

SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLOGIÍ

Zadanie 1

ANALYZÁTOR SIEŤOVEJ KOMUNIKÁCIE

Anna Yuová

Predmet: Počítačové a komunikačné siete
Akademický rok: 2020/2021
Semester: zimný

1. ZADANIE ÚLOHY

Navrhните 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.

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 prijímajúcich uzlov,
- b) IP adresu uzla, ktorý sumárne prijal (bez ohľadu na odosielateľa) najväčší počet paketov a koľko paketov prijal (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 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.

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.

Program musí mať nasledovné vlastnosti (minimálne):

- > Program musí byť implementovaný v jazyku C/C++ s využitím definovaných knižníc a skompilovateľný a spustiteľný na PC v učebniach. Použité knižnice musia byť schválené cvičiacim
- > Poradové číslo rámca vo výpise programu musí byť zhodné s číslom rámca v analyzovanom súbore.
- > Pre každý rámec uviesť použitý protokol na 2. - 4. vrstve OSI modelu.
- > Pre každý rámec uviesť zdrojovú a cieľovú adresu / port na 2. - 4. vrstve OSI modelu.

Napr. výpis celej komunikácie k bodu 1)

rámec 1

dĺžka rámca poskytnutá pcap API – 68 B

dĺžka rámca prenášaného po médiu – 72 B

Ethernet II

Zdrojová MAC adresa: 00 00 C0 D7 80 C2

Cieľová MAC adresa: 00 04 76 A4 E4 8C

IPv4

zdrojová IP adresa: 192.168.1.33

cieľová IP adresa: 147.175.1.55

TCP

00 04 76 A4 E4 8C 00 00 C0 D7 80 C2 08 00 45 00

00 28 0C 36 40 00 80 06 2B 5A 93 AF 62 EE 45 38

87 6A 04 70 00 50 7E 6C 06 32 56 7D 30 A8 50 10

44 70 97 A0 00 00 80 C2 08 0C 36 40 30 A3 23 35

A2 D5 27 81

rámec 2

dĺžka rámca poskytnutá pcap API – 494 B

dĺžka rámca prenášaného po médiu – 498 B

IEEE 802.3 – Raw

Zdrojová MAC adresa: 00 04 76 A4 E4 8C

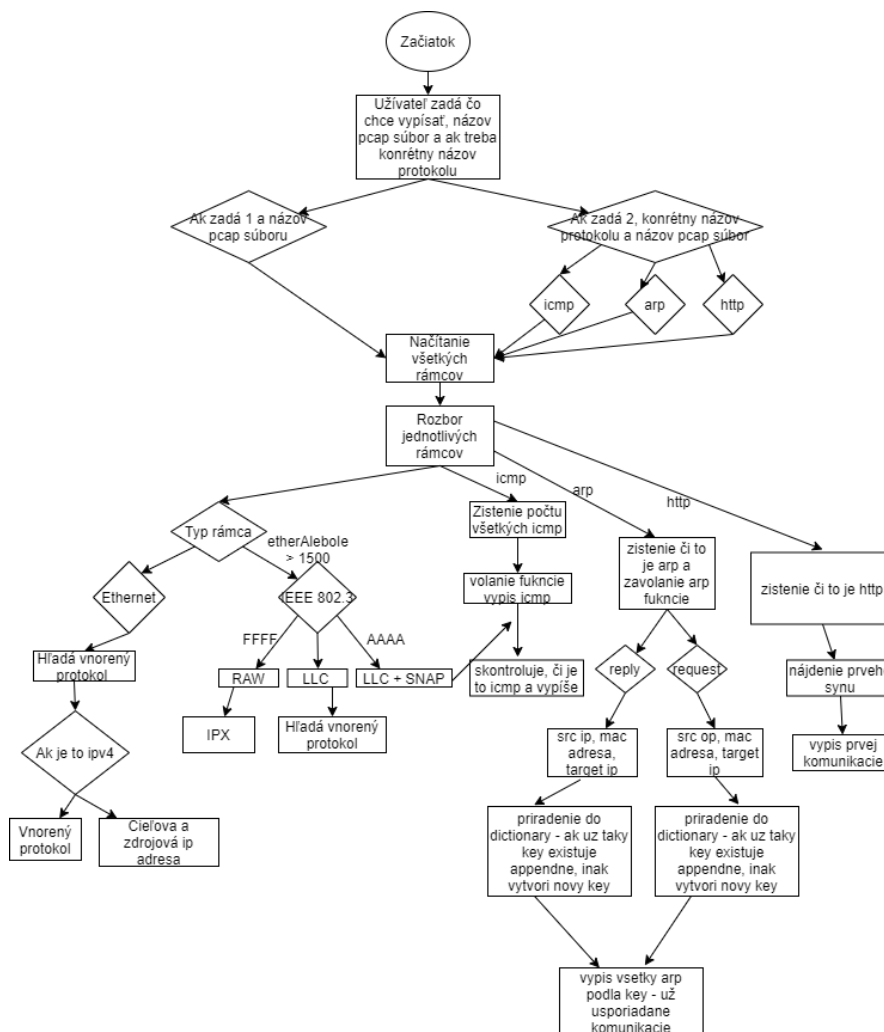
Cieľová MAC adresa: FF FF FF FF FF FF

FF FF FF FF FF FF 00 04 76 A4 E4 8C 01 E0 FF FF

01 E0 00 A1 40 00 80 06 05 B0 93 AF 62 EE 93 AF

63 2A 04 4C 00 50 73 78

2. BLOKOVÝ NÁVRH A OPIS RIEŠENIA



Program na začiatku inicializuje potrebné premenné, otvorí textové súbory na čítanie aj zápis a otvorí .pcap súbor. Zo vstupu prečíta hodnotu, ktorú zadá používateľ – môže zadať buď 1 (ak chce vypísať bod 1,2,3) alebo 2 (ak chce vypísať bod 4). Ak zadá bod 1 následne musí zadať názov pcap súboru, ktorý chce analyzovať. Ak zadá 2, musí zadať konkrétny protokol (iba http, arp a icmp), ktorý chce bližšie analyzovať a ešte aj názov pcap súboru.

