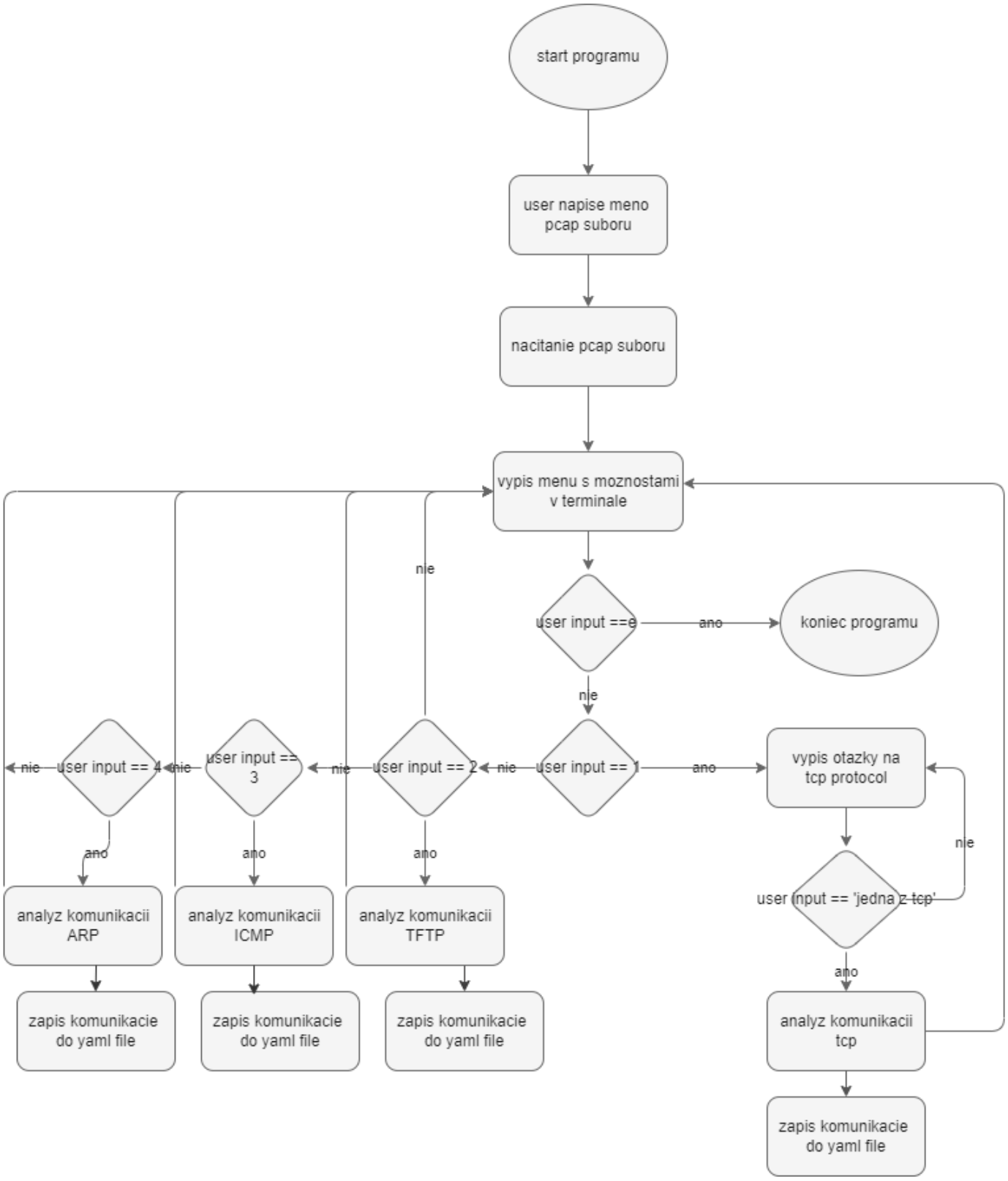
Analyzator sieťovej komunikácie

ZADANIE 1

SKRYPOVA KAROLINA ID:123423

Diagram spracovávania a fungovania riešenia:



Navrhnutý mechanizmus analyzovania protokolov na jednotlivých vrstvách:

Ulohy 1-3:

Po spustení programu užívateľ napíše názov súboru, ktorý chce analyzovať, funkcia read_pcap_to_list(filename) načíta dáta zo súboru pcap, prevedie ich na bajty a následne do hex formátu. Po ktorej sa zavolá funkcia yaml_packets(packets), ktorá ich spracuje a zapíše do Yaml a tiež vráti slovník so spracovanou komunikáciou na následné spracovanie komunikácie medzi protokolmi.

