21/10/2021

Analyzátor sieťovej komunikácie

Počítačové a komunikačné siete – zadanie 1

Samuel Hetteš, ID: 110968
STU FIIT 2020/2021

OBSAH

OBSAH 1
ZADANIE 2
BLOKOVÝ NÁVRH RIEŠENIA 5
MECHANIZMUS 6
- Analýza jedného rámca 6
- Prvotná analýza 10
- Analýza podľa požiadavky užívateľa 11

ŠTRUKTÚRA EXTERNÝCH SÚBOROV 12
OPIS POUŽÍVATEĽSKÉHO ROZHRANIA 13
VOĽBA IMPLEMENTAČNÉHO PROSTREDIA 14
ZOZNAM FUNKCIÍ 14
ZÁVER 15

ZADANIE

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).

IP adresy a počet odoslaných / prijatých paketov sa musia zhodovať s IP adresami vo výpise Wireshark -> Statistics -> IPv4 Statistics -> Source and Destination Addresses.

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 ARPReply 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.

6) V procese analýzy rámcov pri identifikovaní jednotlivých polí rámca ako aj polí hlavičiek vnorených protokolov nie je povolené použiť funkcie poskytované použitým programovacím jazykom alebo knižnicou. **Celý rámec je potrebné spracovať postupne po bajtoch.**

7) Program musí byť organizovaný tak, aby bolo možné jednoducho rozširovať jeho funkčnosť výpisu rámcov pri doimplementovaní jednoduchej funkčnosti na cvičení.

8) Študent musí byť schopný preložiť a spustiť program v miestnosti, v ktorej má cvičenia. V prípade dištančnej výučby musí byť študent schopný prezentovať podľa pokynov cvičiaceho program online, napr. cez Webex, Meet, etc.

V danom týždni, podľa harmonogramu cvičení, musí študent priamo na cvičení doimplementovať do funkčného programu (podľa vyššie uvedených požiadaviek) ďalšiu prídavnú funkčnosť.

Program musí mať nasledovné vlastnosti (minimálne):

