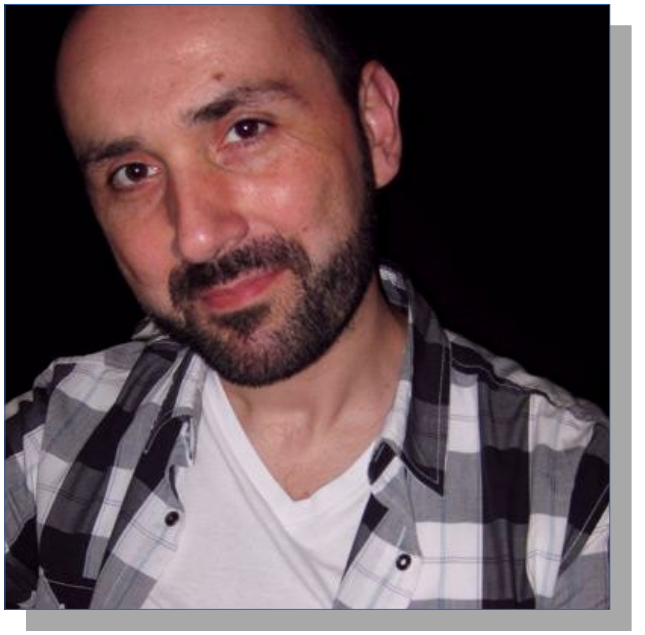
Wireshark – narzędzie do diagnostyki problemów sieciowych

18.10.2024

Jacek Rokicki

- od ponad 25 lat pracuję w branży IT,
- sys/net/dev/sec ops
  projektuje i utrzymuję systemy działające pod dużym obciążeniem i z niskimi opóźnieniami dla rynku mediów,
- lubię OS z rodziny *nix oraz soft Libre/Open Source,
- w 2011 poznałem produkty Mikrotika, cenię je za niezawodność i dobry stosunek ceny do możliwości.



MikroTik
MTCNA
MTCRE
MTCINE
MTCSWE
MTCSE
MTCTCE
MTCIPv6E
MTCUME
MTCWE
MTCEWE

GitHub yacq/MBUM

jacek-rokicki

Agenda

- O narzędziach do obserwowania ruchu w sieci
- PCAP – kontener na pakiety
- Ustawienia programu, różne tryby pracy
- Rodzaje filtrów, przykłady
- Sposoby przechwytywania ruchu
- Jak przechwytywać ruch na MT
- Powtórka z MTCNA – komunikacja sieciowa
- Przykłady użycia
- Zakończenie

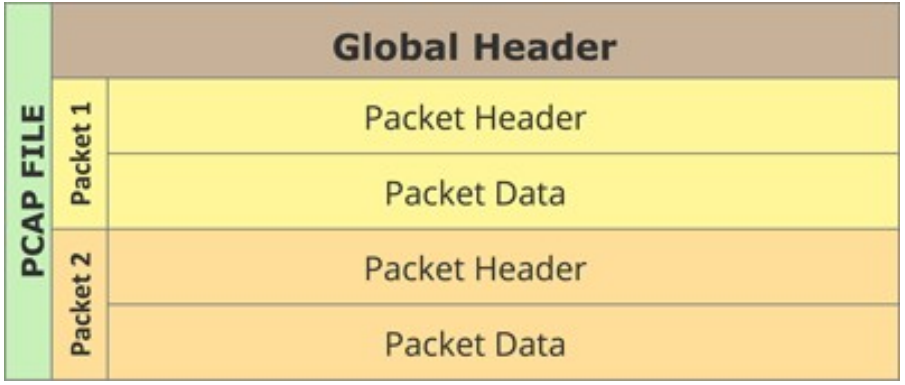
tcpdump, libpcap, wireshark

- W 1991 powstaje tcpdump, sniffer pracujący w trybie tekstowym, który pozwala na wyświetlenie przechwyconych danych na ekranie lub zapis do pliku
- Po pewnym czasie wydzielono część kodu tcpdump do zewnętrznej biblioteki libpcap
- Pod koniec lat 90 powstaje Wireshark, graficzne narzędzie korzystające z libpcap

Plik pcap i jego alternatywy

- Kontener na przechwycone dane
- Wireshark akceptuje także nowsze zamienniki:
  - PcapNG – umożliwia przechowywanie pakietów z różnymi typami warstwy łącza np. Ethernet i 802.11 w tym samym pliku, dołączanie komentarzy, przechowywanie metadanych, łatwe scalanie wielu plików
  - ERF – natywny format dla interfejsów Endace DAG, plik składa się z serii rekordów gdzie każdy opisuje jeden pakiet, nie posiada nagłówka więc łatwo taki plik dzielić i scalać

Format pliku pcap




```
d4 c3 b2 a1 02 00 04 00
00 00 00 00 00 00 00 00
00 00 04 00 01 00 00 00
00 45 d4 5e 18 8e 0c 00
42 00 00 00 42 00 00 00
00 1e ec 26 d2 ac 26 02
06 49 6b 31 08 00 45 02
00 34 30 8c 40 00 72 06
81 7f 2e 69 63 a3 c0 a8
04 02 cf 3a 00 50 8d a5
ee 7b 00 00 00 00 80 c2
20 00 ac 29 00 00 02 04
05 78 01 03 03 08 01 01
04 02 00 45 d4 5e 2c 77
0d 00 36 00 00 00 36 00
00 00 00 1e ec 26 d2 ac
```