Vo funkcii yaml_packets spracujeme postupne každý ramec, nájdeme dĺžku a dĺžku podľa média, ako aj src mac a cieľový mac. Ďalej určíme typ rámca, ak je 13 a 14 byte viac ako 1536, potom to bude Ethernet II a ak nie, potom existujú 3 možné možnosti pre IEEE 802.3, ktoré budú tiež definované v doplnkovej funkcii.

Potom nájdeme aj IP adresy, interné protokoly a všetky informácie, ktoré boli požadované v druhej úlohe. Na koniec súboru si zapíšeme zoznam IP adries a tiež IP adresu, ktorá posielala najväčší počet paketov.

Uloha 4:

Pre štvrtú úlohu som vytvorila menu, v ktorom môže užívateľ zadať číslami, aký druh komunikácie chce analyzovať.

Analýza TCP:

Najprv používateľ dodatočne zadá interný protokol TCP, ktorý chce analyzovať. Pomocou funkcie find_flags si najskôr roztriedime zoznam našich balíčkov len na tie, ktoré hľadáme a ku každému baleniu pridáme v dictionary aj ďalšie údaje, ako sú flagy, ktoré nám pomôžu určiť správne otváranie a zatváranie komunikácie, aby sme mohli určiť, či je komunikácia kompletna alebo nie.

K dispozícii sú 2 možnosti otvárania a 3 možnosti zatvárania, ktoré zaškrtneme vo funkciách: check_open_communication a check_close_communication. Správne otvorenie má nasledujúce možnosti:

1. syn, syn ack, ack
2. syn, syn, ack, ack

Správne zatvaranie má nasledujúce možnosti:

1. fin ack, ack, fin ack, ack
2. fin ack, fin ack, ack
3. zatvaranie pomocou rst

Rozdelíme naše protokoly aj na komunikácie, aby som ich oddelila, použila som dictionary, kde som z každého balíka zobrala kľúč, ktorý pozostával z dvoch IP a som skontrolovala, či je v slovníku alebo nie, ak už existuje, potom som jednoducho pridala balík už k existujúcemu kluču. Potom som vytvorila list, kam som vrátila hodnoty zo slovníka. Vo finálnej funkcii pre tento protokol som implementovala záznam v súbore yaml, ktorý má podobnú štruktúru ako hlavna funkcia, ktora zapisuje všetky pakety z pkap do yaml na začatku.

Analyza TFTP:

Na analýzu tohto protokolu sú pre nás dôležité IP a porty, najskôr nájdeme všetky protokoly UDP, pretože zvyčajne sme označovali protokoly TFTP iba vtedy, ak majú port 69, ale pokračujú po tomto porte, a keď pošleme na port 69, je to začiatok komunikácie. Pomocou IP a portu vytvoríme list párov protokolov a potom vrátime hotový list s pármi na ďalšie spracovanie.

Kontrolujeme tiež kompletnost nášho protokolu; ak je opcode chyba, tak je to koniec komunikácie, alebo ak pošleme súbor s veľkosťou dát menšou ako 512 bajtov a po ňom sa odošle ACK, ktoré upozorní na koniec, tak v týchto prípadoch protokol bude kompletný. Vo všetkých ostatných prípadoch bude protokol nekompletny.

Analyza ICMP:

Najprv si zoraďme všetko ICMP protokoly a zapíšem ich do nového list, ktorý použijeme v ďalších funkciách. Najprv zistíme typ nášho ICMP frame, všetky typy sa zapíšu do súboru a načítajú sa z neho, potom ak je typ nášho balíka replay alebo reques, pridáme aj ich ID a sériové číslo. Komunikáciu budeme triediť podľa balíkov IP a ID. Po roztriedení komunikácií určíme, ktoré z nich sú komletne a ktoré nie. Ak je typ paketu odlišný od request alebo reply, potom automaticky prejde do nekompletnej komunikácie a tiež ak je veľkosť komunikácie iba 1, potom to tiež nie je kompletna komunikácia.