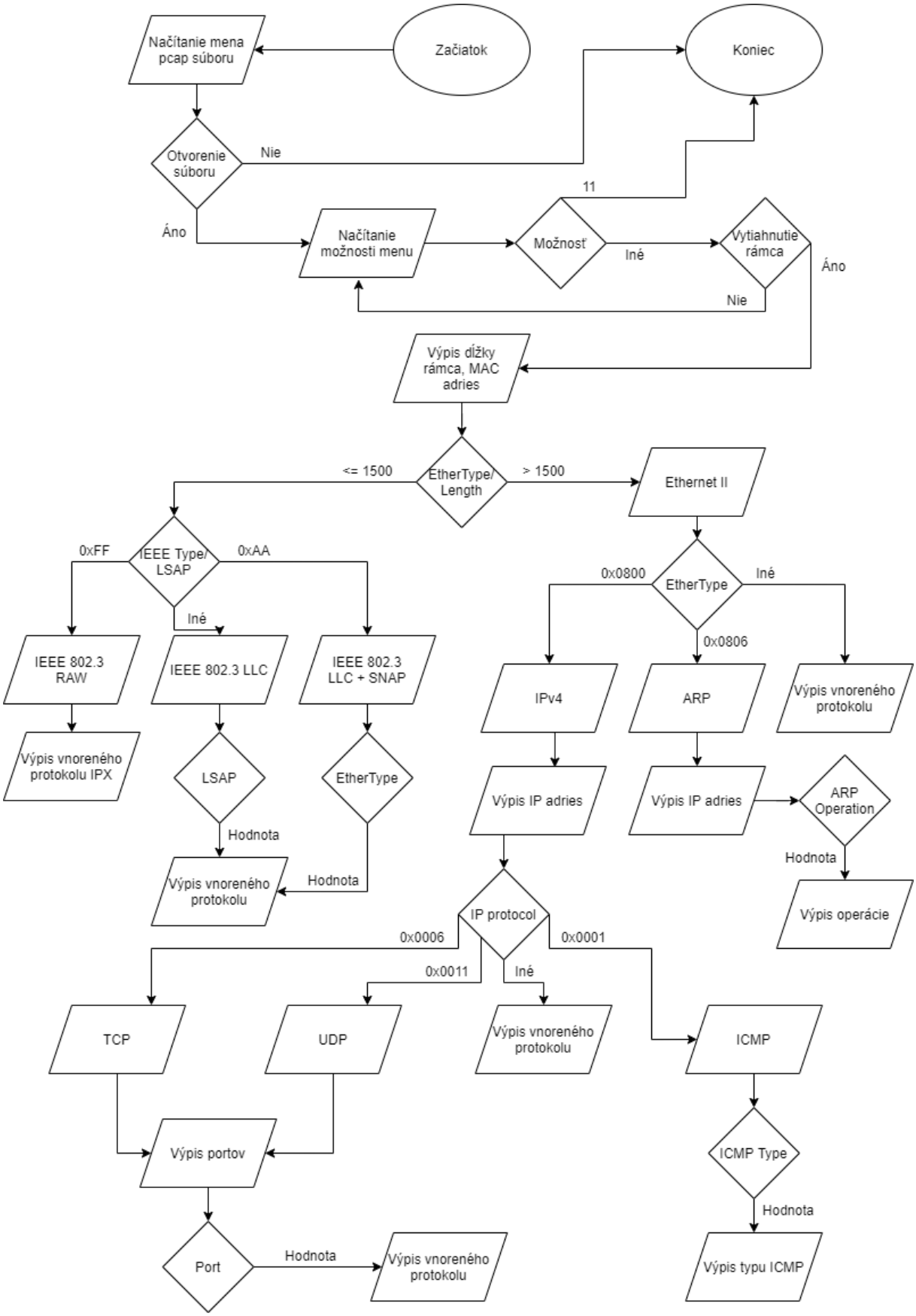
1) Program musí byť implementovaný v jazykoch C/C++ alebo Python s využitím knižnice pcap, skompilovateľný a spustiteľný v učebniach. Na otvorenie pcap súborov použite knižnice libpcap pre linux/BSD a winpcap/ npcap pre Windows. Použité knižnice a funkcie musia byť schválené cvičiacim. V programe môžu byť použité údaje o dĺžke rámca zo struct pcap_pkthdr a funkcie na prácu s pcap súborom a načítanie rámcov:
   pcap_createsrcstr()
   pcap_open()
   pcap_open_offline()
   pcap_close()
   pcap_next_ex()
   pcap_loop()
   Použitie funkcionality libpcap na priamy výpis konkrétnych polí rámca (napr. ih->saddr) bude mať za následok nulové hodnotenie celého zadania.
2) Program musí pracovať s dátami optimálne (napr. neukladať MAC adresy do 6x int).
3) Poradové číslo rámca vo výpise programu musí byť zhodné s číslom rámca v analyzovanom súbore.
4) Pri finálnom odovzdaní, pre každý rámec vo všetkých výpisoch uviesť použitý protokol na 2. 4. vrstve OSI modelu. (ak existuje)
5) Pri finálnom odovzdaní, pre každý rámec vo všetkých výpisoch uviesť zdrojovú a cieľovú adresu / port na 2. - 4. vrstve OSI modelu. (ak existuje)

Nesplnenie ktoréhokoľvek bodu minimálnych požiadaviek znamená neakceptovanie riešenia cvičiacim.

Súčasťou riešenia je aj dokumentácia, ktorá musí obsahovať najmä:

a) zadanie úlohy,
b) blokový návrh (koncepcia) fungovania riešenia,
c) navrhnutý mechanizmus analyzovania protokolov na jednotlivých vrstvách,
d) príklad štruktúry externých súborov pre určenie protokolov a portov,
e) opísané používateľské rozhranie,
f) voľbu implementačného prostredia.

BLOKOVÝ NÁVRH RIEŠENIA



MECHANIZMUS

Hlavná časť mechanizmu riešenia je obsiahnutá vo funkcii main, v ktorej možno ďalej tento mechanizmus rozčleniť na viacero častí na základe požiadavky užívateľa. Jednotlivé časti fungujú na podobnom princípe a v každej sa volá funkcia pre analýzu jedného rámca. Ďalej pre hlbšie pochopenie jednotlivých častí najskôr vysvetlím spôsob analýzy jedného rámca.