24 byte PCAP Header
Link-Layer Type = Ethernet (0x00000001)

16 byte Packet Header
Timestamp = 1 June 2020
Packet length = 66 bytes (0x00000042)

66 bytes of Packet Data
Destination MAC = 00:1e:ec:26:d2:ac
Source MAC = 26:02:06:49:6b:31
Source IP = 46.105.99.163
Destination IP = 192.168.4.2

16 byte Packet Header
Packet length = 54 bytes (0x00000036)

Wireshark

- https://www.wireshark.org/download.html

  - Windows (x64 i Arm64)

  - macOS

  - Source Code

- Linux (pakiet dostępny w main repo dla każdej architektury)
- Uwaga na złośliwe linki, które mogą być na szczycie wyszukiwania

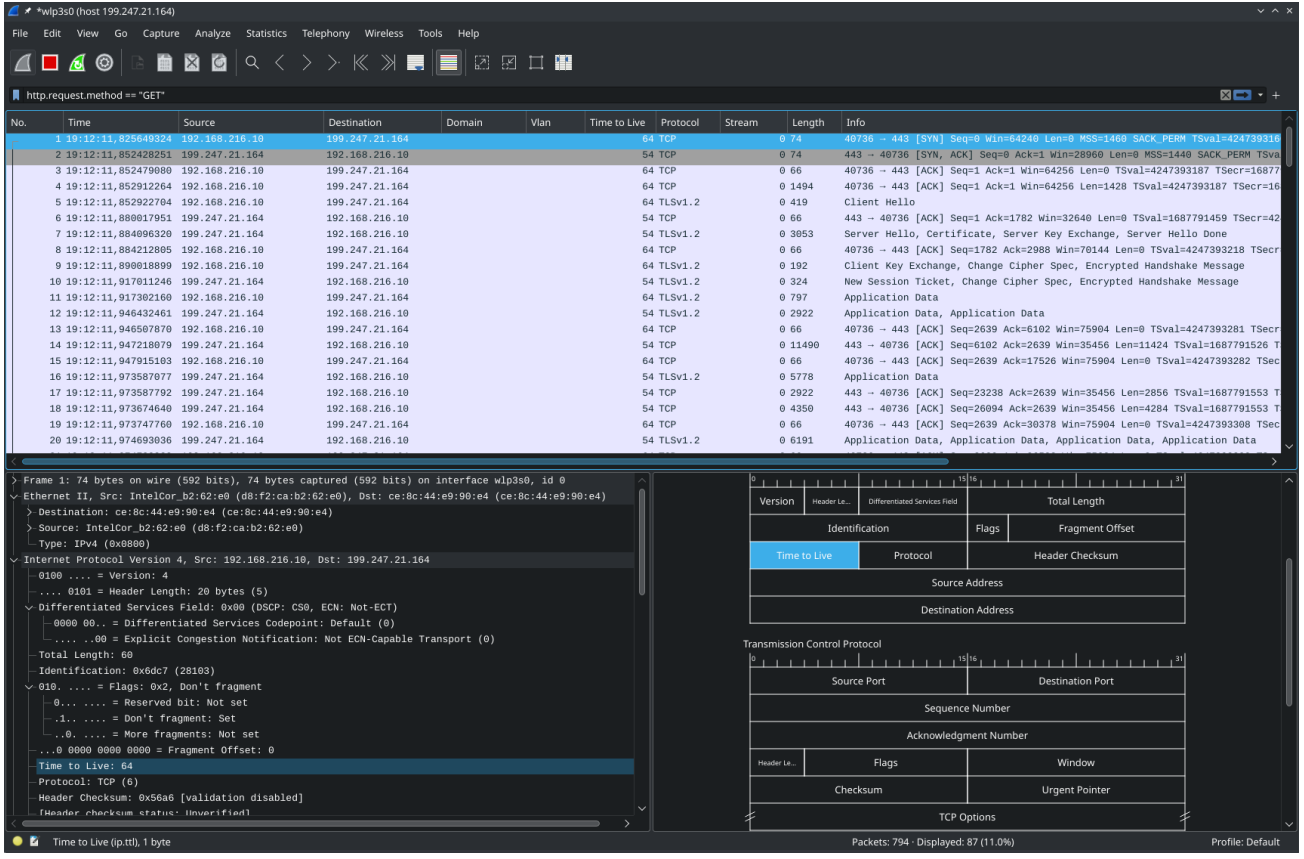
Tryby pracy

- Tryby działania:
  - online (nasłuch interfejsu)
  - offline (analiza danych z pliku)

- Tryby przechwytywania:
  - promiscuous/non-promiscuous
  - monitor (do sieci bezprzewodowych)

Konfiguracja UI

- Format czasu
- Kolorowanie
- Profile
- Dodatkowe kolumny



Filtry capture i display

- Capture Filter – definiowane na poziomie wyboru interfejsów, działaja w warstwie jądra systemu, wydajne, ograniczone możliwości

host 10.0.0.1

udp and host 192.168.0.1

port 80

ether host 00:ff:11:22:33:ff

- Display Filter – definiowane w oknie głównym, działają w userspace, umożliwiają tworzenie skomplikowanych reguł

