SLOVENSKÁ TECHNICKÁ UNIVERZITA
Fakulta informatiky a informačních technologií
Ilkovičova 2, 842 16 Bratislava 4

Analyzátor sieťovej komunikácie

Dávid Penťa ID: 110871
Počítačové a komunikačnné siete
FIIT STU 2021/2022

Zadanie úlohy

Navrhnite a implementujte programový analyzátor Ethernet siete, ktorý analyzuje komunikácie v sieti zaznamenané v .pcap súbore a poskytuje nasledujúce informácie o komunikáciách.
Vypracované zadanie musí spĺňať nasledujúce body:
1) **Výpis všetkých rámcov v hexadecimálnom tvare** postupne tak, ako boli zaznamenané v súbore.

Pre každý rámec uveďte:
a) Poradové číslo rámca v analyzovanom súbore.

b) Dĺžku rámca v bajtoch poskytnutú pcap API, ako aj dĺžku tohto rámca prenášaného po médiu.

c) Typ rámca – Ethernet II, IEEE 802.3 (IEEE 802.3 s LLC, IEEE 802.3 s LLC a SNAP, IEEE 802.3 – Raw).

d) Zdrojovú a cieľovú fyzickú (MAC) adresu uzlov, medzi ktorými je rámec prenášaný.

Vo výpise jednotlivé **bajty rámca usporiadajte po 16 alebo 32 v jednom riadku**. Pre prehľadnosť výpisu je vhodné použiť neproporcionálny (monospace) font.
2) Pre rámce typu **Ethernet II a IEEE 802.3 vypíšte vnorený protokol**. Študent musí vedieť vysvetliť, aké informácie sú uvedené v jednotlivých rámcoch Ethernet II, t.j. vnáranie protokolov ako aj ozrejmiť dĺžky týchto rámcov.

3) Analýzu cez vrstvy vykonajte pre rámce Ethernet II a protokoly rodiny TCP/IPv4:

Na konci výpisu z bodu 1) uveďte pre IPv4 pakety:
a) Zoznam IP adries všetkých odosielajúcich uzlov,

b) IP adresu uzla, ktorý sumárne odoslal (bez ohľadu na prijímateľa) najväčší počet paketov a koľko paketov odoslal (berte do úvahy iba IPv4 pakety).

4) V danom súbore analyzujte komunikácie pre zadané protokoly:

a) HTTP

b) HTTPS

c) TELNET

d) SSH

e) FTP riadiace
f) FTP dátové

g) TFTP, **uveďte všetky rámce komunikácie**, nielen prvý rámec na UDP port 69

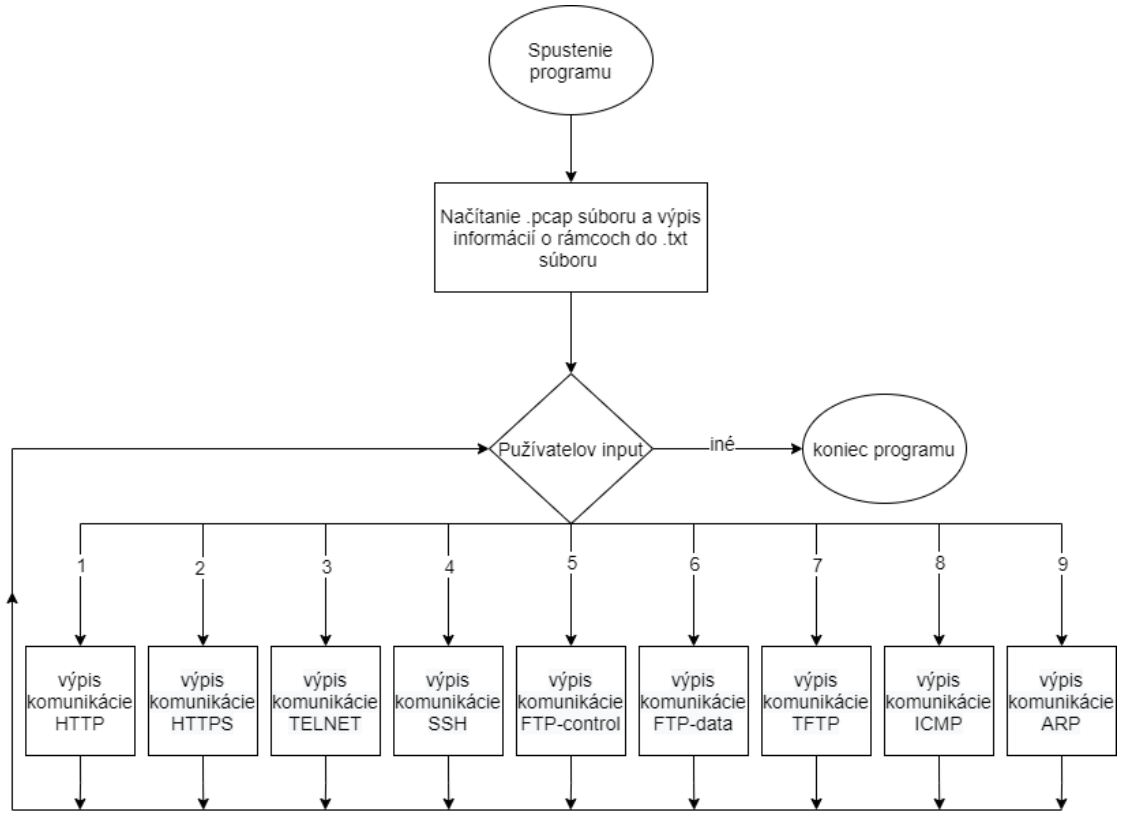
h) ICMP, uveďte aj typ ICMP správy (pole Type v hlavičke ICMP), napr. Echo request, Echo reply, Time exceeded, a pod.

i) **Všetky** ARP dvojice (request – reply), uveďte aj IP adresu, ku ktorej sa hľadá MAC (fyzická) adresa a pri ARP-Reply uveďte konkrétny pár - IP adresa a nájdená MAC adresa. V prípade, že bolo poslaných viacero rámcov ARP-Request na rovnakú IP adresu, vypíšte všetky. Ak sú v súbore rámce ARP-Request bez korešpondujúceho ARP-Reply (alebo naopak ARP-Reply bez ARP-Request), vypíšte ich samostatne.