Analýza jedného rámca

Analýza jedného rámca prebieha vo funkcii *process_frame()*. Táto funkcia sa dá spustiť s argumentom *print*, kedy funkcia volá ďalšie funkcie pre výpis informácií. Dôležité je spomenúť, že pre výpis vnoreného protokolu alebo typu pri ICMP slúži funkcia *print_protocol()*, ktorá ako vstupné argumenty dostane porovnávanú hodnotu, súbor v ktorom má hodnotu porovnávať a výstupný súbor. Táto funkcia sa využíva pri všetkých výpisoch, kde treba hodnotu hľadať v externom súbore. Spustenie funkcie *process_frame()* bez argumentu print je obzvlášť dôležité pri analýze komunikácií, pretože na základe jej návratovej hodnoty sa určtuje typ rámca (napr. návratová hodnota 'A' – ARP, 'I' – ICMP a tak ďalej). Fungovanie tejto funkcie načrtnem v bodoch. Pozície Bytov rámca, ktoré budú uvádzané v týchto bodoch sú rovnaké ako v programe Wireshark, počnúc pozíciou 0.

1) Analýza dĺžky rámca

Zavolá sa funkcia *print_len()*, ktorá dostane ako argument dĺžku rámca poskytnutú pcap API a výstupný súbor. Funkcia následne vyhodnotí dĺžku rámca po médiu – ak je API dĺžka menšia ako 60, tak dĺžka po médiu je 64. Inak je dĺžka prenášaná po médiu API dĺžka + 4.

Príklad funkcie *print_len()*:

```
/* Printing length */
void print_len(int len, FILE *output){
    fprintf(output, "| API length: %d\n", len); //print API length
    if(len < 60) //if API length is < 60, the wire length will always be 64
        fprintf(output, "| Wire length: %d\n", 64);
    else //else the wire length is going to be the API length + 4
        fprintf(output, "| Wire length: %d\n", len + 4);
}
```

2) Analýza typu rámca

Táto analýza prebieha vo funkcii *choose_ethernet_standard()*. Pre analýzu typu rámca a výpis potrebujem poznať štruktúru Ethernet hlavičky, ktorá vyzerá nasledovne:

Vyhodnotenie prebieha spôsobom ako bolo naznačené v blokovej schéme riešenia vyššie. Na základe typu rámca sa rozhodne ako bude mechanizmus ďalej pokračovať. Ak je typ rámca:

a. IEEE 802.3 RAW – vypíše sa vnorený protokol IPX
b. IEEE 802.3 LLC – v tomto prípade sa pozerám na prvý byte, ktorý nasleduje za ethernet hlavičkou, táto hodnota sa porovná s hodnotami v externom súbore *llc_saps.txt* a zistí sa vnorený protokol.
c. IEEE 802.3 LLC+SNAP – porovná sa hodnota *EtherType*, ktorá sa nachádza v SNAP hlavičke (20-21 B) s hodnotami v externom súbore *eth_types.txt* a vypíše sa vnorený protokol

   Štruktúra SNAP hlavičky:

d. Ethernet II – porovná sa hodnota *EtherType* (11-12 B) s hodnotami v externom súbore *eth_types.txt* a vypíše sa vnorený protokol

Príklad funkcie *choose_eth_standard():*

```
/* Printing Ethernet standard */
void choose_eth_standard(unsigned short eth_type, unsigned char ieee_type, FILE *output){
    //checking eth_type/length first
    if(eth_type > 0x05DC) //ETHERNET II
        fprintf(output, "|-- Ethernet II\n");
    //IEEE 802.3 - checking ieee_type (LSAP)
    else if(ieee_type == 0xFF) //RAW
        fprintf(output, "|-- Novel 802.3 RAW\n");
    else if(ieee_type == 0xAA) //LLC + SNAP
        fprintf(output, "|-- IEEE 802.3 LLC + SNAP\n");
    else //LLC
        fprintf(output, "|-- IEEE 802.3 LLC\n");
}
```

3) Výpis MAC adries

Tento výpis zabezpečuje funkcia *print_mac_addresses()*, ktorá ako argumenty dostane *cieľovú MAC adresu* (0-5 B), *zdrojovú MAC adresu* (6-11 B) a výstupný súbor. Touto časťou sa okrem záverečného výpisu paketu analýza pre typy rámca IEEE 802.3 končí. Analýza bude naopak pokračovať pre typ rámca Ethernet II a vnorené protokoly IPv4 a ARP.