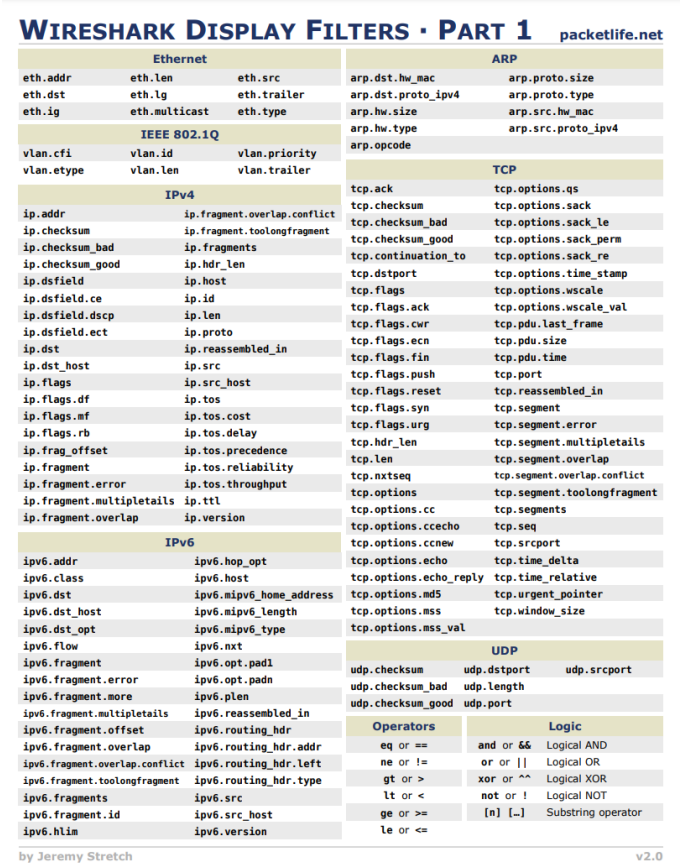
ip.addr == 192.168.0.1

tcp.port == 80

ip.src = 172.16.0.1 && http.request.method == "GET"

tcp.flags.syn == 1

Przykłady filtrów



Wireshark Display Filters · Part 1

packetlife.net

Ethernet

eth.addr	eth.len	eth.src
eth.dst	eth.lg	eth.trailer
eth.ig	eth.multicast	eth.type

IEEE 802.1Q

vlan.cfi	vlan.id	vlan.priority
vlan.etype	vlan.len	vlan.trailer

IPv4

ip.addr	ip.fragment.overlap.conflict
ip.checksum	ip.fragment.toolongfragment
ip.checksum_bad	ip.fragments
ip.checksum_good	ip.hdr_len
ip.dsfield	ip.host
ip.dsfield.ce	ip.id
ip.dsfield.dscp	ip.len
ip.dsfield.ect	ip.proto
ip.dst	ip.reassembled_in
ip.dst_host	ip.src
ip.flags	ip.src_host
ip.flags.df	ip.tos
ip.flags.mf	ip.tos.cost
ip.flags.rb	ip.tos.delay
ip.frag_offset	ip.tos.precedence
ip.fragment	ip.tos.reliability
ip.fragment.error	ip.tos.throughput
ip.fragment.multipletails	ip.ttl
ip.fragment.overlap	ip.version

IPv6

ipv6.addr	ipv6.hop_opt
ipv6.class	ipv6.host
ipv6.dst	ipv6.mipv6_home_address
ipv6.dst_host	ipv6.mipv6_length
ipv6.dst_opt	ipv6.mipv6_type
ipv6.flow	ipv6.nxt
ipv6.fragment	ipv6.opt.pad1
ipv6.fragment.error	ipv6.opt.padn
ipv6.fragment.more	ipv6.plen
ipv6.fragment.multipletails	ipv6.reassembled_in
ipv6.fragment.offset	ipv6.routing_hdr
ipv6.fragment.overlap	ipv6.routing_hdr.addr
ipv6.fragment.overlap.conflict	ipv6.routing_hdr.left
ipv6.fragment.toolongfragment	ipv6.routing_hdr.type
ipv6.fragments	ipv6.src
ipv6.fragment.id	ipv6.src_host
ipv6.hlim	ipv6.version

ARP

arp.dst.hw_mac	arp.proto.size
arp.dst.proto_ipv4	arp.proto.type
arp.hw.size	arp.src.hw_mac
arp.hw.type	arp.src.proto_ipv4
arp.opcode	