Ak si vyberie 1:

Program postupne číta vo for cykle všetky rámce ako prichádzajú (v tomto cykle rovno aj vypisujem koľký rámec analyzujem). Pomocou funkcie hexstr() oddelím \x od 00 a pod. Ďalej si tieto hexadecimálne čísla pomocou funkcie split() rozdelím tak, aby som s nim vedela pracovať ako so stringami. Prechádzam ešte jedným vnoreným cyklom, kde si na konci oddelím posledné znaky, ktoré nie sú hexadecimálne čísla a sú to nejaké znaky z wiresharku (preto aby mi to nezaberalo zbytočne miesto v pamäti, keď s tým nebudem nič robiť). Potom postupne prechádzam cez tieto stringy (v jednom riadku) a ukladám si do premenných tie, ktoré budem potrebovať (napr. na zistenie zdrojového portu si uloží 35. a 36.bajt a podobne). Pri zadaní 1 sa teda zavolá funkcia vypis(), ktorá vypisuje do textovýSubor.txt všetko, čo sa má v bode 1, 2 a 3 vypísať. Začne dĺžkou rámca poskytnutou api (počet bajtov – orátam akoby tie dvojice), potom je dĺžka rámca prenášaná po médiu (počet bajtov + 4 ak je to viac ako 60, pretože minimum musí byť 64 kvôli FCS. Potom zistím, či to je ethernet alebo ieee (ethernet > 1500). Ak je to ieee, zistím o ktorý konkrétny typ 802.3 ide – buď raw, llc + snap alebo lcc. Ak je to llc ešte bližšie určím vnorený protokol z externého textového súboru (externé textové súbory vysvetlím ešte nižšie), raw je ipx a llc + snap ďalej netreba hlbšie analyzovať. Ak je to ethernet, zistím, či to je ipv4 alebo iný protokol. Ak je to iný protokol vypíšem zdrojovú a cieľovú MAC adresu a ak je to ipv4 musím navyše zistiť zdrojovú a cieľovú ip adresu, zdrojový a cieľový port a vnorený protokol (tcp, udp, icmp,...). Potom z tejto funkcie volám funkciu vysielajúce(), do ktorej si pošlem cieľovú ip adresu a tieto adresy si ukladám do dictionary. Ako key je samotná adresa a jej value je počet, koľkokrát sa tam daná adresa nachádza. Vďaka tomuto viem vypísať po všetkých rámcov ip adresy vysielajúcich uzlov a aj adresu uzla s najväčším počtom prijatých paketov (tento výpis dictionary robím na konci mainu, keď skončí analýza jednotlivých rámcov). Ďalej vypíšem pri ipv4 ešte vnorený protokol pre tcp, udp,... a potom už iba samotné rámce. Toto vypisujem do textového súboru textovýSubor.txt.