Potom zapíšeme dva listy do súboru Yaml pomocou podobného princípu, aký bol implementovaný skôr.

Analyza ARP:

Najprv som roztriedila protokoly arp do samostatného listu a pridala som k nim aj apr_opcode. Potom som vytvoril dva hárky pre request a reply a roztriedila arp packets. Porovnaním IP adries každého listu s druhým som vytvorila páry a bez páru zostala len nekompletne komunikácie.

Príklad štruktúry externých súborov pre určenie protokolov a portov:

ETHERNET:

```
0200 XEROX PUP
0201 PUP Addr Trans
0800 IPv4
0801 X.75 Internet
0805 X.25 Level 3
0806 ARP
8035 Reverse ARP
809b Appletalk
80f3 AppleTalk AARP
8100 IEEE 802.1Q VLAN-tagged frames
8137 Novell IPX
86dd IPv6
880b PPP
8847 MPLS
8848 MPLS with upstream-assigned label
8863 PPPoE Discovery Stage
8864 PPPoE Session Stage
88cc Link Layer Discovery Protocol (LLDP)
9000 Loopback
```

ICMP:

```
0  Echo Reply
3  Destination Unreachable
4  Source Quench
5  Redirect
8  Echo Request
9  Router Advertisement
10 Router Solicitation
11 Time Exceeded
12 Parameter Problem
13 Timestamp
14 Timestamp Reply
15 Information Request
16 Information Reply
17 Address Mask Request
18 Address Mask Reply
30 Traceroute
```

Ipv4

```
1   ICMP
2   IGMP
6   TCP
9   IGRP
17  UDP
47  GRE
50  ESP
51  AH
57  SKIP
88  EIGPR
115 L2TP
```

PORTS

```
0007 echo
0019 chargen
0020 ftp_data
0021 ftp_control
0022 ssh
0023 telnet
0025 smtp
0053 domain
```

```
0069 tftp
0079 finger
0080 http
0110 pop3
0111 sunrpc
0119 nntp
0139 netbios-ssn
0146 imap
0179 dgp
0389 ldap
0443 https
0445 microsoft-ds
1080 socks
```

TCP:

```
20 FTP-DATA
21 FTP-CONTROL
22 SSH
23 TELNET
25 SMTP
53 DNS
80 HTTP
110 POP3
119 NNTP
137 NETBIOS-NS
139 NETBIOS-SSN
143 IMAP
162 SNMP-TRAP
179 BGP
389 LDAP
443 HTTPS
514 SYSLOG
17500 DB-LSP-DISC
```

UDP:

```
37 TIME
53 DNS
67 DHCP
68 DHCP
69 TFTP
80 HTTP
137 NETBIOS-NS
138 NETBIOS-DGM
161 SNMP
162 SNMP-TRAP
443 HTTPS
514 SYSLOG
520 RIP
1900 SSDP
5353 MDNS
17500 DB-LSP-DISC
33434 TRACEROUTE
```

Používateľské rozhranie:

Na začiatku dostane používateľ možnosť napísať názov súboru, ktorý chce spracovať, potom bude súbor spracovaný, zaznamenaný v Yaml a používateľ dostane na výber pri analýze nasledujúceho komunikácie.

ak stlačí 1, potom program analyzuje komunikáciu TSP, ktorej názov používateľ zadá, a zapíše ju do Yaml, ak 2, potom analyzuje protokoly TFT a zapíše ju do Yaml, ak 3, potom analyzujte komunikáciu ISP a zapíšte ju do Yaml, ak 4, potom arp a ak používateľ zadá e, program sa ukončí.

Implementačné prostredie:

Uloha bola implementovaná v Pythone pomocou IDE PyCharm.

Použité knižnice:

Scapy - na čítanie súboru pcap

Ruamel.yaml - pre prácu so súborom Yaml

Zhodnotenie a prípadné možnosti rozšírenia:

V tomto projekte boli úspešne splnené podmienky úlohy, môj projekt úspešne analyzuje protokoly, získava o nich informácie z bajtov a zapisuje informaciu o nich do Yaml súboru. Možnosti rozšírenia môžu zahŕňať schopnosť analyzovať väčší počet rôznych protokolov, grafické rozhranie, ako aj metódy filtrovania podobné wireshark.