TCP

tcp.ack	tcp.options.qs
tcp.checksum	tcp.options.sack
tcp.checksum_bad	tcp.options.sack_le
tcp.checksum_good	tcp.options.sack_perm
tcp.continuation_to	tcp.options.sack_re
tcp.dstport	tcp.options.time_stamp
tcp.flags	tcp.options.wscale
tcp.flags.ack	tcp.options.wscale_val
tcp.flags.cwr	tcp.pdu.last_frame
tcp.flags.ecn	tcp.pdu.size
tcp.flags.fin	tcp.pdu.time
tcp.flags.push	tcp.port
tcp.flags.reset	tcp.reassembled_in
tcp.flags.syn	tcp.segment
tcp.flags.urg	tcp.segment.error
tcp.hdr_len	tcp.segment.multipletails
tcp.len	tcp.segment.overlap
tcp.nxtseq	tcp.segment.overlap.conflict
tcp.options	tcp.segment.toolongfragment
tcp.options.cc	tcp.segments
tcp.options.ccecho	tcp.seq
tcp.options.ccnew	tcp.srcport
tcp.options.echo	tcp.time_delta
tcp.options.echo_reply	tcp.time_relative
tcp.options.md5	tcp.urgent_pointer
tcp.options.mss	tcp.window_size
tcp.options.mss_val	

UDP

udp.checksum	udp.dstport	udp.srcport
udp.checksum_bad	udp.length	
udp.checksum_good	udp.port	

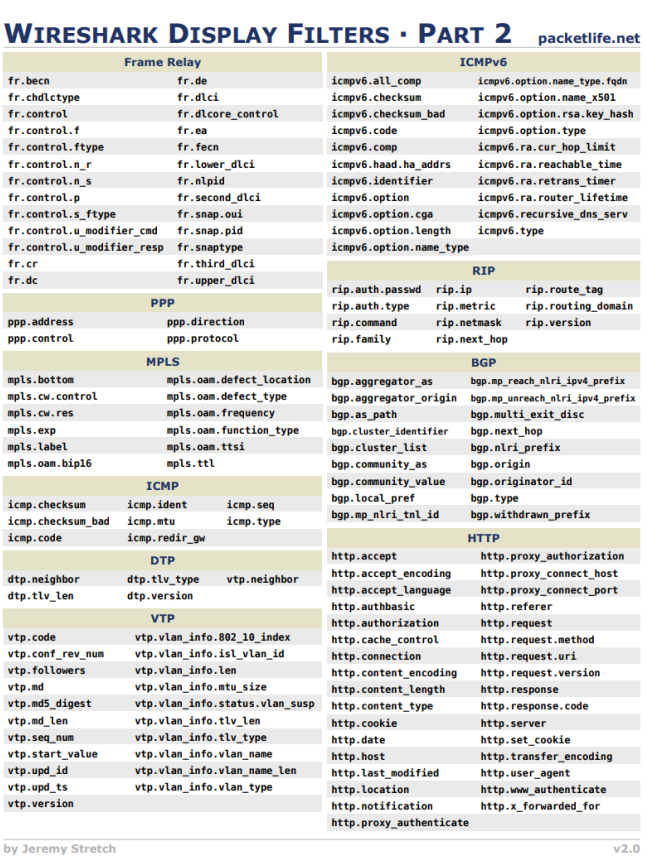
Operators

eq or ==	
ne or !=	
gt or >	
lt or <	
ge or >=	
le or <=	

Logic

and or &&	Logical AND
or or ||	Logical OR
xor or ^^	Logical XOR
not or !	Logical NOT
[n] […]	Substring operator

by Jeremy Stretch v2.0



Wireshark Display Filters · Part 2

packetlife.net

Frame Relay

fr.becn	fr.de
fr.chdlctype	fr.dlci
fr.control	fr.dlcore_control
fr.control.f	fr.ea
fr.control.ftype	fr.fecn
fr.control.n_r	fr.lower_dlci
fr.control.n_s	fr.nlpid
fr.control.p	fr.second_dlci
fr.control.s_ftype	fr.snap.oui
fr.control.u_modifier_cmd	fr.snap.pid
fr.control.u_modifier_resp	fr.snaptype
fr.cr	fr.third_dlci
fr.dc	fr.upper_dlci

PPP

ppp.address	ppp.direction
ppp.control	ppp.protocol

MPLS

mpls.bottom	mpls.oam.defect_location
mpls.cw.control	mpls.oam.defect_type
mpls.cw.res	mpls.oam.frequency
mpls.exp	mpls.oam.function_type
mpls.label	mpls.oam.ttsi
mpls.oam.bip16	mpls.ttl

ICMP

icmp.checksum	icmp.ident	icmp.seq
icmp.checksum_bad	icmp.mtu	icmp.type
icmp.code	icmp.redir_gw	

DTP

dtp.neighbor	dtp.tlv_type	vtp.neighbor
dtp.tlv_len	dtp.version	

VTP

vtp.code	vtp.vlan_info.802_10_index
vtp.conf_rev_num	vtp.vlan_info.isl_vlan_id
vtp.followers	vtp.vlan_info.len
vtp.md	vtp.vlan_info.mtu_size
vtp.md5_digest	vtp.vlan_info.status.vlan_susp
vtp.md_len	vtp.vlan_info.tlv_len
vtp.seq_num	vtp.vlan_info.tlv_type
vtp.start_value	vtp.vlan_info.vlan_name
vtp.upd_id	vtp.vlan_info.vlan_name_len
vtp.upd_ts	vtp.vlan_info.vlan_type
vtp.version	