Ak vyberie 2:

- a ako druhé zadám arp

Prechádzam vo fore rovnako všetky packety ako pri bode 1, akurát volám inú funkciu arpVypis(). V tejto funkcii skontrolujem, či to je naozaj ethernet -> arp a ak je splnená táto podmienka idem si ukladať do dictionary jednotlivé arp. Ako key si pošlem moju ip adresu (zdrojová), mac adresu a cieľovú ip. To sú request arp a ak nájde nejaké arp s týmito adresami, uloží si tie adresy ako key a value budú všetky hodnoty, ktoré treba na výpis. Ak nájde arp reply – čiže ak pri reply sa zhoduje cieľová adresa s mojou zdrojovou adresou, mac adresy, tak si na ten istý key uloží potrebné veci, ktoré si musí vypísať. Ak by prišla ďalšia arp request alebo reply s týmto istým key, appenduje do poľa values potrebné veci na výpis. Konečný výpis arp komunikácie je za forom, keď skončí analýza jednotlivých rámcov. Vo fore prechádzam keys aj v dictionary pre arp request a aj v dictionary pre arp reply. Ak sa tieto keys rovnajú viem že som našla jednu komunikáciu request <-> reply. A vtedy ich vypíšem, ak nájdem samotné requesty, iba vypisujem všetky pod sebou ako prichádzajú. Toto vypisujem do textového súboru vypisArpKomunikacii.txt.

- a ako druhé zadám icmp

Ešte pred prvým cyklom, v ktorom prechádzam všetky rámce, si prejdem ešte raz všetky rámce, aby som si spočítala všetky icmp, ktoré sa tam vyskytujú – je to kvôli tomu, že ak je tých icmp viac ako 20, musím vypísať prvých 10 a posledných 10, preto som si zistila celkový počet icmp. Potom postupujem ako pri bodoch vyššie, kde v cykle prechádzam všetky rámce a zavolám funkciu icmpVypis(). Táto funkcia skontroluje, či to je icmp protokol a ak je, pozrie sa či je počet icmp <20 – vtedy vypisuje všetky doradu, inak vypíše prvých 10 a posledných 10 (mám tam globálnu premennú, aby som vedela, koľkokrát to už vypísal a tu porovnávam, či je <= 10 alebo > počet icmp – 10). Toto vypisuje do textového súboru vypisIcmpKomunikacii.txt.

- a ako druhé zadám http

Prechádzam jednotlivé rámce rovnako ako vo vyšších bodoch v cykle a zavolám funkciu httpVypis, do ktorej si pošlem syn (00010 – to je začiatok komunikácie) a ak nájde prvý syn, uloží si aktuálne porty a tie už nemením. Ak prichádzajú ďalšie http, ktoré majú rovnaké porty s tými, ktoré som si uložila, vypisujem všetky a ak príde fin (00001 – ukončenie komunikácie), tak skončí. Toto vypíše do súboru komunikacie.txt a nie je to podľa zadania.

