



# Implementačná dokumentácia k druhému IPK projektu: Sniffer packetov

Veronika Molnárová, xmolna08

23.4.2022

# Obsah

<b>1</b>	<b>Úvod</b>	<b>2</b>
<b>2</b>	<b>Základ problematiky</b>	<b>2</b>
2.1	ARP packet . . . . .	3
2.2	IP packet . . . . .	3
2.2.1	ICMP packet . . . . .	4
2.2.2	TCP packet . . . . .	4
2.2.3	UDP packet . . . . .	5
<b>3</b>	<b>Implementácia projektu</b>	<b>5</b>
3.1	Parsovanie argumentov . . . . .	5
3.2	Získanie a filtácia packetov . . . . .	5
3.3	Rozbaľovanie packetov . . . . .	6
<b>4</b>	<b>Testovanie</b>	<b>6</b>
<b>5</b>	<b>Použitá literatúra</b>	<b>7</b>

# 1 Úvod

Cieľom projektu bolo vytvoriť jednoduchý packet sniffer implementovaný v jazyku C. Programu je možné špecifikovať dané rozhranie, port, vybraný typ packetov a počet packetov, ktoré má program vyhľadať a ich informácie vypísať na štandardný výstup. V prípade nastania vnútornej chyby je vypísaná chybová hláška a program ukončený. Beh programu je tiež možné ukončiť SIGINT signálom, ktorý preruší beh programu.

## 2 Základ problematiky

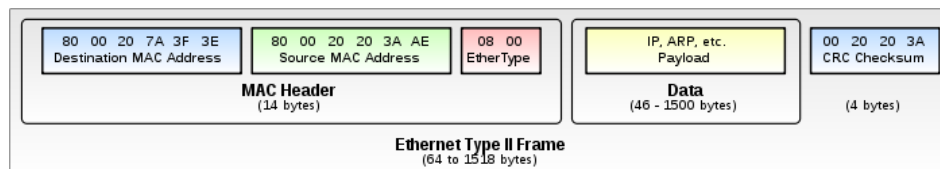
V našom projekte bolo možné špecifikovať nasledujúce typy packetov, ktoré mohol program odchytať:

- ARP packet (sieťová vrstva)
- ICMP packet (sieťová vrstva)
- TCP packet (transportná vrstva)
- UDP packet (transportná vrstva)

Pričom program podporuje IPv4 aj IPv6 protokol.

Základný koncept projektu pozostáva z odchyťovania jednotlivých packetov prichádzajúcich na dané rozhranie, ich následnej filtrácie podľa špecifikácie typov packetov na odchyťovanie a neposledne získavania enkapsulovaných dát z odchyteného packetu a vypísania jeho obsahu na štandardný výstup.

Odchytený datagram získaváme ako frame vrstvy sieťového rozhrania. Môžeme ho vidieť na nasledujúcom obrázku.

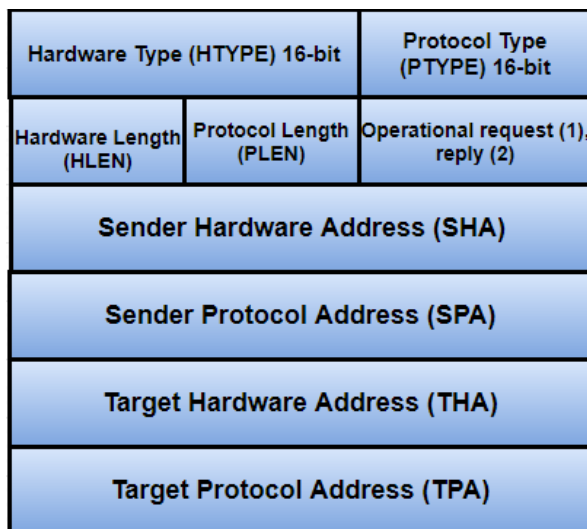


Z daného framu je potrebné oddelenie jeho hlavičky, z ktorej sa vyčíta MAC adresa zdrojového aj cieľového zariadenia. Posledná získaná informácia z ethernetovej hlavičky je ethernetový typ, podľa ktorého dokážeme rozlíšiť, aký typ packetu sa nachádza v payloade daného frame-u.