ICMPv6

icmpv6.all_comp	icmpv6.option.name_type.fqdn
icmpv6.checksum	icmpv6.option.name_x501
icmpv6.checksum_bad	icmpv6.option.rsa.key_hash
icmpv6.code	icmpv6.option.type
icmpv6.comp	icmpv6.ra.cur_hop_limit
icmpv6.haad.ha_addrs	icmpv6.ra.reachable_time
icmpv6.identifier	icmpv6.ra.retrans_timer
icmpv6.option	icmpv6.ra.router_lifetime
icmpv6.option.cga	icmpv6.recursive_dns_serv
icmpv6.option.length	icmpv6.type
icmpv6.option.name_type	

RIP

rip.auth.passwd	rip.ip	rip.route_tag
rip.auth.type	rip.metric	rip.routing_domain
rip.command	rip.netmask	rip.version
rip.family	rip.next_hop	

BGP

bgp.aggregator_as	bgp.mp_reach_nlri_ipv4_prefix
bgp.aggregator_origin	bgp.mp_unreach_nlri_ipv4_prefix
bgp.as_path	bgp.multi_exit_disc
bgp.cluster_identifier	bgp.next_hop
bgp.cluster_list	bgp.nlri_prefix
bgp.community_as	bgp.origin
bgp.community_value	bgp.originator_id
bgp.local_pref	bgp.type
bgp.mp_nlri_tnl_id	bgp.withdrawn_prefix

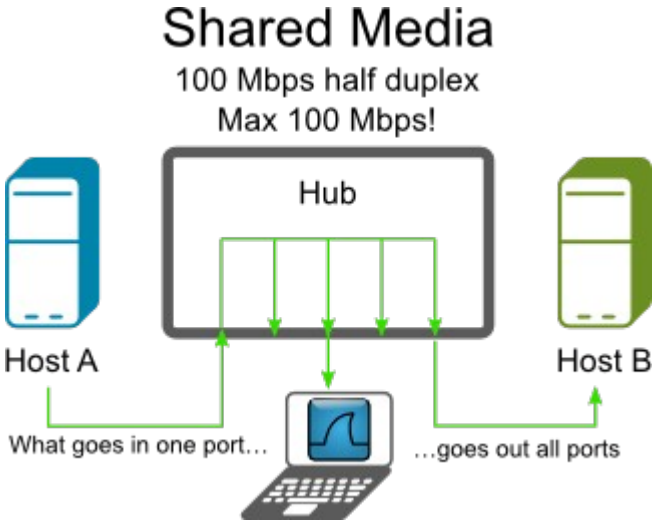
HTTP

http.accept	http.proxy_authorization
http.accept_encoding	http.proxy_connect_host
http.accept_language	http.proxy_connect_port
http.authbasic	http.referer
http.authorization	http.request
http.cache_control	http.request.method
http.connection	http.request.uri
http.content_encoding	http.request.version
http.content_length	http.response
http.content_type	http.response.code
http.cookie	http.server
http.date	http.set_cookie
http.host	http.transfer_encoding
http.last_modified	http.user_agent
http.location	http.www_authenticate
http.notification	http.x_forwarded_for
http.proxy_authenticate	

by Jeremy Stretch v2.0

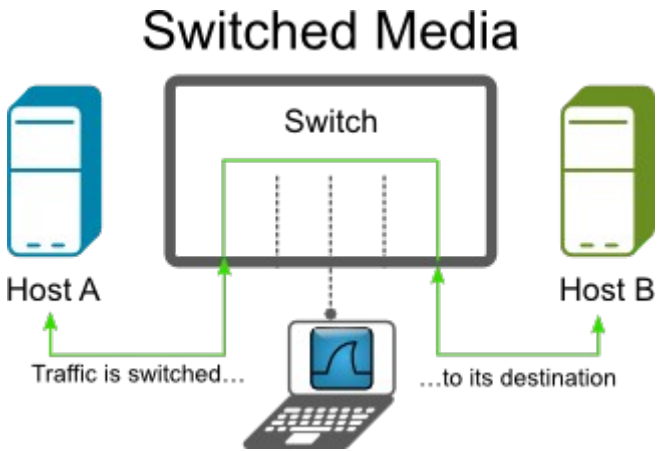
Metody przechwytywania ruchu w Ethernecie

- Współdzielone medium – ethernet hub
  - hub wysyła ramki na wszystkie porty
  - obecnie praktycznie nieużywany



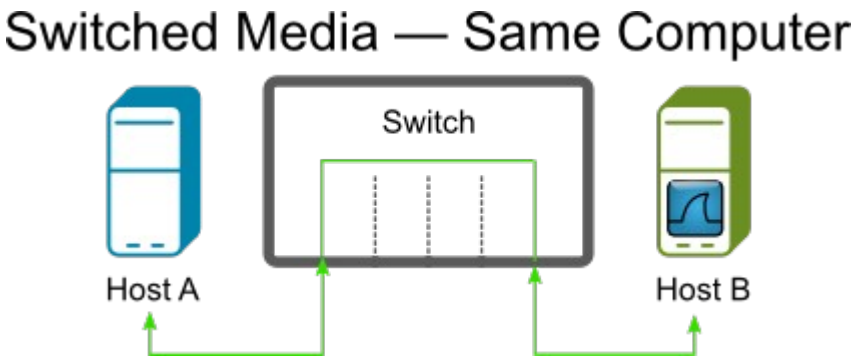
Metody przechwytywania ruchu w Ethernecie

