.0.0.106	10.0.0.109	128	0x692e	TCP	1198	[TCP Retransmission] 49928 → 22 [PSH, ACK] Seq=22 Ack=1123 Win=64412 Len=1144
25	14:00:37.95522…	5.087303…	10.0.0.109	10.0.0.106	64	0x5e45	TCP	60	22 → 49928 [ACK] Seq=1123 Ack=1166 Win=64128 Len=0
26	14:00:37.95522…	5.087303…	10.0.0.109	10.0.0.106	64	0x5e45	TCP	60	[TCP Dup ACK 25#1] 22 → 49928 [ACK] Seq=1123 Ack=1166 Win=64128 Len=0
27	14:00:37.95724…	5.089328…	10.0.0.106	10.0.0.109	128	0x692f	SSHv2	134	Client: Elliptic Curve Diffie-Hellman Key Exchange Init
28	14:00:37.95724…	5.089328…	10.0.0.106	10.0.0.109	128	0x692f	TCP	134	[TCP Retransmission] 49928 → 22 [PSH, ACK] Seq=1166 Ack=1123 Win=64412 Len=80
29	14:00:37.95730…	5.089386…	10.0.0.109	10.0.0.106	64	0x5e46	TCP	60	22 → 49928 [ACK] Seq=1123 Ack=1246 Win=64128 Len=0
30	14:00:37.95730…	5.089387…	10.0.0.109	10.0.0.106	64	0x5e46	TCP	60	[TCP Dup ACK 29#1] 22 → 49928 [ACK] Seq=1123 Ack=1246 Win=64128 Len=0
31	14:00:37.95928…	5.091365…	10.0.0.109	10.0.0.106	64	0x5e47	SSHv2	366	Server: Elliptic Curve Diffie-Hellman Key Exchange Reply, New Keys
32	14:00:37.95928…	5.091366…	10.0.0.109	10.0.0.106	64	0x5e47	TCP	366	[TCP Retransmission] 22 → 49928 [PSH, ACK] Seq=1123 Ack=1246 Win=64128 Len=312
33	14:00:37.96771…	5.099796…	10.0.0.106	10.0.0.109	128	0x6930	SSHv2	70	Client: New Keys
34	14:00:37.96771…	5.099796…	10.0.0.106	10.0.0.109	128	0x6930	TCP	70	[TCP Retransmission] 49928 → 22 [PSH, ACK] Seq=1246 Ack=1435 Win=64100 Len=16
35	14:00:37.96771…	5.099796…	10.0.0.109	10.0.0.106	64	0x5e48	TCP	60	22 → 49928 [ACK] Seq=1435 Ack=1262 Win=64128 Len=0
36	14:00:37.96771…	5.099796…	10.0.0.109	10.0.0.106	64	0x5e48	TCP	60	[TCP Dup ACK 35#1] 22 → 49928 [ACK] Seq=1435 Ack=1262 Win=64128 Len=0

Spanning a vlan, the wrong way ;-)

```
test-sw01#sh run | incl mon
monitor session 1 source vlan 100
monitor session 1 destination interface Gi1/0/23
test-sw01#
```



```
test-sw01#sh mon ses 1
Session 1
---------
Type                   : Local Session
Source VLANs           :
    Both               : 100
Destination Ports      : Gi1/0/23
    Encapsulation      : Native
          Ingress      : Disabled


test-sw01#
```

Oversubscribing...



... DISK I/O on capturing device:

Capture

Hardware:	Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz (with SSE4.2)
OS:	Linux 6.8.0-90-generic
Application:	Dumpcap (Wireshark) 4.0.6 (Git v4.0.6 packaged as 4.0.6-1~exp1~ubuntu22.04.0~ppa1)

Interfaces

Interface	Interface Description	Dropped packets	Capture filter	Link type	Packet size limit (snaplen)
enp0s31f6	Unknown	252146 (16.4%)	none	Ethernet	262144 bytes

... SPAN destination port on the switch:

```
test-sw01#clear counters gi1/0/23
Clear "show interface" counters on this interface [confirm]y
test-sw01#sh int gi1/0/23 | incl drop
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 0
     0 unknown protocol drops
test-sw01#sh int gi1/0/23 | incl drop
  Input queue: 0/75/0/0 (size/max/drops/flushes); Total output drops: 122038
     0 unknown protocol drops
test-sw01#
```

Solutions...



- Match the capture setup to the infrastructure
- Make sure you only capture packets once
  - Use **`editcap -d`** to remove duplicates (might also delete valid duplicates!)
- Make sure you dimension your capture setup right
- Monitor the resources during capture

Packet lies due to Wireshark's dissection engine...



Packet lies due to Wireshark's dissection engine...