Po rozbalení daného payloadu môžeme získať nasledujúce packety:

## 2.1 ARP packet

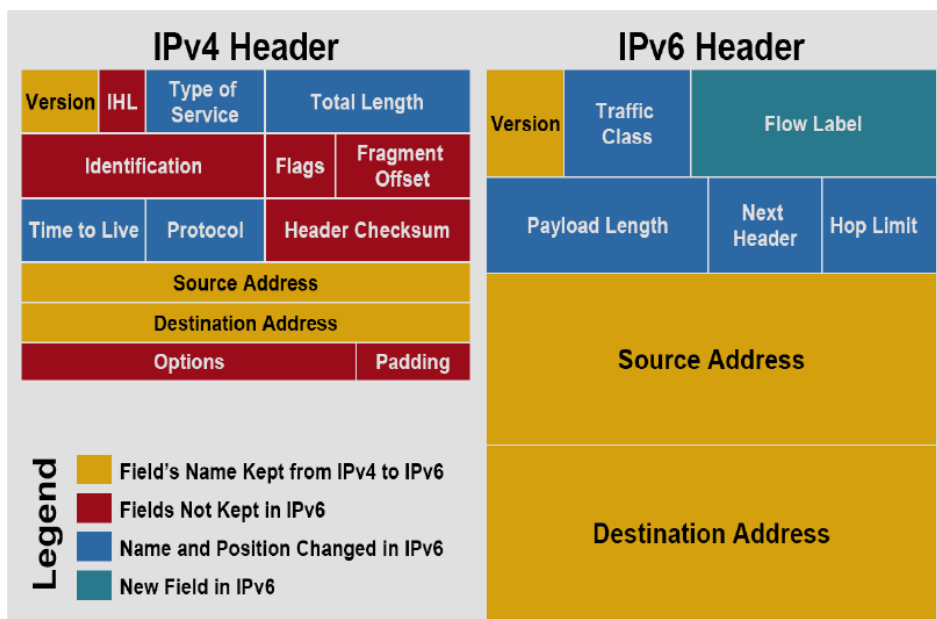
Štruktúra ARP packetu vyzerá nasledovne:



Tieto druhy packetov sa používajú napr. pri identifikácii pripojených zariadení v lokálnej sieti, kedy po zadaní IP adresy sa zariadenie snaží túto adresu pingnúť čakajúc, či niektoré zo zariadení odpovie. Z daného packetu teda vieme zistiť IP adresy cieľového aj zdrojového zariadenia.

## 2.2 IP packet

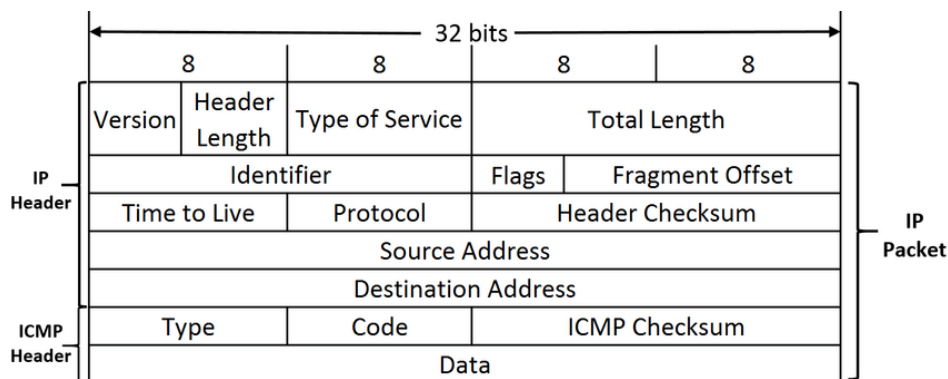
Pri IP packetoch môžeme naraziť na ich IPv4 alebo IPv6 verziu, ktorých hlavičky sa líšia.