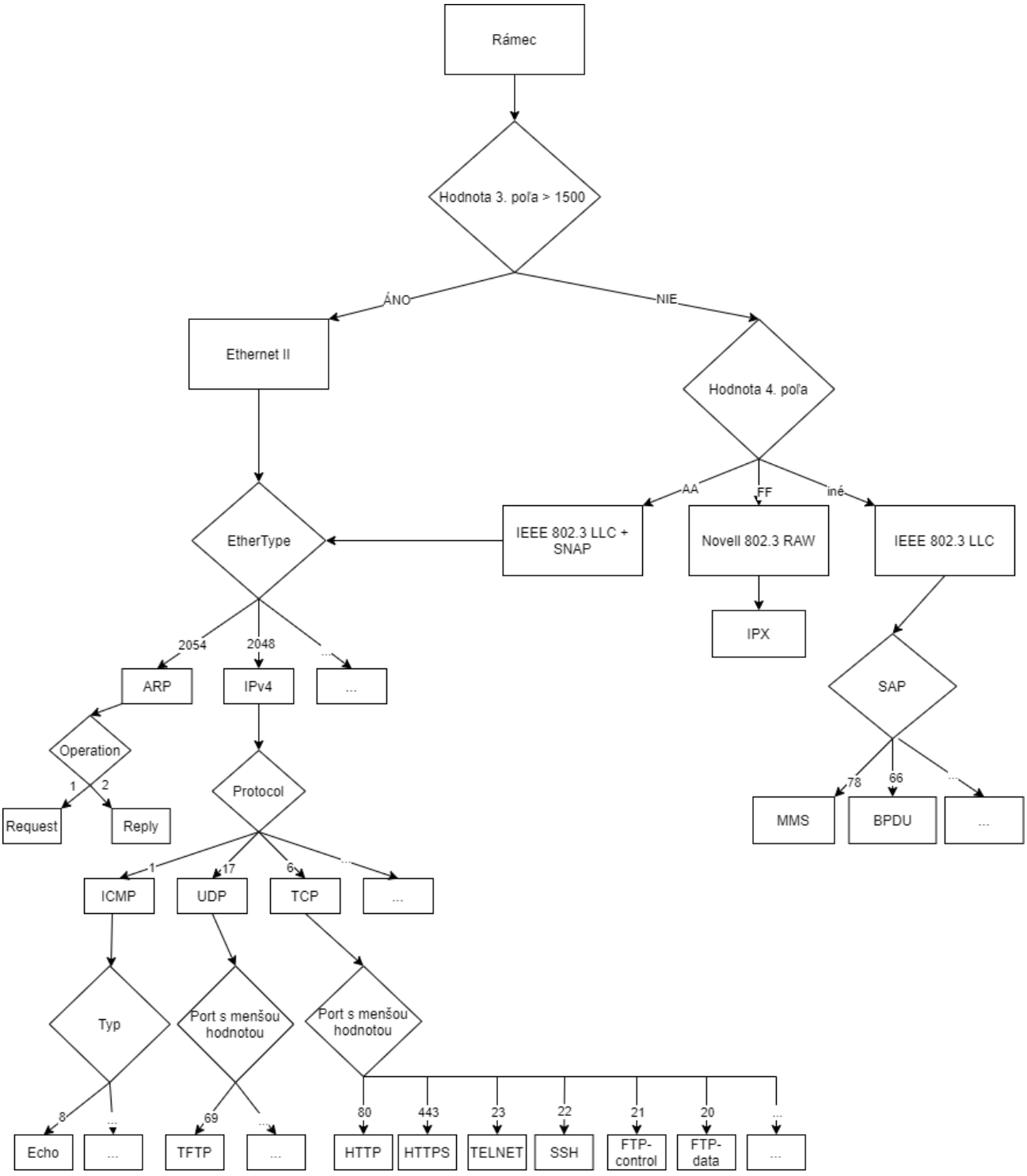
Vo všetkých výpisoch treba uviesť aj IP adresy a pri transportných protokoloch TCP a UDP aj porty komunikujúcich uzlov.
V prípadoch komunikácií so spojením vypíšte iba jednu kompletnú komunikáciu - obsahuje otvorenie (SYN) a ukončenie (FIN na oboch stranách alebo ukončenie FIN a RST alebo ukončenie iba s RST) spojenia a aj prvú nekompletnú komunikáciu, ktorá obsahuje iba otvorenie spojenia. Pri výpisoch vyznačte, ktorá komunikácia je kompletná.
Ak počet rámcov komunikácie niektorého z protokolov z bodu 4 je väčší ako 20, vypíšte iba 10 prvých a 10 posledných rámcov tejto komunikácie. **(Pozor: toto sa nevzťahuje na bod 1, program musí byť schopný vypísať všetky rámce zo súboru podľa bodu 1.)** Pri všetkých výpisoch musí byť poradové číslo rámca zhodné s číslom rámca v analyzovanom súbore.
5) Program musí byť organizovaný tak, aby čísla protokolov v rámci Ethernet II (pole Ethertype), IEEE 802.3 (polia DSAP a SSAP), v IP pakete (pole Protocol), ako aj čísla portov v transportných protokoloch boli programom **načítané z jedného alebo viacerých externých textových súborov.** Pre známe protokoly a porty (minimálne protokoly v bodoch 1) a 4) budú uvedené aj ich názvy. Program bude schopný uviesť k rámcu názov vnoreného protokolu po doplnení názvu k číslu protokolu, resp. portu do externého súboru. Za externý súbor sa nepovažuje súbor knižnice, ktorá je vložená do programu.