- Sieć przełączana
  - można podsłuchać tylko broadcasty, multicasty i unicasty kierowane do nas



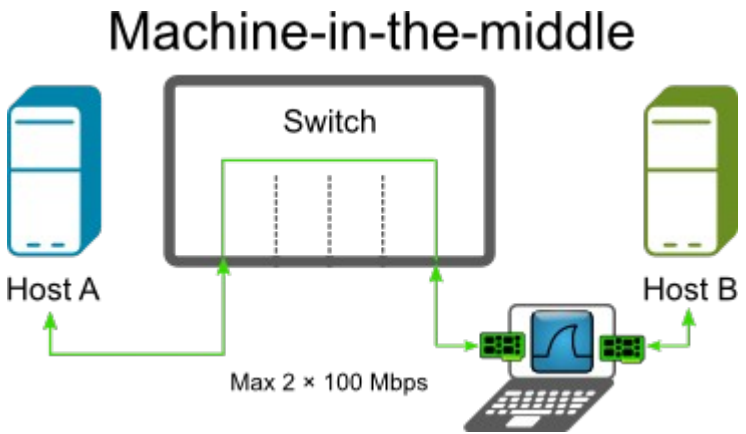
Metody przechwytywania ruchu w Ethernecie

- Przechwytywanie bezpośrednio na maszynie
  - jeśli jest taka możliwość przechwytujmy ruch na źródle lub celu



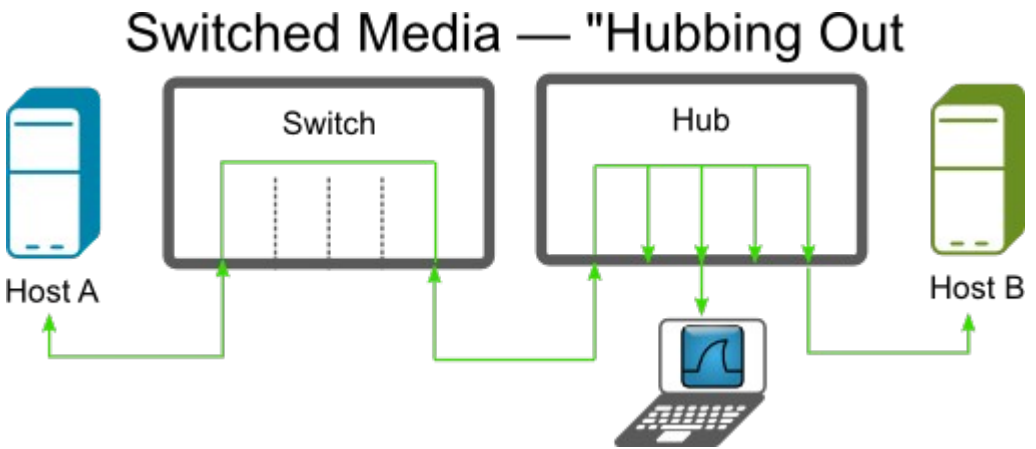
Metody przechwytywania ruchu w Ethernecie

- Przechwytywanie za pośrednictwem maszyny z dwoma interfejsami
  - wady to niewielkie opóźnienie w transmisji oraz fakt, że interfejsy hosta pośredniczącego będą odpowiadały na broadcasty



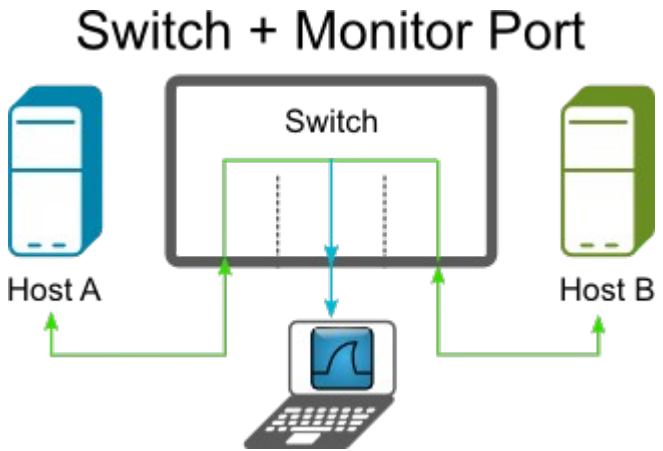
Metody przechwytywania ruchu w Ethernecie

- Przechwytywanie za pośrednictwem dodatkowego huba
  - konieczne rozpięcie i związana z tym przerwa, spadek wydajności sieci



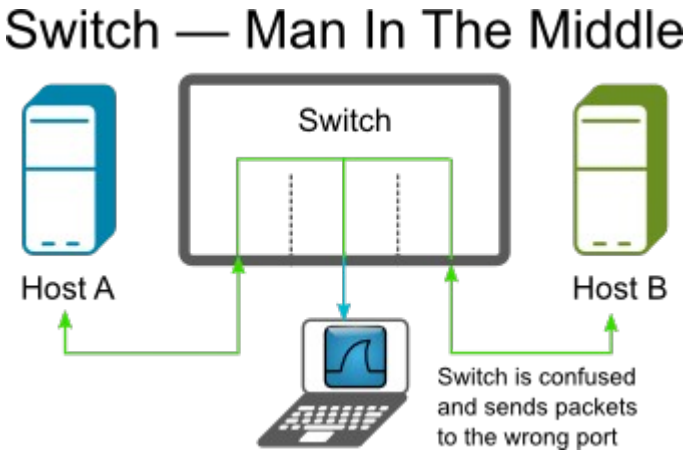
Metody przechwytywania ruchu w Ethernecie

