Slovenská technická univerzita v Bratislave

Fakulta informatiky a informačných technológií

Tibor Vanek

Zadanie 1 –Analyzátor sieťovej komunikácie

Počítačové a komunikačné siete

Predmet: Počítačové a komunikačné siete
Čas cvičenia: Piatok 8:00
Cvičiaci: Ing. Matej Janeba 2021/2022

Obsah

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 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.

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).

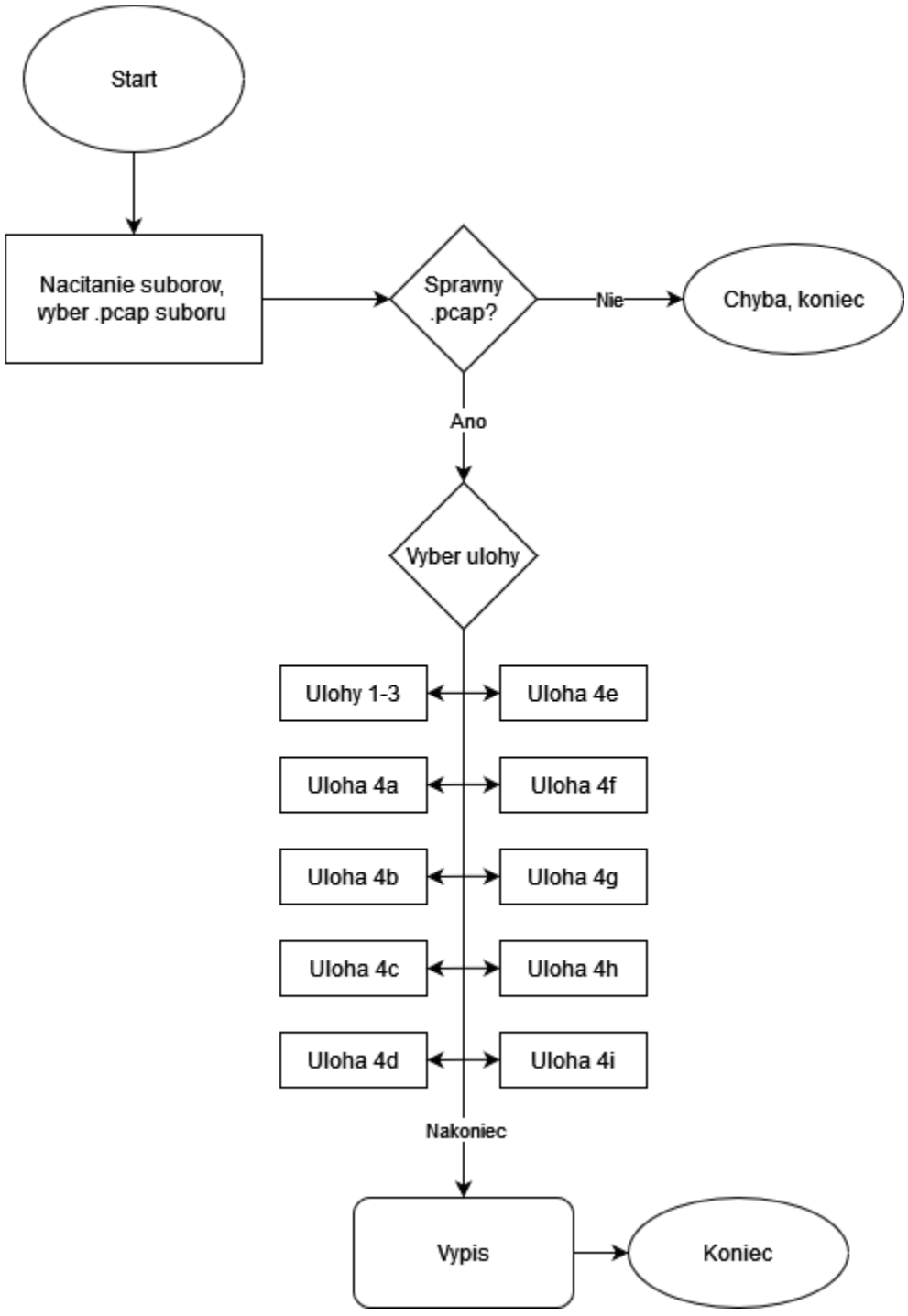
3) Poradové číslo rámca vo výpise programu musí byť zhodné s číslom rámca v analyzovanom súbore.

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.

Blokový návrh fungovania riešenia



Navrhnutý mechanizmus analyzovania protokolov na jednotlivých vrstvách

Na začiatku si načítam potrebný obsah z textových súborov do slovníkov a určím si súbor na analyzovanie a úlohu, ktorá sa bude vykonávať. Otvorím súbor pomocou knižnice *scapy* a cez for cyklus analyzujem jednotlivé rámce. V priebehu analyzovanie používam knižnicu *binascii* na *hexlify* a *unhexlify* a natívnu funkciu *.hex()* z Pythonu.