4) Analýza vnorených protokolov IPv4 a ARP

➢ Analýza IPv4

Štruktúra IP hlavičky:

a. *Výpis IP adries*
Výpis zabezpečuje funkcia *print_ip_addresses()*, ktorej vstupnými argumentami sú *zdrojová IP adresa* (26-29 B), *cieľová IP adresa* (30-33 B) a výstupný súbor.

b. *Výpis vnoreného protokolu*
Porovná sa hodnota *Protocol* (23 B) s hodnotami v externom súboru *ip_protocols.txt* a vypíše sa vnorený protokol.

c. *Analýza vnorených protokolov TCP, UDP a ICMP*
Pre zistenie, kde začínajú hlavičky jednotlivých protokolov musím poznať dĺžku IP hlavičky. Pozriem sa na hodnotu *IHL* (14 B), ktorej prvé 4 bity hovoria o verzii a ďalšie 4 o dĺžke IP hlavičky. Toto rozdelenie a vytiahnutie dĺžky zabezpečí funkcia *get_ihl()*.

- ***Analýza TCP***

  Štruktúra TCP hlavičky (za ňou môžu byť ešte options):

  - Výpis *zdrojového portu* (0-1 B TCP hlavičky) a *cieľového portu* (2-3 B TCP hlavičky)
  - Výpis vnoreného protokolu – vyhodnotí sa menší z portov a ten sa následne porovná s hodnotami v externom súbore *tcp_ports.txt*
  - Výpis flagov, ktoré daný TCP rámec má nastavené.

- ***Analýza UDP***

  Štruktúra UDP hlavičky:

- Výpis *zdrojového portu* (0-1 B UDP hlavičky) a *cieľového portu* (2-3 B UDP hlavičky)
- Výpis vnoreného protokolu – vyhodnotí sa menší z portov a ten sa následne porovná s hodnotami v externom súbore *udp_ports.txt*
- V prípade, že sa nenašiel vnorený protokol, funkcia kontroluje či nedostala ako vstupný argument bool TFTP, čo značí, že daný rámec je typu TFTP.

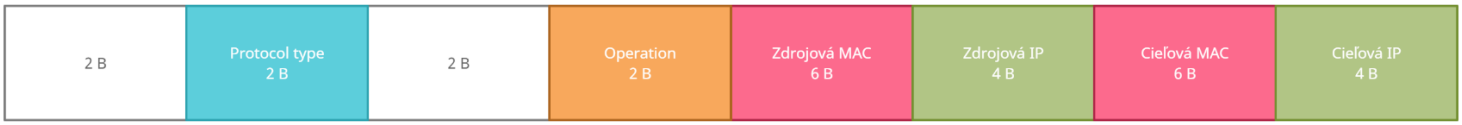
- ***Analýza ICMP***

  Štruktúra ICMP hlavičky:

  - Výpis typu ICM – zistí sa hodnota *Type* (0 B ICMP hlavičky) a porovná sa s hodnotami v externom súbore *icmp_types.txt*

➢ Analýza ARP

Štruktúra ARP hlavičky:



a. Výpis vnoreného protokolu

   Porovná sa hodnota *Protocol Address Type* (16-17 B) s hodnotami v externom súboru *eth_types.txt* a vypíše sa vnorený protokol.

b. *Výpis IP adries*

   Vypíše sa *zdrojová IP adresa* (28-31 B) a *cieľová IP adresa* (38-41 B).

c. *Výpis ARP operácie + typu ARP*

   Porovná sa hodnotá *Operation* (20-21 B), ak je táto hodnota 1 ide o operáciu ARP Request – tu sa zisťuje, či zdrojová adresa nie je 0.0.0.0, v tom prípade ide o ARP Probe. Ďalej sa zisťuje či zdrojová a cieľová adresa nie sú rovnaké, v tom prípade ide o ARP Announcement. Nakoniec sa vypíše *cieľová IP adresa* (38-41 B), ku ktorej sa *hľadá MAC adresa*. Ak je táto hodnota 2, ide o operáciu *ARP Reply* – tu sa zisťuje či zdrojová a cieľová adresa opäť nie je rovnaká, v tom prípade ide o Gratuitous ARP. Na záver sa opäť vypíše *IP adresa odosielateľa* (28-31 B) a jeho *nájdená MAC adresa* (22-27 B).