- Packets are marked incorrectly by the TCP dissector
  - ... because TCP analysis is a tough cookie
  - ... e.g. Out-Of-Order instead of Retransmission

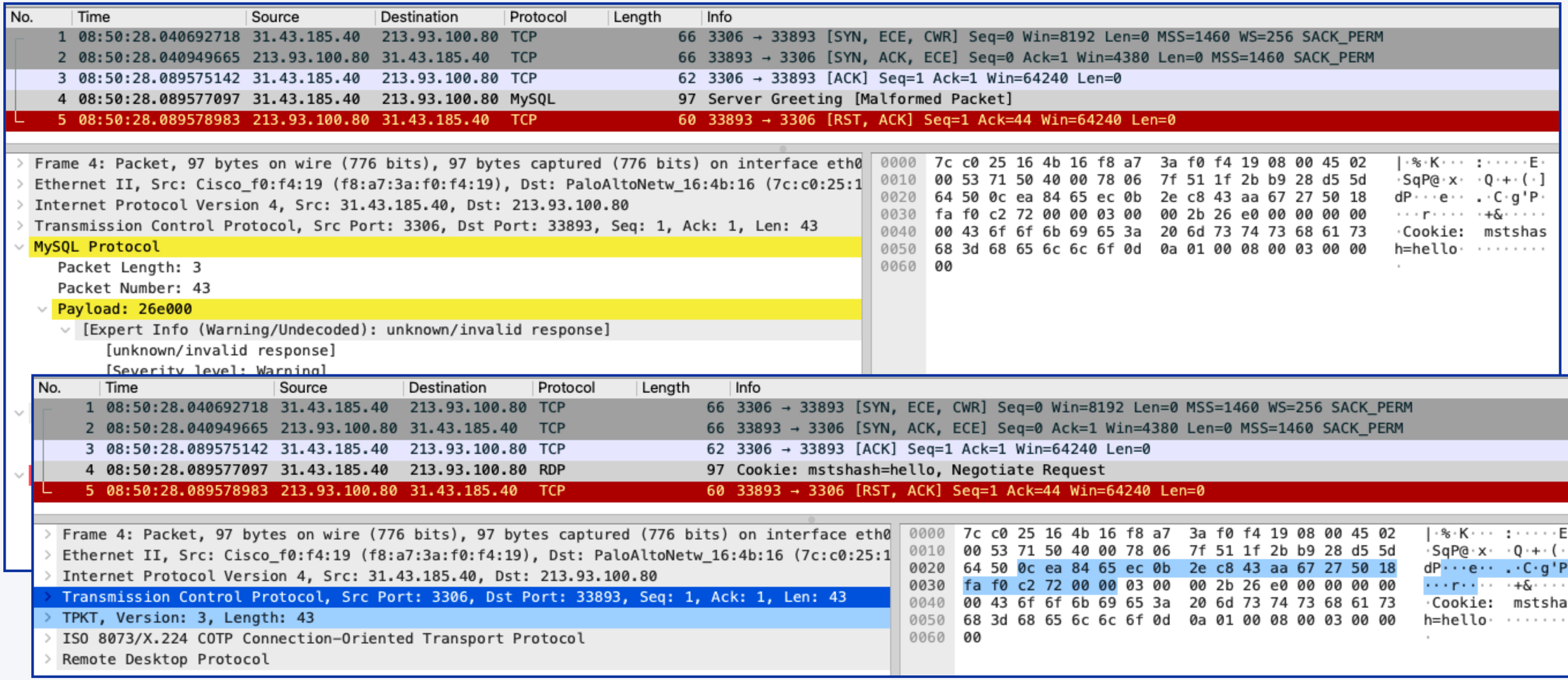
- Packets are dissected as protocol X instead of protocol Y
  - ... because of greedy heuristics
  - ... because sometimes client ports were used to select the protocol
    - Fixed after a presenter at Sharkfest misinterpreted Wireshark's wrong dissector choices

10	2.809247	2.552311	10.3.3.51	10.1.1.51	HTTP	1,314	[TCP Retransmission] Continuation
11	8.048384	5.239137	10.3.3.51	10.1.1.51	HTTP	1,314	[TCP Retransmission] Continuation
12	18.398673	10.350289	10.3.3.51	10.1.1.51	HTTP	590	[TCP Retransmission] Continuation
13	18.614806	0.216133	10.1.1.51	10.3.3.51	TCP	60	49620 → 80 [ACK] Seq=504 Ack=773 Win=66008 Len=0
14	18.616121	0.001315	10.3.3.51	10.1.1.51	HTTP	590	[TCP Retransmission] Continuation
15	18.616636	0.000515	10.3.3.51	10.1.1.51	HTTP	590	[TCP Out-Of-Order] Continuation
16	18.626722	0.010086	10.1.1.51	10.3.3.51	TCP	60	49620 → 80 [ACK] Seq=504 Ack=1845 Win=66780 Len=0
17	18.628192	0.001470	10.3.3.51	10.1.1.51	HTTP	590	[TCP Retransmission] Continuation
18	18.628691	0.000499	10.3.3.51	10.1.1.51	HTTP	590	[TCP Out-Of-Order] Continuation
19	18.629176	0.000485	10.3.3.51	10.1.1.51	HTTP	590	[TCP Retransmission] Continuation

Let's look at the packets...