Komunikáciu pre TFTP som nerobila.

3. ANALÝZA PROTOKOLOV NA JEDNOTLIVÝCH VRSTVÁCH

Analýza na 1.vrstve

Zistím, či rámec Ethernet alebo IEEE 802.3 – ak je 13. a 14. bajt > 1500 tak je to ethernet

-v maine si uloží 13. a 14. bajt

```
if (dlzkaRamca == 13 or dlzkaRamca == 14):  
    etherAleboIe = etherAleboIe + list
```

-vo funkcii vypis() si to premením na z hexa tvaru na decimálne číslo a porovnam, potom ešte konkrétne porovnávam ak je to FF tak to bude RAW a ak je to AA tak to bude LLC + SNAP

```
pomocnaIE = 0  
#vypis ci to je ethernet alebo ieee  
if (etherAleboIePom > 1500):  
    file.write("Ethernet II" + "\n")  
else:  
    file.write("IEEE 802.3 ")  
    pomocnaIE = 2  
    if (typIeeePom == 255):  
        file.write("- RAW")  
        ieeeKonkrétne = "IPX"  
    elif (typIeeePom == 170):  
        file.write("LLC + SNAP")  
    else:  
        file.write("LLC ")  
        try:  
            file.write(llcTypes[typIeeePom])  
        except KeyError:  
            file.write("")  
    file.write("\n")
```

Analýza na 2.vrstve

Pre ethernet mám externý súbor(tie sú vysvetlené neskôr) ethernet.txt a pre iee mám llc.txt

-v maine si úplne na začiatku vytvorím dictionary, do ktorých si poukladám na jednotlivé keys názvy (to som vysvetlila konkrétne v bode o externých súboroch)

```
#do nasledujucich Types si nacitam vsetko co je  
ethernetTypes = nacitajSubory("ethernet.txt")  
llcTypes = nacitajSubory("llc.txt")  
arpTypes = nacitajSubory("arp.txt")
```

-funkcia ktorá mi to poukladá do dictionaries

```
#funkcia, ktora precita externe subory  
def nacitajSubory(nazovSuboru):  
    vysledok = {}  
    subory = open(nazovSuboru, "r")  
    for subor in subory:  
        (key, val) = subor.split()  
        vysledok[int(key)] = val  
    return vysledok
```

-vo vypis() si tam pošlem 13. a 14. bajt – z ethernetTypes dictionary vyberám názov pre ethernet

-z llcTypes vyberám názov pre llc, pre LLC raw je to vždy ipx a llc + snap netreba hlbšie analyzovať

```
if (etherAleboIePom > 1500):  
    try:  
        file.write(ethernetTypes[etherAleboIePom] + "\n")  
        if (ethernetTypes[etherAleboIePom] == 'IPv4'):  
            pomocna = 1  
        if (ethernetTypes[etherAleboIePom] == 'ARP'):  
            pomocna = 5  
    except KeyError:  
        file.write("")
```

```

        file.write("LLC + SNAP")
    else:
        file.write("LLC ")
        try:
            file.write(llcTypes[typIeeePom])
        except KeyError:
            file.write("")
    file.write("\n")

```

Analýza na 3.vrstve

-ak je to ethernet a hlbšie ipv4, idem ešte konkrétne pozrieť protokoly na 3. vrstve, vyberám z protokolTypes dictionary, kde mám uložené hodnoty z protokol.txt

#zistim si aj konkretne typy protokolov z protokol.txt externenc

```

if(pomocna == 1):
    file.write("zdrojova IP adresa: " + ipZdrojova + "\n")
    file.write("cielova IP adresa: " + ipCielova + "\n")
    vysielajuca(ipCielova)
    try:
        file.write(protokolTypes[udpPom] + "\n")
        if (protokolTypes[udpPom] == 'TCP'):
            protokolKonkretne = "tcp"
        if (protokolTypes[udpPom] == 'UDP'):
            protokolKonkretne = "udp"
        if (protokolTypes[udpPom] == 'ICMP'):
            protokolKonkretne = "icmp"
    except KeyError:
        file.write("")