5) Výpis jednotlivých bytov paketu

Výpis zabezpečuje *funkcia print_packet()*, ktorej vstupnými argumentami sú pointer na začiatok paketu, dĺžka paketu poskytnutá pcap API a výstupný súbor.

Teraz už poznáte základ fungovania jednotlivých častí. Ďalej opíšem čo sa v týchto častiach deje.

Prvotná analýza

Táto časť mechanizmu spočíva v tom, že predtým než sa niečo začne vypisovať, prebehne analýza celého PCAP súboru, to znamená, že pre každý rámec sa zavolá funkcia *process_frame()* a na základe jej návratovej hodnoty sa zozbierajú dáta, ktoré sú následne využívané pri výpise. Napríklad pri požiadavke pre výpis HTTP komunikácií sa zanalyzujú všetky HTTP komunikácie. Pre ukladanie dát mám pre každý typ komunikácie vytvorené vlastné štruktúry a funkcie *insert*, ktoré zaraďujú nový rámec do zoznamu existujúcich komunikácií.

- TRIEDENIE TCP KOMUNIKÁCIÍ

Rozlišujem tieto typy komunikácií: HTTP, HTTPS, TELNET, SSH, FTP-CONTROL, FTP-DATA. Pre každú komunikáciu uchovávam informáciu o zdrojovej a cieľovej IP adrese a komunikujúcom porte na základe čoho viem priradiť daný rámec ku komunikácii. Ďalej uchovávam údaje o počte rámcov, stave (úplná, neúplná), splnených častiach otvorenia alebo ukočenia komunikácie a čísla rámcov, ktoré do komunikácie patria.

Stavy komunikácie:

a) -1: tento stav značí, že 3-way handshake otvorenia komunikácie je nasledovný: SYN alebo SYN, SYN-ACK, teda nie je ešte dokončený úplne
b) 0: tento stav značí, že komunikácia je otvorená (neúplná), avšak ešte nebola ukončená a 3-way handshake prebehol v poriadku: SYN, SYN-ACK, ACK
c) 1: tento stav značí, že komunikácia bola otvorená a ukončená RST rámcom, za ktorým však ešte môže nasledovať ACK.
d) 2: tento stav značí, že komunikácia bola taktiež otvorená a ukončená, avšak už nemôže prísť žiadny iný rámec ako napríklad v stave 1. Ukončenia mohli byť nasledovné: FIN-ACK, ACK, FIN-ACK, ACK alebo RST, ACK.

- TRIEDENIE TFTP KOMUNIKÁCIÍ

Pre tieto komunikácie uchovávam zdrojovú a cieľovú IP adresu, zdrojový port prvého rámca, zmenený cieľový port, počet rámcov a čísla

rámcov. Začiatočný rámec komunikácie zistím tak, že komunikácia začne na cieľovom porte 69. Tento port bude nasledovne zmenený pri druhom rámci komunikácie, kedy ho uchovám. Ak následne nájdem rámec, ktorý má rovnaké IP adresy a rovnaké porty, viem, že patrí do TFTP komunikácie.

- TRIEDENIE ICMP KOMUNIKÁCIÍ

  Tieto komunikácie rozlišujem iba na základe zdrojovej a cieľovej IP adresy. Ako pri každej uchovávam taktiež počet rámcov a čísla rámcov.

- TRIEDENIE ARP KOMUNIKÁCIÍ

  Pri ARP komunikáciach rozlišujem viacero typov: komunikácie, ktoré obsahovali Request aj Reply, komunikácie, ktoré mali iba Request alebo Reply a samostatnú kategóriu tvoria ARP Probe, Announcement a Gratuitous ARP. Pre zaradenie rámca do komunikácie si uchovávam informáciu o zdrojovej a cieľovej IP adrese, zdrojovej MAC adrese a stave komunikácie:

  a) stav -1: iba Reply, do tejto komunikácie už nemožno zaradiť žiadny iný rámec
  b) stav 0: iba Request, do tejto komunikácie možno zaradiť ďalšie Requesty alebo Reply
  c) stav 1: Request aj Reply – ukončená komunikácia
  d) stav 5: ARP Probe, stav 6: ARP Announcement, stav 7: Gratuitous ARP – tieto stavy tvoria samostatnú kategóriu, ak je napr. viacero ARP Probe/Announcement s rovnakou IP adresou, tak ich zoskupujem, Gratuitous ARP vypisujem každé samostatne – riadil som sa zoskupovaním requestov a výpisom reply samosatne