Misinterpretation of Wireshark's faulty choice of dissector...



No.	Time	Source	Destination	Protocol	Length	Info
1	08:50:28.040692718	31.43.185.40	213.93.100.80	TCP	66	3306 → 33893 [SYN, ECE, CWR] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM
2	08:50:28.040949665	213.93.100.80	31.43.185.40	TCP	66	33893 → 3306 [SYN, ACK, ECE] Seq=0 Ack=1 Win=4380 Len=0 MSS=1460 SACK_PERM
3	08:50:28.089575142	31.43.185.40	213.93.100.80	TCP	62	3306 → 33893 [ACK] Seq=1 Ack=1 Win=64240 Len=0
4	08:50:28.089577097	31.43.185.40	213.93.100.80	MySQL	97	Server Greeting [Malformed Packet]
5	08:50:28.089578983	213.93.100.80	31.43.185.40	TCP	60	33893 → 3306 [RST, ACK] Seq=1 Ack=44 Win=64240 Len=0

```
> Frame 4: Packet, 97 bytes on wire (776 bits), 97 bytes captured (776 bits) on interface eth0
> Ethernet II, Src: Cisco_f0:f4:19 (f8:a7:3a:f0:f4:19), Dst: PaloAltoNetw_16:4b:16 (7c:c0:25:1
> Internet Protocol Version 4, Src: 31.43.185.40, Dst: 213.93.100.80
> Transmission Control Protocol, Src Port: 3306, Dst Port: 33893, Seq: 1, Ack: 1, Len: 43
v MySQL Protocol
    Packet Length: 3
    Packet Number: 43
  v Payload: 26e000
    v [Expert Info (Warning/Undecoded): unknown/invalid response]
        [unknown/invalid response]
        [Severity level: Warning]
```

```
0000  7c c0 25 16 4b 16 f8 a7  3a f0 f4 19 08 00 45 02   |·%·K··· :·····E·
0010  00 53 71 50 40 00 78 06  7f 51 1f 2b b9 28 d5 5d   ·SqP@·x· ·Q·+·(·]
0020  64 50 0c ea 84 65 ec 0b  2e c8 43 aa 67 27 50 18   dP···e·· .·C·g'P·
0030  fa f0 c2 72 00 00 03 00  00 2b 26 e0 00 00 00 00   ···r···· ·+&·····
0040  00 43 6f 6f 6b 69 65 3a  20 6d 73 74 73 68 61 73   ·Cookie:  mstshas
0050  68 3d 68 65 6c 6c 6f 0d  0a 01 00 08 00 03 00 00   h=hello· ········
0060  00                                                ·
```

No.	Time	Source	Destination	Protocol	Length	Info
1	08:50:28.040692718	31.43.185.40	213.93.100.80	TCP	66	3306 → 33893 [SYN, ECE, CWR] Seq=0 Win=8192 Len=0 MSS=1460 WS=256 SACK_PERM
2	08:50:28.040949665	213.93.100.80	31.43.185.40	TCP	66	33893 → 3306 [SYN, ACK, ECE] Seq=0 Ack=1 Win=4380 Len=0 MSS=1460 SACK_PERM
3	08:50:28.089575142	31.43.185.40	213.93.100.80	TCP	62	3306 → 33893 [ACK] Seq=1 Ack=1 Win=64240 Len=0
4	08:50:28.089577097	31.43.185.40	213.93.100.80	RDP	97	Cookie: mstshash=hello, Negotiate Request
5	08:50:28.089578983	213.93.100.80	31.43.185.40	TCP	60	33893 → 3306 [RST, ACK] Seq=1 Ack=44 Win=64240 Len=0

```
> Frame 4: Packet, 97 bytes on wire (776 bits), 97 bytes captured (776 bits) on interface eth0
> Ethernet II, Src: Cisco_f0:f4:19 (f8:a7:3a:f0:f4:19), Dst: PaloAltoNetw_16:4b:16 (7c:c0:25:1
> Internet Protocol Version 4, Src: 31.43.185.40, Dst: 213.93.100.80
> Transmission Control Protocol, Src Port: 3306, Dst Port: 33893, Seq: 1, Ack: 1, Len: 43
> TPKT, Version: 3, Length: 43
> ISO 8073/X.224 COTP Connection-Oriented Transport Protocol
> Remote Desktop Protocol
```

```
0000  7c c0 25 16 4b 16 f8 a7  3a f0 f4 19 08 00 45 02   |·%·K··· :·····E·
0010  00 53 71 50 40 00 78 06  7f 51 1f 2b b9 28 d5 5d   ·SqP@·x· ·Q·+·(·]
0020  64 50 0c ea 84 65 ec 0b  2e c8 43 aa 67 27 50 18   dP···e·· .·C·g'P·
0030  fa f0 c2 72 00 00 03 00  00 2b 26 e0 00 00 00 00   ···r···· ·+&·····
0040  00 43 6f 6f 6b 69 65 3a  20 6d 73 74 73 68 61 73   ·Cookie:  mstshas
0050  68 3d 68 65 6c 6c 6f 0d  0a 01 00 08 00 03 00 00   h=hello· ········
0060  00                                                ·
```