IPv4 packet má premennú veľkosť packetu, zatiaľ čo IPv6 packet ju má presne danú. Z daných hlavičiek si vieme skontrolovať správnosť verzie, získať dané IP adresy zdrojového aj cieľového zariadenia aj typ IP packetu podľa daného protokolu (next header pri IPv6). Pri IPv4 packete je potrebné zistiť aj veľkosť danej hlavičky z IHL (počet dvojslov), aby sme vedeli povedať, kde začína payload. Podľa protokolu IP packetu môžeme rozlíšiť ICMP, UDP a TCP datagramy.

### 2.2.1 ICMP packet

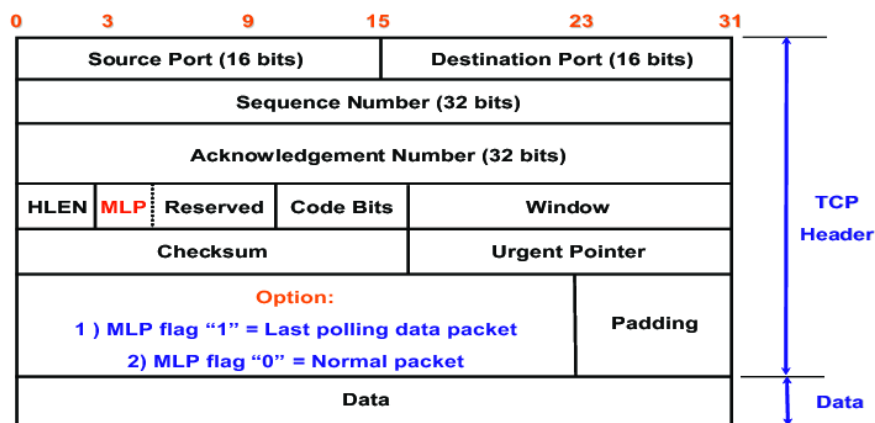
Ako môžeme vidieť z obrázku, ICMP packet obsahuje ďalšiu hlavičku hneď za IP hlavičkou.



Tieto packety majú jednoduchú štruktúru a sú využívané napr. na pingovanie.

### 2.2.2 TCP packet

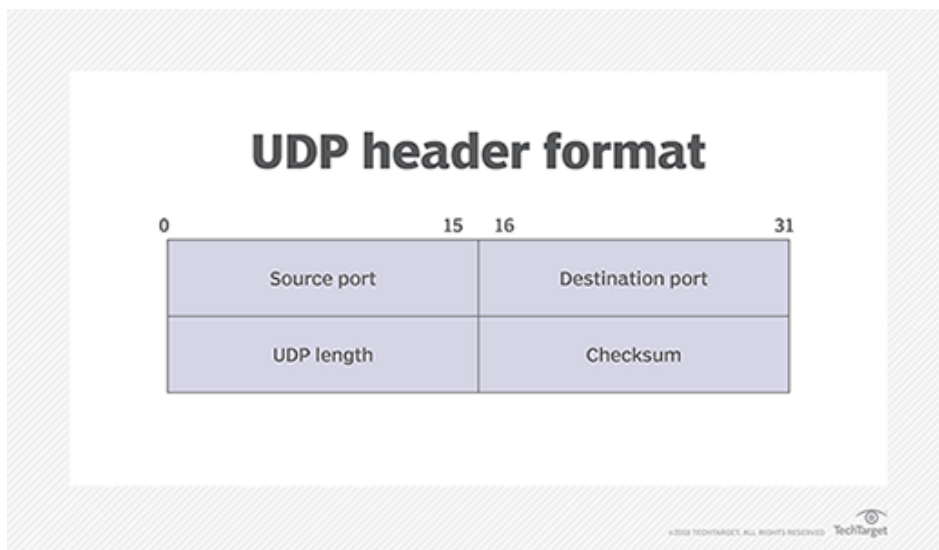
Podsledná informácia, ktorú z datagramov potrebujeme získať, sú dané porty, na ktorých prebieha komunikácia.