Blokový návrh (koncepcia) fungovania riešenia

Ovládanie programu používateľom



Princíp určovanie protokolov rámca



Navrhnutý mechanizmus analyzovania protokolov na jednotlivých vrstvách

Úloha 1

Pri spustení si programu načíta .pcap súbor s rámcami a súbor, do ktorého vypíše úlohy 1-3. Tieto tri úlohy sa vykonajú automaticky pri spustení programu. Rámce zo súboru načíta pomocou knižnice scapy. Z tejto knižnice používa len príkazy scapy.rdpcap() pre načítanie .pcap súboru a scapy.raw(), aby rámce mohol spracovať postupne po bajtoch. Program do výstupového súboru vypíše dĺžku rámca poskytnutú pcap API, dĺžku rámca prenášaného po médiu a nájdenú zdrojovú a cieľovú MAC adresu. Potom program skontroluje hodnotu 3. poľa rámca. Ak nájdená hodnota je väčšia ako 1500, tak sa jedná o typ Ethernet II. Ak je menšia tak sa pozriem na 4. pole rámca, a ak sa tam nachádza hexadecimálna hodnota aa tak sa jedná o rámec typu IEEE 802.3 - LLC + SNAP, ak sa tam nachádza hexadecimálna hodnota ff tak sa jedná o IEEE 802.3 - "Raw ". Ak sa tam nachádza niečo iné, tak potom to je typ rámca IEEE 802.3-LLC.

Úloha 2

Pre všetky rámce typu IEEE 802.3 - "Raw " program vypíše ako vnorený protokol IPX. Ak sa jedná o rámec typu IEEE 802.3 – LLC, tak sa pozrie do polí DSAP a SSAP, a zistí SAP. Zistí ho tak, že sa nájdená hodnota prepočíta na desiatkovú sústavu a skúsi ju nájsť v textovom súbore SAPs.txt. Ak ju tam nenájde, tak vypíše, že takú hodnotu nepozná. Ak sa jedná o rámec typu Ethernet II alebo IEEE 802.3 - LLC + SNAP, tak sa pozrie na hodnotu poľa EtherType, prepočíta ju na dekadickú sústavu a skúsi ju nájsť v súbore EtherTypes.txt. Ak ju tam nenájde, tak vypíše, že takú hodnotu nepozná.

Úloha 3

Ak program nájde v rámci protokol IPv4, tak vypíše zdrojovú a cieľovú IP adresu a skúsi nájsť protokol v ňom pomocou textového súboru IPv4_Protocols.txt. Zdrojové adresy pridáva do slovníka SourceIPAddresses ako kľúč a ich počet ako ich hodnotu. Ak sa tam už IP adresa nachádza v slovníku, tak jej len zvýši hodnotu. Na koniec výstupového súboru vypíše zo slovníka všetky zdrojové IP adresy a nájde najčastejšie sa nachádzajúcu, ku ktorej vypíše aj počet.

Úloha 4a, 4b, 4c, 4d, 4e, 4f

Program pri nájdení IPv4 protokolu pokračuje a hľadá v ňom vnorený protokol. Ak to je TCP, tak program vypíše v ňom nájdené porty. Podľa portu s nižšou hodnotou program z externého textového súboru zistí, o akú komunikáciu sa jedná, napríklad ak port bude mať dekadickú hodnotu 80, tak patrí do http komunikácie. Ak nájde port: http - 80, https - 443, telnet - 23, ssh - 22, ftp-control -21 a ftp-data -20, tak program použije triedu TCP, a priradí ho do správnej komunikácie. To vykoná pomocou súboru