- Przechwytywanie za pomocą trybu port mirroring
  - różne zachowanie u różnych producentów, przekazywany ruch może pochodzić ze wszystkich portów lub tylko określonego



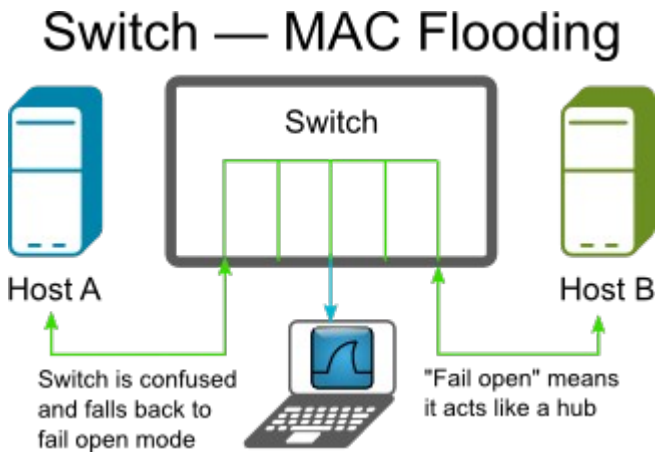
Metody przechwytywania ruchu w Ethernecie

- ARP Poisoning jako metoda przechwytywania ruchu
  - polega na wysyłaniu spreparowanych pakietów ARP do celów ataku tak aby przełącznik przekierował ruch między nimi przez sniffer



Metody przechwytywania ruchu w Ethernecie

- MAC flooding jako metoda przechwytywania ruchu
  - polega na zalewaniu przełącznika fałszywymi adresami MAC doprowadzając do przepełnienia tablicy. Przełącznik przechodzi wtedy w tryb fail-open wysyłając ruch na wszystkie porty (jak hub)



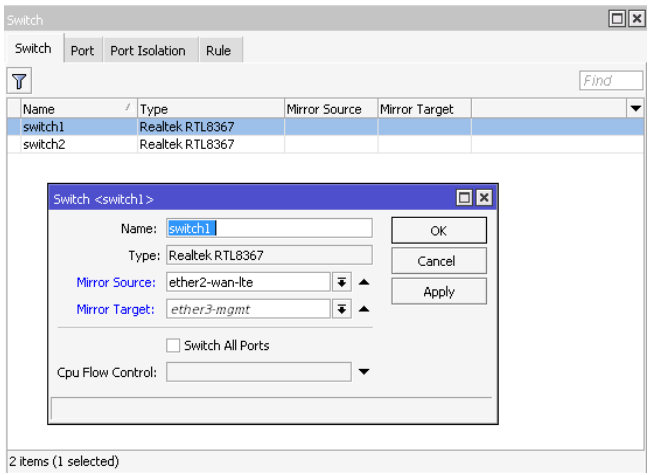
Metody przechwytywania ruchu na MT

- Port mirroring
  - dostępny na ograniczonej ilości urządzeń

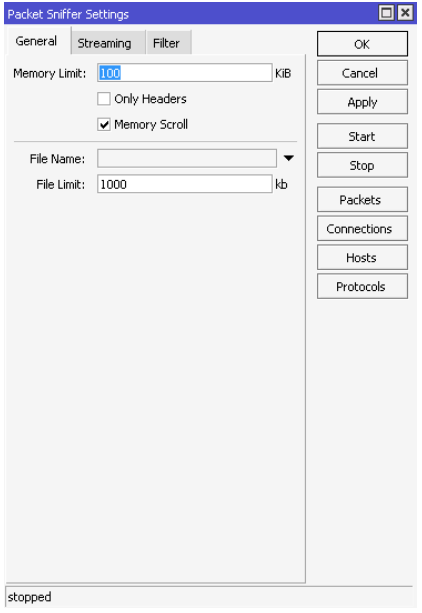
    (https://help.mikrotik.com/docs/spaces/ROS/pages/15302988/Switch+Chip+Features)

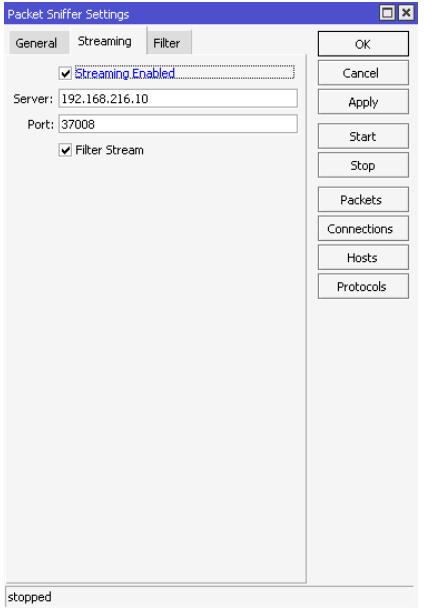
Feature	QCA8337	Atheros8327	Atheros8316	Atheros8227	Atheros7240	IPQ-PPE	ICPlus175D	MT7621, MT7531	RTL8367	88E6393X	88E6191X, 88E6190	98PX1012
Port Mirroring	yes	yes	yes	yes	yes	no	yes	yes	yes	yes	yes	no