Tak tomu je pri TCP datagramoch, ktoré dokážu prenášať správy z jedného zariadenia na druhé. Ich hlavičky nemajú pevnú dĺžku a sú značne obsiahle.

### 2.2.3 UDP packet

Druhý spôsob posielania správ je pomocou UDP datagramov. Tie sú značne jednoduchšie, keďže ponúkajú menej spoľahlivú komunikáciu medzi zariadeniami.



Obsahujú zdrojový a cieľový port, na ktorých prebieha daná komunikácia.

## 3 Implementácia projektu

Pri implementácii projektu bolo využitých množstvo knižníc obsahujúcich predovšetkým jednotlivé štruktúry pre hlavičky datagramov. Program vieme rozdeliť do niekoľkých segmentov.

### 3.1 Parsovanie argumentov

Pre parsovanie argumentov programu bola použitá štruktúra `option` a funkcia `getopt_long()` z knižnice `getopt.h`. Danej funkcii sú predané jednotlivé možnosti dlhých a krátkych argumentov a flag, či obsahujú aj hodnotu. Funkcia vracia daný znak podľa práve načítaného argumentu spoločne s hodnotou argumentu v štruktúre `optarg`. V prípade neznámeho argumentu je vypísaná nápoveda užívateľovi.

### 3.2 Získanie a filtrácia packetov

V prípade nezadanie rozhrania sú užívateľovi na výstup vypísané všetky aktívne rozhrania a program je ukončený. Pre tento účel je využitá funkcia `pcap_findalldevs()` z knižnice `pcap.h`, ktorá vracia viazaný zoznam nájdených rozhraní.

V prípade, že je zadané rozhranie sa môže nadviazať spojenie s daným rozhraním a začať filtrácia a získavanie packetov.

Prvotne sa pokúsi získať maska na dané rozhranie pomocou `pcap_lookupnet()` a nadviaže sa spojenie s daným rozhraním pomocou `pcap_open_live()`. Takto získame tzv. handle, pomocou ktorého sa uskutočňuje komunikácia s daným rozhraním. Jednotlivé packety sa odchyťávajú pomocou funkcie `pcap_loop()`, ktorému sa špecifikuje počet packetov, ktorý sa má odchytiť a callback funkcia volaná na jednotlivé packety. Po ukončení práce s packetmi sa handle uzavrie pre správne ukončenie komunikácie s rozhraním.

Pre filtráciu packetov sa vytára štruktúra `bpf_program`, ktorá vzniká na základe filtrujúceho výrazu. Ten sa kompiluje cez funkciu `pcap_compile()` a aplikuje na daný handle pomocou `pcap_setfilter()`.