  Príklad štruktúr pre ukladanie dát o ARP komunikácii/ach:

```c
typedef struct arp_communication{
    unsigned char source_ip[IP_ADD_SIZE]; //source
    unsigned char target_ip[IP_ADD_SIZE]; //target
    unsigned char mac_address[MAC_ADD_SIZE]; //sour
    int state; //state of communication: 0 = request
    int frames; //number of communication frames
    int *indexes; //frame numbers
} ARP_COMM;
```

```c
typedef struct arp_data{
    ARP_COMM **arp_comm;
    int arp_counter;
} ARP_DATA;
```

Analýza podľa požiadavky užívateľa

Začne sa prvotná analýza, zozbierajú sa dáta pre konkrétnu požiadavku a následne sa začne výpis. Pri výpise sa porovnávajú čísla rámcov zo získaných dát.

- ANALÝZA VŠETKÝCH RÁMCOV (BODY 1-3)

  Táto analýza sa spustí pri voľbe 1 od používateľa. Predtým než začne, sa zanalyzujú TFTP komunikácie, pretože tie nevieme jednoducho rozlíšiť na základe portu. Ak pri porovnávaní čísla rámcov TFTP komunikácií nastane zhoda, funkcia process_frame sa zavolá s nastaveným argumentom bool TFTP. Okrem iného pri tejto analýze prebieha aj zozbieranie štatistík o zdrojových adresách a nimi poslaných paketov, ktoré sa na záver vypíšu.

- ANALÝZA TCP KOMUNIKÁCIÍ (BOD 4, A - F)

  Táto analýza sa spustí pri voľbe 2-7 od užívateľa. Zúžitkujú sa dáta získané prvotnou analýzou TCP komunikácie a vytiahne sa iba prvá úlná a prvá neúplná komunikácia (ak existuje). Následne sa prechádza všetkými rámcami a porovnávaním čísiel rámcov sa zisťuje, ktoré rámce sa majú vypísať. Pri tejto analýze sa taktiež kontroluje, aby bolo vypísaných iba prvých a posledných 10 rámcov danej komunikácie. Jednotlivé voľby majú svoje vlastné výstupné súbory, ktoré budú opísané v časti štruktúra externých súborov.

- ANALÝZA TFTP A ICMP KOMUNIKÁCIÍ (BOD 4, G, F)

  Tieto typy analýz sa spustia pri voľbe 8 (ICMP) alebo 9 (TFTP) od užívateľa. Podobne ako pri TCP komunikáciach sa na základe zozbieraných dát porovnávajú čísla rámcov. Taktiež je zabezpečené, aby sa vypísalo iba prvých a posledných 10 rámcov komunikácie. V tomto prípade sa vypisujú všetky komunikácie.

- ANALÝZA ARP KOMUNIKÁCIÍ

  Táto analýza sa spúšta pri voľbe 10. Prebehne prvotná analýza a následne sa začnú vypisovať typy komunikácií v nasledovnom poradí: klasické Request a Reply, iba Request, iba Reply, ARP Probe, ARP Announcement a Gratuitous ARP. Rovnako sa vypisuje iba prvých a posledných 10 rámcov a vypisujú sa všetky komunikácie.