Pri **úlohách 1-3** vypíšem číslo rámca, ktoré inkrementujem po každom vypísanom rámci a vypíšem dĺžku rámca **poskytnutú API** a dĺžku prenášanú **po médiu**. Potom si určím **ether type** rámca a prípadné DSAP/SSAP, ak by rámec nebol *eth II*. Ak je ether type > 1500, vypíšem ho a **MAC adresy** rámca. Potom určujem protokoly pod *eth II* a vypisujem zdrojovú a cielovú IPv4 adresu v prípade, že sa jedná o rámec s IPv4. Po vypísaní IPv4 adresy ju pridám **do listu**, cez ktorý riešim **úlohu 3**. Určím si „offset“ portu a typ protokolu TCP/UDP... a vypíšem. Ak rámec nie je *eth II*, určím, či je 802.3 Raw, LLC + SNAP alebo 802.3 LLC a vypíšem MAC adresy a prípadné podvrstvy 802.3 LLC.

Úlohy **4a – 4f** riešim len ako **filter**, čiže vypíšem na výstup všetky rámce, ktoré spadajú pod tento protokol. Ak sa nič z požadovaného protokolu **nezachytí** v určenom súbore, ani sa **nič nevypíše**.

4g – TFTP rámce radím do komunikácií a postupne vypisujem. Pri prvom TFTP rámci (cieľový port **69**) označím, že **začala** TFTP **komunikácia** a zapamätám si cez štruktúru zdrojový a cieľový port. Všetky rámce, ktoré prídu po začatí komunikácie a vyhovujú týmto stanoveným podmienkam sú zaradené do TFTP komunikácie. Ak príde **nový** rámec s cieľovým portom 69, začala sa nová TFTP komunikácia a zmením podmienky v štruktúre a vypisujem túto novú komunikáciu. Na konci výpisu píšem, **koľko** TFTP **komunikácií** bolo **zachytených** v súbore.

4h – ICMP pakety jednoducho kontrolujem, či sú ICMP a vypisujem rámec a typ ICMP (echo / echo - reply).

4i – ARP dvojice párujem tak, že ak príde ARP request, uložím si ho a nasledujúce ARP kontrolujem, či je reply a ak áno, po výpise tohto reply rámca vypíšem aj že je v komunikácii s jeho ARP dvojicou. Ak príde nový ARP request, vypíšem ho samostatne a nedávam ho do komunikácie. *ARP Probe / ARP Announcement / ARP gratuitous reply* rámce neriešim a len ich vypíšem.

Príklad štruktúry externých súborov pre určenie protokolov a portov

Využívam niekoľko textových súborov pri určovaní protokolov a portov, nachádzajú sa v `/txt_files/` a majú rovnakú štruktúru, napríklad:
`tcp_ports.txt`

07:echo
13:chargen
14:ftp-data
15:ftp-control
16:ssh
17:telnet
19:smtp
35:domain
4f:finger
50:http
6e:pop3
6f:sunrpc
77:nntp
8b:netbios-ssn
8f:imap
185:ldap
1bb:https
1bd:microsoft-ds
438:socks

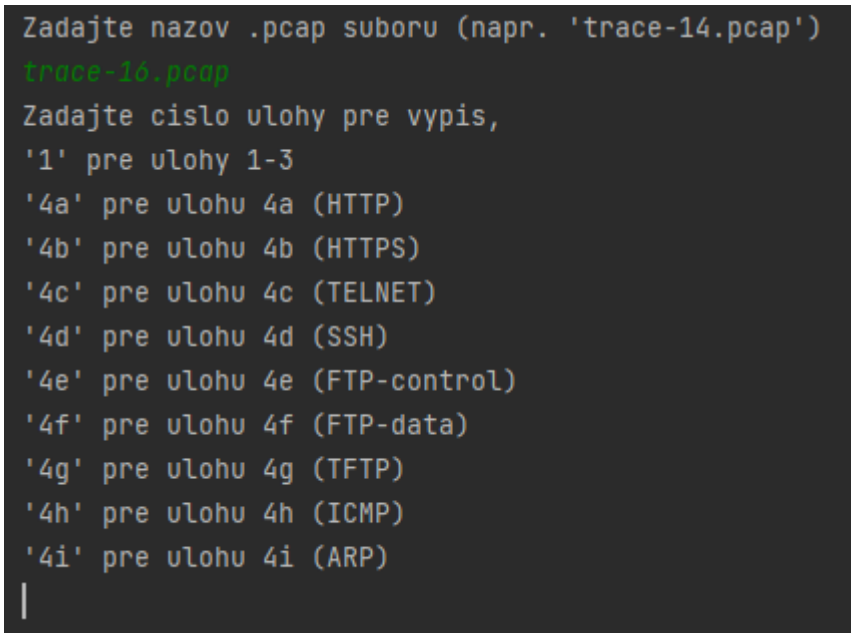
Najprv je napísané číslo portu alebo protokolu, potom oddeľovač „**:**“ a následne meno protokolu / aký port slúži na čo.

Opísané používateľské rozhranie

Program môžem spúšťať s tým, že si manuálne v kóde nastavím hodnoty pre lepšie a rýchlejšie testovanie:

```
575     pcap = scapy.rdpcap("vzorky_pcap_na_analyzu/trace-17.pcap")  # manualne otvorenie pcap na analyzu
576
577     uloha = "1"        # manualne zadavanie ulohy
```

alebo sa dá použiť rozhranie:



POZOR: Pri prvom inpute je dôležité zadať korektný názov súboru, keďže napríklad len ‚***trace-5***‘ by zlyhal, keďže tam nie je „***.pcap***“.

Voľba implementačného prostredia

Na implementáciu zadania som programoval v jazyku Python a použil som PyCharm. Zvolil som si python, kvôli jednoduchej práci s analyzovaním bytov, aj kvôli knižniciam *scapy* a *binascii(hexlify, unhexlify).*