```

Analýza na 4.vrstve

-ak je to ethernet -> ipv4 -> udp alebo tcp alebo icmp, treba ešte hlbšie analyzovať, vyberám z udpTypes dictionary, kde mám uložené hodnoty z udp.txt, tcpTypes dictionary, kde mám uložené hodnoty z tcp.txt alebo icmp dictionary, kde mám uložené hodnoty z icmp.txt

```

if (protokolKonkretne == 'icmp'):
    if (srcPort > dstPort):
        try:
            file.write(icmpTypes[icmpPom] + "\n")
        except KeyError:
            file.write("")
    else:
        try:
            file.write(icmpTypes[icmpPom] + "\n")
        except KeyError:
            file.write("")
            file.write("")
if (protokolKonkretne == 'tcp'):
    if (srcPort > dstPort):
        try:
            file.write(tcpTypes[dstPort] + "\n")
        except KeyError:
            file.write("")
    else:
        try:
            file.write(tcpTypes[srcPort] + "\n")
        except KeyError:
            file.write("")

```

4. EXTERNÉ SÚBORY

Pri bližšom určovaní protokolov do hĺbky čítam informácie z externých súborov. Mám 6 externých súborov – ethernet, llc, udp, tcp, icmp, protokol.

Napr. Ak zistím, že daný rámec je ethernet z ethernet.txt si zistím konkrétne o aký typ ethernetu ide.

Štruktúra súborov:

(napr. z udp.txt a ethernet.txt)

- mám číslo v decimálnom tvare a k tomu priradený názov v jednom riadku, ak mám napríklad názov, ktorý sa skladá z viacerých slov, musí byť medzi nimi pomlčka

udp – Poznámkový blok					ethernet – Poznámkový blok					
Súbor	Úpravy	Formát	Zobraziť	Po	Súbor	Úpravy	Formát	Zobraziť	Pomocník	
7	ECHO				0512	XEROX-PUP				
19	CHARGEN				0513	PUP-ADDR-TRANS				
37	TIME				2048	IPv4				
53	DNS				2049	X.75-INTERNET				
67	BOOTPS				2053	X.25-LEVEL-3				
68	BOOTPC				2054	ARP				
69	TFTP				32821	REVERSE-ARP				
137	NETBIOS-NS				32923	APPLETALK				
138	NETBIOS-DGM				33011	APPLETALK-AARP				
161	SNMP				33024	IEEE-802.1Q-VLAN-TAGGED-FRAMES				
162	SNMP-TRAP				33079	NOVELL-IPX				
500	ISAKMP				34525	IPv6				
514	SYSLOG				34827	PPP				
520	RIP				34887	MPLS				
33434	TRACEROUTE				34888	MPPLS-WITH-UPSTREAM-ASSIGNED-LABEL				
					34915	PPPoE-DISCOVERY-STAGE				
					34916	PPPoE-SESSION-STAGE				
					35020	LLDP				

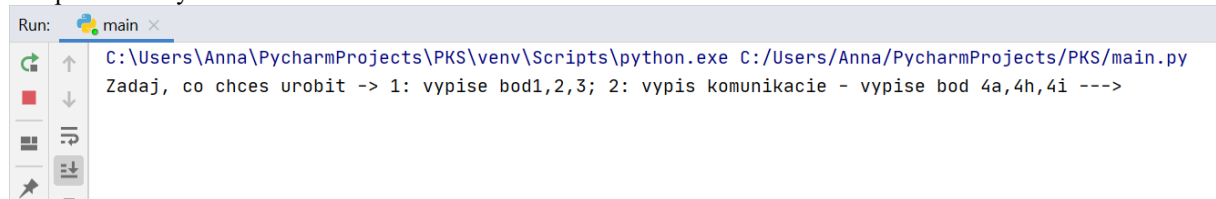
Na začiatku mám funkciu nactajSubory(), do ktorej pošlem daný názov súboru a vytvorím si dictionary, kde si na dané číslo zo súboru – key, uložíam daný názov – value. Vytvorím si páry key-value, kedy si zaplním iba tie indexy, ktoré potrebujem. Takto si zabezpečím, že môžem otvárať veľa súborov bez toho, aby som niečo vo funkcii musela prepisovať alebo písať pre každý externý txt novú funkciu, čo by len zaberalo viac času a pamäte.

```
def nactajSubory(nazovSuboru):  
    vysledok = {}  
    subory = open(nazovSuboru, "r")  
    for subor in subory:  
        (key, val) = subor.split()  
        vysledok[int(key)] = val  
    return vysledok
```