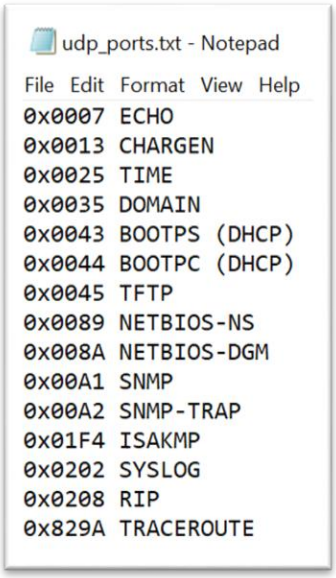
ŠTRUKTÚRA EXTERNÝCH SÚBOROV

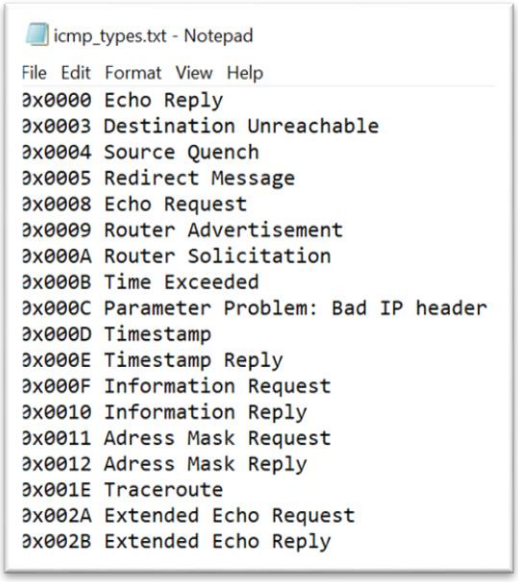
Pre lepšiu prehľadnosť som v samotnom priečinku programu vytvoril dva adresáre – **input a output**.

Input adresár obsahuje všetky súbory, z ktorých sa čítajú dáta – pcap súbory, eth_types.txt, icmp_types.txt, ip_protocols.txt, llc_saps.txt, tcp_ports.txt,

udp_ports.txt. Štruktúra textových súborov je rovnaká: v riadku sa vždy nachádza hexadecimálna hodnota a za ňou názov protokolu/typu.

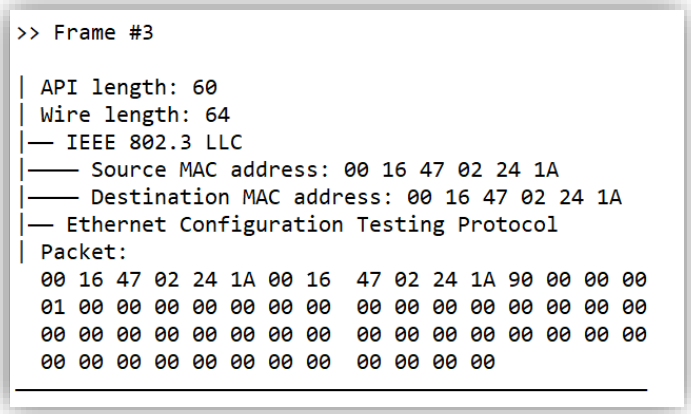
Príklady štruktúry vstupných súborov:





Output adresár obsahuje textové súbory do ktorých sa zapisuje výstup. Užívateľ má vždy možnosť do akého súbora chce daný výstup požiadavky zapísať.

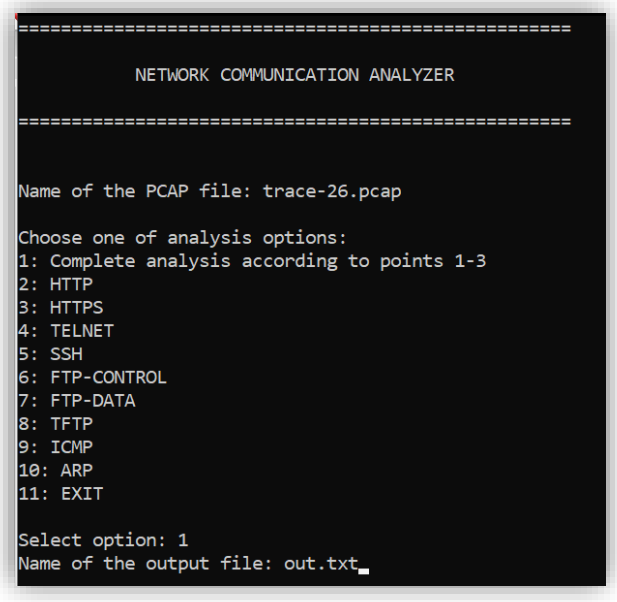
Príklad výstupu (výpisu rámca) z možnosti 1:



OPIS POUŽÍVATEĽSKÉHO ROZHRANIA

Pri prvom spustení sa zobrazí hlavička programu a následne sa od používateľa vypýta meno pcap súboru, ktorý si želá analyzovať. Následne mu bude poskytnutá menu, v ktorom si môže vybrať konkrétnu požiadavku. Po vybratí požiadavky si program vyžiada meno výstupného súboru. Po skončení požiadavky sa opäť zobrazí menu.