TCP_Ports.txt. Komunikácie sú vytvorené pomocou triedy TCP_communication. Týchto šesť typov TCP komunikácií si pamätá v šiestich poliach, ako napríklad v HTTP_communications_list. Komunikácia obsahuje rámce s rovnakými IPv4 adresami a portami. Prvý rámec TCP komunikácie je SYN a má hodnotu flagu 2. Druhý je SYN, ACK a má hodnotu flagu 18. Tretí je ACK a má hodnotu flagu 16. Aby bola TCP komunikácia kompletná, môže skončiť viacerými spôsobmi. Prvý spôsob je, že obsahuje rámec, ktorý má RST flag nastavený na 1. To zistí tak, že hodnotu flagu vydelí 8 a ak je zvyšok väčší alebo rovný 4, tak obsahuje RES flag. Druhý spôsob je, že obe strany pošlú FIN flag. To je keď hodnota flagu je nepárna. V triede TCP_communication si program zapamätá, z ktorej strany prišiel FIN flag, a ak príde z oboch strán, tak sa komunikácia označí ako kompletná. Ak si užívateľ zvolí, že chce vypísať niektorú z šiestich TCP komunikácií, tak program z nej vypíše jednu kompletnú a jednu nekompletnú komunikáciu. Ak má niektorá komunikácia viac ako 20 rámcov, tak sa vypíše iba jej prvých a posledných 10 rámcov.

Úloha 4g

Program hľadá vnorený protokol v IPv4 a ak to je UDP, tak vypíše nájdené porty. Ak sa jeden z portov v desiatkovej sústave rovná 69, tak sa jedná o začiatok tftp komunikácie. Pomocou triedy TFTP sa zapamätá tento rámec a začne sa ním nová tftp komunikácia, ktorú si program pamätá pomocou triedy TFTP_communication a pridá ju do poľa TFTP_communications_list. Všetky ostatné UDP rámce, ktoré nemajú port s hodnotou 69 majú porovnávané MAC adresy, IP adresy a porty s už začatými TFTP komunikáciami. Pri druhom rámci komunikácie sa zistí, na aký port sa zmenil ten, ktorý mal hodnotu 69, a bude sa porovnávať aj ten. Ak si užívateľ zvolí, že chce vypísať tftp komunikácie, tak ak má niektorá komunikácia viac ako 20 rámcov, tak sa vypíše iba jej prvých a posledných 10 rámcov. Pred každou komunikáciou sa vypíše jej poradie.

Úloha 4h

Ak program pri nájdení IPv4 protokolu nájde v ňom vnorený ICMP protokol, tak v ňom zistí jeho typ pomocou externého textového súboru ICMP_Types.txt a zapamätá si ho pomocou triedy ICMP. Podľa IP adries ho priradí do správnej komunikácie. ICMP komunikácia je vytvorená pomocou triedy ICMP_communication a delí rámce do troch polí podľa typu, Echo, Echo reply a ostatné typy. Ak užívateľ zvolí výpis ICMP komunikácie, tak sa najprv z komunikácie vypíšu popárované Echo s Echo reply, a potom všetky ostatné rámce danej komunikácie.

Úloha 4i

Ak program zistí, že EtherType II obsahuje ARP protokol, tak podľa políčka Operation zistí, či sa jedná o ARP-Request alebo ARP-Reply, a adresy uvedené v ARP hlavičke. Podľa adries program hľadá správnu komunikáciu. Ak žiadnu vyhovujúcu nenájde, tak vytvorí novú. Komunikáciu si program pamätá pomocou triedy ARP_communication v poli ARP_communications_list. V jednej komunikácii rozdeľuje

rámce na ARP-Request a ARP-Reply a ak má z oboch aspoň jeden rámec, tak sa označí ako kompletná. Ak si užívateľ zvolí výpis ARP komunikácií, tak sa najprv vypíšu kompletné komunikácie a potom tie ku ktorým sa nenašlo ARP-Request alebo ARP-Reply.