5. POUŽÍVATEĽSKÉ ROZHRAŇIE

Pred spustením programu musíme mať v jednom priečinku spolu kód (main.py), daný .pcap súbor, ktorý chceme analyzovať a externé súbory (llc.txt, ethernet.txt, protokol.txt, icmp.txt, tcp.txt, udp.txt).

Po spustení si vyberiem buď 1 alebo 2:



ak zadám 1- ide sa vykonávať bod 1,2,3

ak zadám 2 – ide sa vykonávať bod 4a alebo 4h alebo 4i

Vyberiem si 1:

a ďalej musím zadať názov pcap súboru, ktorý chcem analyzovať – bez prípony pcap (napr. eth-1, trace-26)

```
main x
C:\Users\Anna\PycharmProjects\PKS\venv\Scripts\python.exe C:/Users/Anna/PycharmProjects/PKS/main.py
Zadaj, co chces urobit -> 1: vypise bod1,2,3; 2: vypis komunikacie - vypise bod 4a,4h,4i ---> 1
Zadaj nazov pcap suboru(eth-1, trace-26): trace-25
Idu sa analyzovat data z: trace-25.pcap

Process finished with exit code 0
|
```

Výstup po zadání 1:

-v textovom súbore textovySubor.txt – výpis rámcov a informácií z nich a na konci textového súboru, sú vysielajúce ip adresy a adresa uzla s najväčším počtom prijatých paketov:

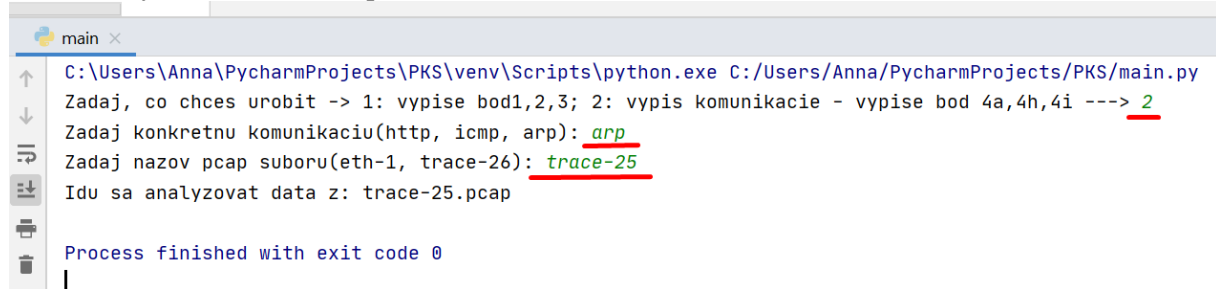
-príklad z trace-25

```
textovySubor – Poznámkový blok
Súbor Úpravy Formát Zobrazit Pomocník
ramec 1
dlzka ramca poskytnuta pcap API - 98 B
dlzka ramca prenasaneho po mediu - 102 B
IEEE 802.3 LLC IPX
zdrojova MAC adresa: 00 04 76 13 97 DF
cielova MAC adresa: FF FF FF FF FF FF
FF FF FF FF FF FF 00 04 76 13 97 DF 00 54 E0 E0
03 FF FF 00 50 00 14 00 00 00 00 FF FF FF FF FF
FF 04 55 30 09 80 00 00 04 76 13 97 DF 04 55 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
01 43 41 26 4F 20 20 20 20 20 20 20 20 20 20
1B FF
-----
ramec 2
dlzka ramca poskytnuta pcap API - 60 B
dlzka ramca prenasaneho po mediu - 64 B
Ethernet II
zdrojova MAC adresa: 00 04 76 A4 E4 8C
cielova MAC adresa: FF FF FF FF FF FF
ARP
FF FF FF FF FF FF 00 04 76 A4 E4 8C 08 06 00 01
08 00 06 04 00 01 00 04 76 A4 E4 8C 93 AF 62 01
00 00 00 00 00 00 93 AF 62 4A 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
-----
ramec 3
-----
IP adresy vysielajucich uzlov:
147.175.98.255
147.175.99.30
147.175.98.238
147.175.99.42
147.175.98.40
147.175.99.62
147.175.1.16
Adresa uzla s najvacsim poctom prijatych paketov:
111 paketov
147.175.98.238
```