Príklad používateľského rozhrania:



VOĽBA IMPLEMENTAČNÉHO PROSTREDIA

Na implementáciu som si vybral programovací jazyk C. Túto možnosť som si zvolil preto, že som dlho nepracoval s inými jazykmi a C poskytuje dobrú manipuláciu so smerníkmi.

V programe som využil knižnice:

- stdio.h – základná knižnica
- stdlib.h – alokácia pamäte
- stdbool.h – práca s bool hodnotami
- libpcap – práca s pcap súbormi

ZOZNAM FUNKCIÍ

- structure_name *init_structure_name(arguments) – *funkcie pre inicializáciu dátových štruktúr*
- unsigned char extract_?_bit(unsigned char value*)* – *funkcie, ktoré vrátia hodnotu po vytiahnutí určitého bitu (pri TCP flagoch)*
- unsigned char get_ihl(unsigned char version_and_ihl*)* – *vráti dĺžku IP hlavičky v bytoch*
- unsigned short swap_bytes(unsigned short value) – *vymení byty a vráti hodnotu*
- void print_mac_adresses(unsigned char *source_address, unsigned char *dest_address, FILE *output) – *vypíše MAC adresy*
- void print_ip_adresses(unsigned char *source_address, unsigned char *dest_address, FILE *output) – *vypíše IP adresy*
- void print_packet(unsigned char *pkt_header, unsigned short pkt_len, FILE *output) – *vypíše celý paket*

- void choose_eth_standard(unsigned short eth_type, unsigned char ieee_type, FILE *output) – *vypíše ethernet štandard*
- void print_len(int len, FILE *output) – *vypíše dĺžku paketu*
- int print_protocol(FILE *file, unsigned short value, FILE *output) – *vypíše vnorený protokol*
- void print_ipv4_stats(IPV4_DATA *ipv4_data, FILE *output) – *vypíše štatistiky o IPV4 adresách*
- void print_tcp_flags(unsigned char flags, FILE *output) – *vypíše TCP flagy*
- pcap_t *open_pcap_file(char *file_name) – *otvorí PCAP súbor*
- void close_files(pcap_t *pcap_ptr, FILE *files[6], FILE *output) – *zatvorí súbory*
- int process_frame(const unsigned char *packet,struct pcap_pkthdr *pkt_hdr, int frame_counter, FILE *files[6], FILE *output, bool tftp, bool print) – *zanalyzuje rámec*
- TCP_DATA *tcp_insert(TCP_DATA *tcp_data, const unsigned char *packet, unsigned short port_id, int frame_counter, bool *error) – *vloží rámec do zoznamu TCP komunikácií*
- TFTP_DATA *tftp_insert(TFTP_DATA *tftp_data, const unsigned char *packet, int frame_counter) – *vloží rámec do zoznamu TFTP komunikácií*
- ICMP_DATA *icmp_insert(ICMP_DATA *icmp_data, const unsigned char *packet, int frame_counter) – *vloží rámec do zoznamu ICMP komunikácií*
- ARP_DATA *arp_insert(ARP_DATA *arp_data, const unsigned char *packet, int frame_counter) – *vloží rámec do zoznamu ARP komunikácií*
- IPV4_DATA *ipv4_insert(IPV4_DATA *ipv4_data, const unsigned char *packet) – *vloží IP adresu do zoznamu IP adries*

ZÁVER

Myslím, že môj program splnil požiadavky zadania. Dokáže pekne roztriediť jednotlivé komunikácie, poskytuje prehľadný výstup, v ktorom sa dá dobre orientovať. Taktiež poskytuje jednoduché rozhranie pre prácu užívateľa. Na záver by som sa už iba posťažoval na vybraté implementačné prostredie, práca so smerníkmi bola síce fajn, ale alokácia a uvoľnovanie pamäte mi dosť sťažila prácu.