### 3.3 Rozbaľovanie packetov

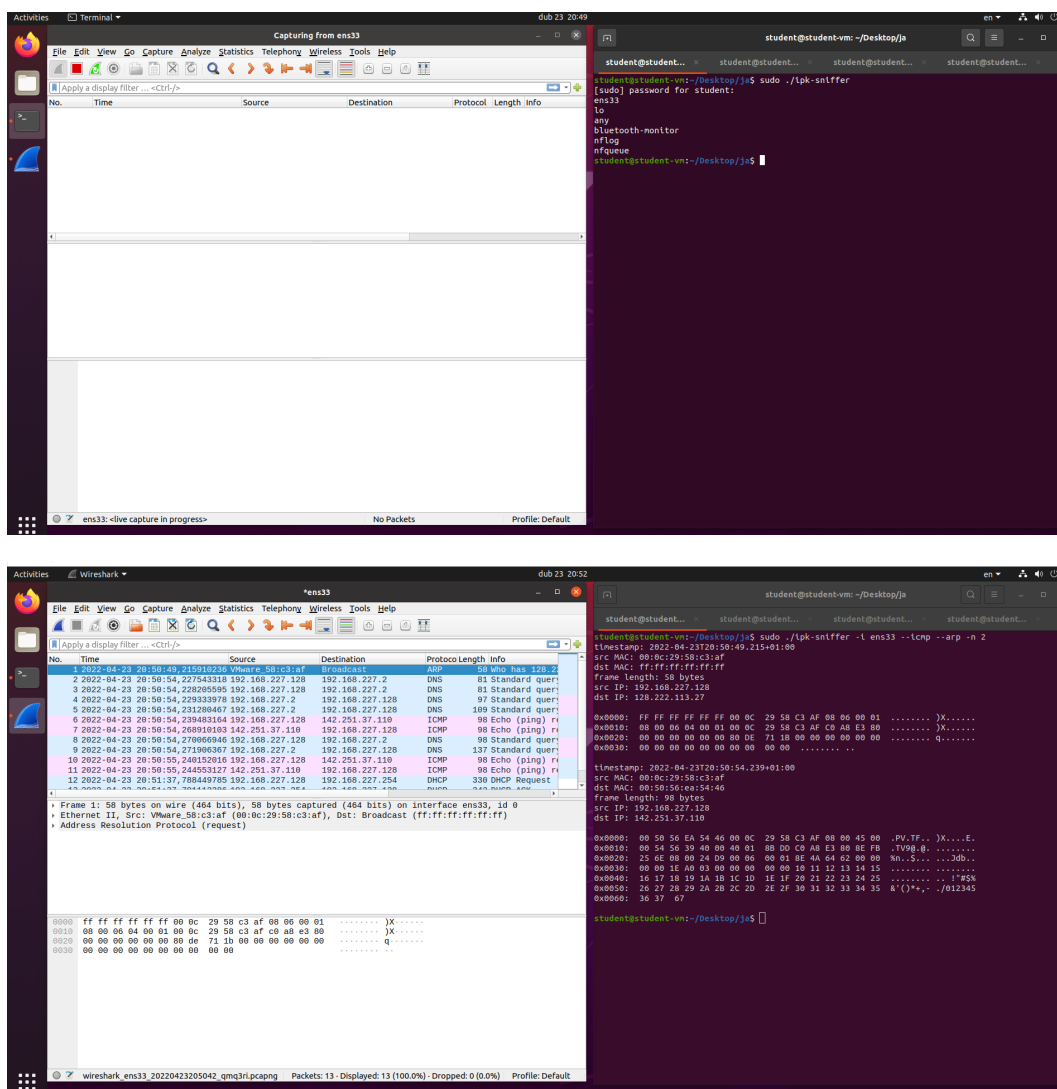
Rozbaľovanie získaného datagramu prebieha postupným odstraňovaním hlavičiek, získavaním informácií z nich a zvyšné dáta sú posielané na ďalšiu úroveň odstraňovania hlavičiek. Sú využívané štruktúry `ether_header`, `ip6_hdr`, `ip`, `ether_arp` a `tcphdr`.

Jednotlivé adresy sú často uložené s kódovaním big endian a v prípade vypisovania je ich potreba previesť pre účely jazyka C funkciou `ntohs()`. Spoločne s počiatočným datagramom je tiež získaná hlavička s časom odoslania packetu a jeho veľkosťou v bytoch.

## 4 Testovanie

Pri testovaní sme využívali nástroj Wireshark, ktorý dokáže odchyťovať jednotlivé packety na danom rozhraní a zobraziť ich obsah. Testovanie prebiehalo na referenčnom virtuálnom stroji, ktorý nám bol poskytnutý pre účely projektu. Spočívalo predovšetkým v porovnávaní informácií získaných pomocou nášho packet snifferu s informáciami z Wiresharku.