Solutions...

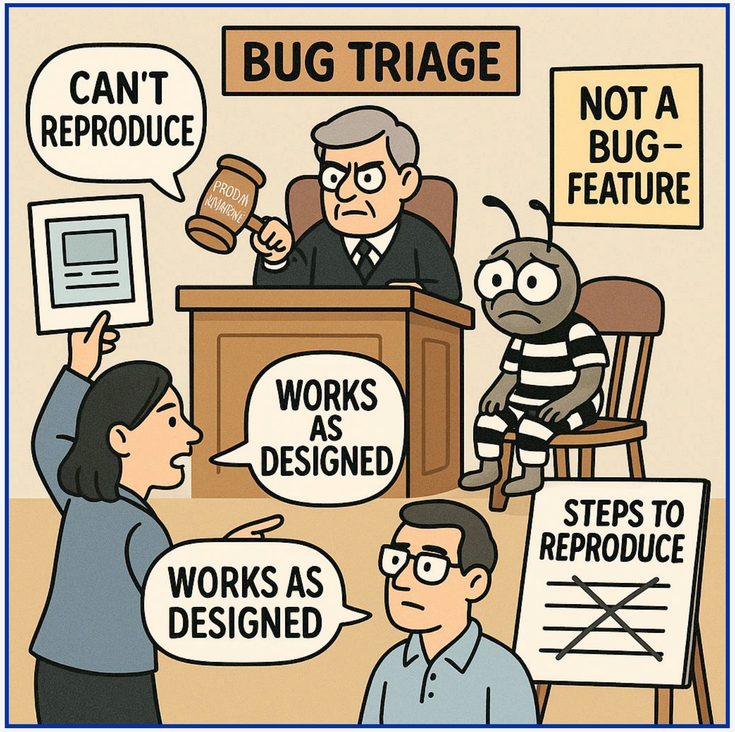
- Get good at TCP analysis so you can verify!
  - ... this means putting in a lot of practice time
- Always... ALWAYS... check if Wireshark is not mistaken!
  - ... this means learning about the protocols you need to analyse
- Disable all the dissectors that you don't use/expect
- Use decode-as for individual cases

Packet lies due to network device issues...



Packet lies due to network device issues...

- VLAN-tagged ingress frames and untagged egress frames
  - ... because that is how the mirroring on Ubiquity works
- Inside IP on egress and outside IP on ingress packets
  - ... because that is how Palo Alto captures packets (pre-NAT)
- IP address filter does not match PPPoE packets on ingress
  - ... because that is how Palo Alto matching filters work
  - ... feature request filed under NSFR-I-29610
- First fragment of fragmented IP datagrams not mirrored
  - ... because there is a bug in the HPE-Aruba mirror functionality
- The list continues...



Let's look at the packets...



Solutions...

- Get to know how your devices work before you need to do a packet capture for a high priority incident
- Double-check your capture setup on things that seem odd
  - ... and report a bug if you suspect one
- Use a TAP to get a clean capture without device artefacts

Debunking the lies...



Debunking the lies...

- Most packets don't lie!
  - The captured packets show what is seen on the network
  - Often only way to prove a bug or application issue
- Packets do lie sometimes though!
  - Where and how you are capturing has a influence on what you get to see
  - Wireshark's dissection can be wrong or misleading
- Resolution
  - Make sure to arrange a proper capture setup using the right hardware
  - Educate yourself in protocol analysis so that you can recognise when packets...
    *"do **not** tell the truth, the whole truth and nothing but the truth!"*



Thank you!

WIRESHARK

Sake Blok
sake.blok@SYN-bit.nl

SYN-bit
deep traffic analysis

Please provide feedback
Either by scanning the qr-code or by
clicking on the feedback button in the agenda!