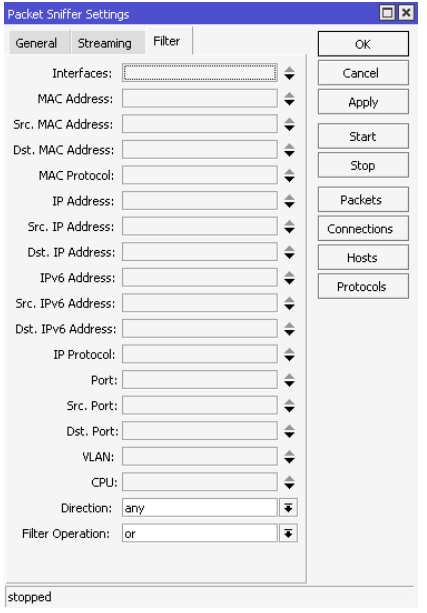
  - pozwala na obserwacje ruchu z określonego portu



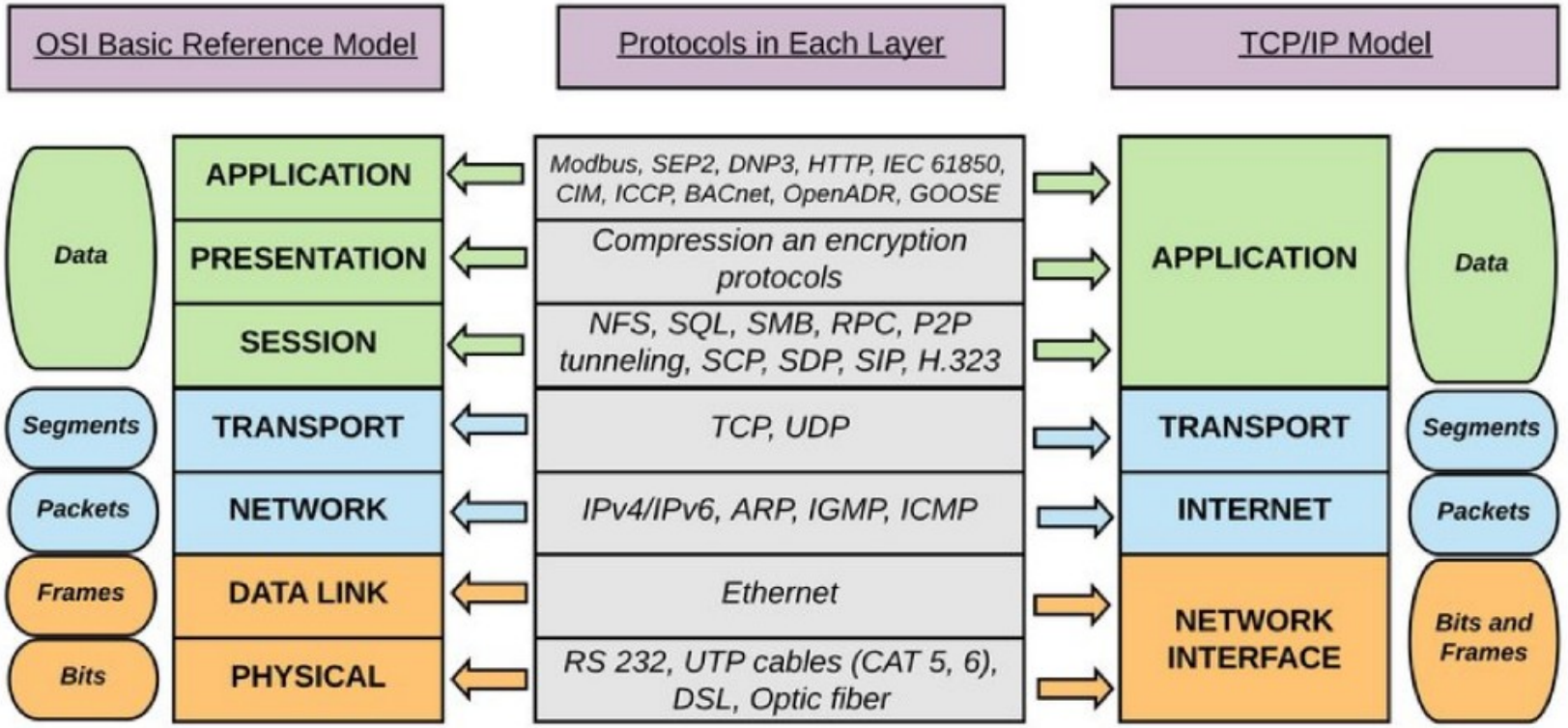
- Packet sniffer + streaming (Tools → Packet Sniffer)
  - nie zobaczymy danych dla połączeń między klientami WLAN jeśli włączona jest opcja client isolation
  - nie zobaczymy części danych jeśli bridge działa z HW offload







Komunikacja sieciowa



Live demo
/
pytania

Dziękuję za uwagę