Následne je priložených pár screenshotov z testovania odchyťovania packetov. Môžeme vidieť vedľa seba porovnávané informácie ako timestamp packetu, jeho veľkosť v bytoch, MAC a IP adresy a ak sú prítomné tak aj jednotlivé porty. Ďalej vidíme obsah celých frame-ov a v prehľade môžeme vidieť aj úspešnú filtráciu packetov, kedy sú ignorované nesprávne typy packetov.



The terminal window shows the following commands and output:

```
student@student-vm: ~/Desktop/ja
student@student-vm: ~/Desktop/ja$ sudo ./tpk-sniffer -i ens33 --arp -n 2
timestamp: 2022-04-23T20:50:49.215+01:00
src MAC: 00:0c:29:58:c3:af
dst MAC: ff:ff:ff:ff:ff:ff
frame length: 58 bytes
src IP: 192.168.227.128
dst IP: 192.222.113.27
0x0000: ff ff ff ff ff ff 00 0c 29 58 c3 af 08 06 00 01 .....XJ.....
0x0010: 08 06 04 00 01 00 0c 29 58 c3 af c0 a8 e3 80 .....TV98@.....
0x0020: 00 00 00 00 00 00 00 00 71 18 00 00 00 00 00 00 .....XJ.....
0x0030: 00 00 00 00 00 00 00 00 .....
timestamp: 2022-04-23T20:50:54.239+01:00
src MAC: 00:0c:29:58:c3:af
dst MAC: 00:56:56:ea:54:46
frame length: 98 bytes
src IP: 192.168.227.128
dst IP: 142.251.37.110
0x0000: 00 56 56 ea 54 46 00 0c 29 58 c3 af 08 06 45 00 .....PV.TF...XJ....E.
0x0010: 08 06 04 00 00 01 00 0c 29 58 c3 af c0 a8 e3 80 .....TV98@.....
0x0020: 25 6c 08 00 24 d9 00 00 01 8e 4a 64 62 00 00 00 .....Xn.S...3db.
0x0030: 00 00 1e ad 03 00 00 00 00 10 11 12 13 14 15 .....
0x0040: 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 ...../*$%
0x0050: 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 .....&()*+,-./012345
0x0060: 36 37 67
```

The Wireshark capture shows the following traffic:

No.	Time	Source	Destination	Protocol	Length	Info
1	2022-04-23 20:50:49.215910236	Vmware_58:c3:af	Broadcast	ARP	58	who has 128.2...
2	2022-04-23 20:50:54.227543318	192.168.227.128	192.168.227.2	DNS	81	Standard query 0x3...
3	2022-04-23 20:50:54.228305096	192.168.227.128	192.168.227.2	DNS	81	Standard query 0x7f...
4	2022-04-23 20:50:54.229333978	192.168.227.2	192.168.227.128	DNS	97	Standard query resp...
5	2022-04-23 20:50:54.231284467	192.168.227.2	192.168.227.128	DNS	109	Standard query resp...
6	2022-04-23 20:50:54.23153164	192.168.227.128	192.168.227.2	ICMP	98	Echo (ping) reques...
7	2022-04-23 20:50:54.236910183	142.251.37.110	192.168.227.128	ICMP	98	Echo (ping) reply
8	2022-04-23 20:50:54.270666946	192.168.227.128	192.168.227.2	DNS	98	Standard query 0xa...
9	2022-04-23 20:50:54.271906367	192.168.227.2	192.168.227.128	DNS	137	Standard query resp...
10	2022-04-23 20:50:55.248152916	192.168.227.128	142.251.37.110	ICMP	98	Echo (ping) reques...
11	2022-04-23 20:50:55.24453127	142.251.37.110	192.168.227.128	ICMP	98	Echo (ping) reply
12	2022-04-23 20:51:37.788449785	192.168.227.128	192.168.227.254	DHCP	330	DHCP Request

The packet details for the first packet (ARP) show:

- Frame 0: 98 bytes on wire (784 bits), 98 bytes captured (784 bits) on interface ens33, id 0
- Ethernet II, Src: Vmware\_58:c3:af (00:0c:29:58:c3:af), Dst: Vmware\_ea:54:46 (00:50:56:ea:54:46)
- Internet Protocol Version 4, Src: 192.168.227.128, Dst: 142.251.37.110
- Internet Control Message Protocol

The packet bytes show:

```
0000 00 50 56 ea 54 46 00 0c 29 58 c3 af 08 06 45 00 .....PV.TF...XJ....E.
0010 08 06 04 00 00 01 00 0c 29 58 c3 af c0 a8 e3 80 .....TV98@.....
0020 25 6c 08 00 24 d9 00 00 01 8e 4a 64 62 00 00 00 .....Xn.S...3db.
0030 00 00 1e ad 03 00 00 00 00 10 11 12 13 14 15 .....
0040 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 ...../*$%
0050 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 .....&()*+,-./012345
0060 36 37
```

The terminal window shows the following commands and output:

```
student@student-vm: ~/Desktop/ja$ sudo ./tpk-sniffer -i ens33 -u
timestamp: 2022-04-23T20:51:57.884+01:00
src MAC: 00:0c:29:58:c3:af
dst MAC: 00:56:56:ea:54:46
frame length: 81 bytes
src IP: 192.168.227.128
dst IP: 192.168.227.2
src port: 54936
dest port: 53
0x0000: 00 56 56 ea 54 46 00 0c 29 58 c3 af 08 06 45 00 .....PV.TF...XJ....E.
0x0010: 08 06 04 00 00 01 00 0c 29 58 c3 af c0 a8 e3 80 .....TV98@.....
0x0020: e3 02 06 98 00 35 00 2f 48 15 38 59 01 00 00 01 .....S/H;Y...
0x0030: 00 00 00 00 00 01 06 07 6f 6f 67 6c 65 03 63 6f .....g oogle co
0x0040: 6d 00 00 01 00 01 00 00 29 02 00 00 00 00 00 00 .....d oogle co
0x0050: 00
```

The Wireshark capture shows the following traffic:

No.	Time	Source	Destination	Protocol	Length	Info
1	2022-04-23 20:53:57.884183130	192.168.227.128	192.168.227.2	DNS	81	Standard query 0x3...
2	2022-04-23 20:53:57.884484524	192.168.227.128	192.168.227.2	DNS	81	Standard query 0x7f...
3	2022-04-23 20:53:57.885083819	192.168.227.2	192.168.227.128	DNS	97	Standard query resp...
4	2022-04-23 20:53:57.88598448	192.168.227.2	192.168.227.128	DNS	109	Standard query resp...
5	2022-04-23 20:53:57.88615066	192.168.227.128	142.251.37.110	ICMP	98	Echo (ping) reques...
6	2022-04-23 20:53:57.89806056	142.251.37.110	192.168.227.128	ICMP	98	Echo (ping) reply
7	2022-04-23 20:53:57.891081517	192.168.227.128	192.168.227.2	DNS	98	Standard query 0xa...
8	2022-04-23 20:53:57.892255209	192.168.227.2	192.168.227.128	DNS	137	Standard query resp...
9	2022-04-23 20:54:03.853098470	Vmware_58:c3:af	Vmware_ea:54:46	ARP	42	who has 192.168.227.2 is at
10	2022-04-23 20:54:03.85492545	Vmware_ea:54:46	Vmware_58:c3:af	ARP	60	192.168.227.2 is at

The packet details for the first packet (DNS) show:

- Frame 1: 81 bytes on wire (648 bits), 81 bytes captured (648 bits) on interface ens33, id 8
- Ethernet II, Src: Vmware\_58:c3:af (00:0c:29:58:c3:af), Dst: Vmware\_ea:54:46 (00:50:56:ea:54:46)
- Internet Protocol Version 4, Src: 192.168.227.128, Dst: 192.168.227.2
- User Datagram Protocol, Src Port: 54936, Dst Port: 53
- Domain Name System (query)