Príklad štruktúry externých súborov pre určenie protokolov a portov

Všetky externé textové súbory obsahujú desiatkovú hodnotu čísla a medzerou oddelený názov. Každé číslo začína na novom riadku. Takto napríklad vyzerá súbor EtherType.txt.

```
512 XEROX PUP
513 PUP Addr Trans
2048 IPv4
2049 X.75 Internet
2053 X.25 Level 3
2054 ARP
32821 Reverse ARP
32923 Appletalk
33011 Apple Talk AARP
33024 IEEE 802.1Q VLAN-tagged frames
33079 Novell IPX
34525 IPv6
34827 PPP
34887 MPLS
34888 MPLS with upstream-assigned label
34915 PPPoE Discovery Stage
34916 PPPoE Session Stage
35020 Link Layer Discovery Protocol LLDP
36864 Loopback
```

Opísané používateľské rozhranie

Výpis úloh 1 až 3 prebieha automaticky po spustení programu výpisom do externého textového súboru a nie je potrebná interakcia s používateľom.

Príklad výpisu úloh 1-3 do textového súboru PKS1-output.txt:

```
Ramec 1
Dlzka ramca poskytnuta pcap API: 60 B
Dlzka ramca prenasaneho po mediu: 64 B
Zdrojova MAC adresa: 00 16 47 02 24 1a
Cielova MAC adresa: 01 80 c2 00 00 00
IEEE 802.3 - LLC
BPDU
01 80 c2 00 00 00 00 16  47 02 24 1a 00 26 42 42
03 00 00 00 00 00 80 01  00 16 47 02 24 00 00 00
00 00 80 01 00 16 47 02  24 00 80 1a 00 00 14 00
02 00 0f 00 00 00 00 00  00 00 00 00

Ramec 2
```

```
Dlzka ramca poskytnuta pcap API: 73 B
Dlzka ramca prenasaneho po mediu: 77 B
Zdrojova MAC adresa: 00 16 e6 54 c8 97
Cielova MAC adresa: 08 00 27 f6 18 f9
Ethernet II
IPv4
Zdrojova IP adresa: 172.26.0.4
Cielova IP adresa: 172.26.0.20
UDP
tftp
Zdrojovy port: 42301
Cielovy port: 69
08 00 27 f6 18 f9 00 16  e6 54 c8 97 08 00 45 00
00 3b 00 00 40 00 40 11  e2 65 ac 1a 00 04 ac 1a
00 14 a5 3d 00 45 00 27  58 85 00 01 6b 65 69 72
61 2d 6b 6e 69 67 68 74  6c 65 79 2e 6a 70 67 00
6e 65 74 61 73 63 69 69  00
```

Výpis komunikácií z úlohy 4 prebieha tak, že sa po spustení programu do konzoly vypíše menu a užívateľ si môže zvoliť, ktorú komunikáciu chce vypísať.

```
Nacitany subor s protokolmi: C:\Users\david\Desktop\vzorky_pcap_na_analyzu\eth-9.pcap
Vystupny subor:  PKS1-output.txt

Vypis HTTP komunikacie: 1
Vypis HTTPS komunikacie: 2
Vypis TELNET komunikacie: 3
Vypis SSH komunikacie: 4
Vypis FTP-riadiace komunikacie: 5
Vypis FTP-datove komunikacie: 6
Vypis TFTP komunikacie: 7
Vypis ICMP komunikacie: 8
Vypis ARP komunikacie: 9
Koniec: 0
Zadajte akciu:
```

Po zadaní čísla od 1 do 9 sa do konzoly vypíšu nájdené komunikácie zvoleného protokolu. Ak sa nenájde žiadna hľadaná komunikácia, tak sa menu vypíše znovu. Ak užívateľ zadá niečo iné ako čísla od 1 do 9, tak sa program ukončí.

Voľba implementačného prostredia

Môj program som naprogramoval v programovacom jazyku Python 3.7 v IDE PyCharm od JetBrains. Programoval som v Pythone, pretože sa v ňom pracuje jednoduchšie s dátami ako v jazykoch C a C++. Využil som v ňom knižnicu scapy na načítanie rámcov z .pcap súboru. Prostredie PyCharm som si zvolil najmä vďaka predošlým skúsenostiam s prostrediami od firmy JetBrains a celkovo vysokej prehľadnosti a jednoduchosti používania.