The packet bytes show:

```
0000 00 50 56 ea 54 46 00 0c 29 58 c3 af 08 06 45 00 .....PV.TF...XJ....E.
0010 08 06 04 00 00 01 00 0c 29 58 c3 af c0 a8 e3 80 .....TV98@.....
0020 e3 02 06 98 00 35 00 2f 48 15 38 59 01 00 00 01 .....S/H;Y...
0030 00 00 00 00 00 01 06 07 6f 6f 67 6c 65 03 63 6f .....g oogle co
0040 6d 00 00 01 00 01 00 00 29 02 00 00 00 00 00 00 .....d oogle co
0050 00
```

The terminal window shows the following commands and output:

```
student@student-vm: ~/Desktop/ja$ sudo ./tpk-sniffer -i lo -p 8080 --udp --tcp
timestamp: 2022-04-23T21:04:52.658+01:00
src MAC: 00:00:00:00:00:00
dst MAC: 00:00:00:00:00:00
frame length: 102 bytes
src IP: ::1
dst IP: ::1
src port: 55416
dest port: 8080
0x0000: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x0010: 86 f1 00 30 00 00 00 00 00 00 00 00 00 00 00 00 .....
0x0020: 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 00 .....
0x0030: 00 00 00 00 00 01 00 70 1f 9b ce 0c 99 63 f3 d2 .....X.....C...
0x0040: 5d 07 80 18 02 00 00 38 00 00 01 01 00 0a 1c 72 .....8.....r
0x0050: 36 5d 1c 70 80 09 70 6f 73 09 65 6c 61 60 20 70 .....p.tpo sielan p
0x0060: 61 63 08 65 74 0a .....ack
```

The Wireshark capture shows the following traffic:

No.	Time	Source	Destination	Protocol	Length	Info
1	2022-04-23 21:04:52.658487278	:::1	:::1	TCP	112	55416 -> 8080 [PSH]
2	2022-04-23 21:04:52.658497591	:::1	:::1	TCP	66	8080 -> 55416 [ACK]

The packet details for the first packet (TCP) show:

- Frame 1: 102 bytes on wire (816 bits), 102 bytes captured (816 bits) on interface lo, id 0
- Ethernet II, Src: 00:00:00:00:00:00 (00:00:00:00:00:00), Dst: 00:00:00:00:00:00 (00:00:00:00:00:00)
- Internet Protocol Version 6, Src: ::1, Dst: ::1
- Transmission Control Protocol, Src Port: 55416, Dst Port: 8080, Seq: 1, Ack: 1, Len: 16

The packet bytes show:

```
0000 00 00 00 00 00 00 00 00 00 00 00 00 86 dd 00 07 .....
0010 86 f1 00 30 00 00 00 00 00 00 00 00 00 00 00 00 .....
0020 00 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 .....
0030 00 00 00 00 00 01 d0 70 1f 9b ce 0c 99 63 f3 d2 .....X.....C...
0040 5d 07 80 18 02 00 00 38 00 00 01 01 00 0a 1c 72 .....8.....r
0050 36 5d 1c 70 80 09 70 6f 73 09 65 6c 61 60 20 70 .....p.tpo sielan p
0060 61 63 08 65 74 0a .....ack
```

## 5 Použitá literatura

- Použité stránky pri tvorení programu:  
<https://www.tcpdump.org/pcap.html>



<https://www.devdungeon.com/content/using-libpcap-c>  
<https://www.binarytides.com/category/general/>

- Študentský stream od Tedro:

<https://www.youtube.com/watch?v=-tR2XqAZm0>

- Riešenia problémov počas implementácie a referenčné stránky:

<https://stackoverflow.com/>

<https://www.wikipedia.org/>

<https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml>

<https://docs.huihoo.com/doxygen/linux/kernel/3.7/structip6hdr.html>

<https://sites.uclouvain.be/SysInfo/usr/include/netinet/ip.h.html>