Vyberiem si 2:

a ďalej musím zadať konkrétny protokol (buď http alebo arp alebo icmp) a názov pcap súboru, ktorý chcem analyzovať – bez prípony pcap (napr. eth-1, trace-26)

Ak som si vybrala 2 a chcem arp



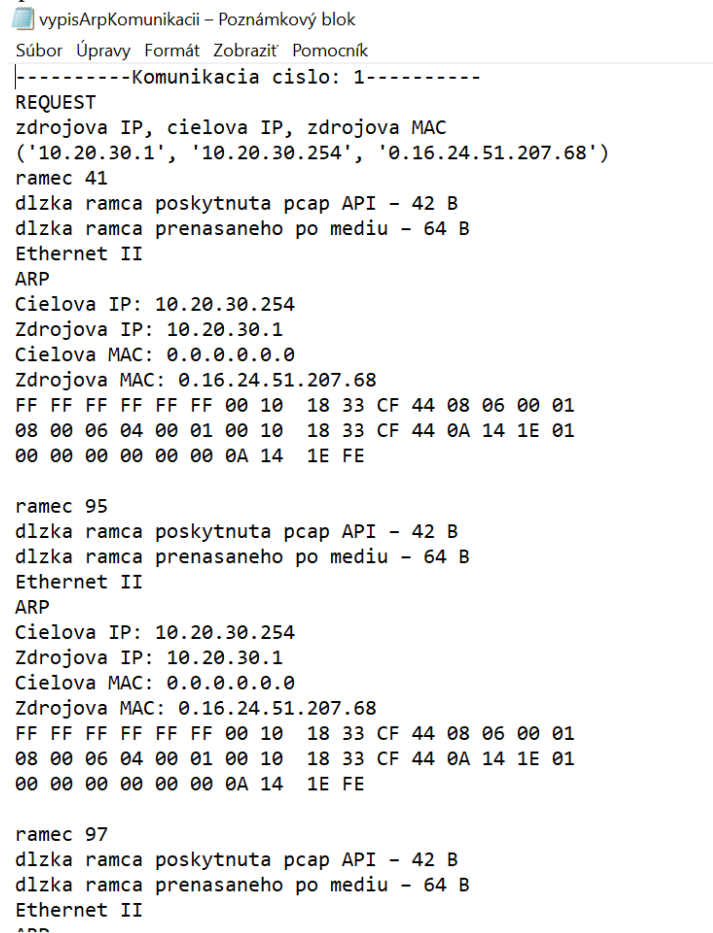
```
main x
C:\Users\Anna\PycharmProjects\PKS\venv\Scripts\python.exe C:/Users/Anna/PycharmProjects/PKS/main.py
Zadaj, co chces urobit -> 1: vypise bod1,2,3; 2: vypis komunikacie - vypise bod 4a,4h,4i ---> 2
Zadaj konkretnu komunikaciu(http, icmp, arp): arp
Zadaj nazov pcap suboru(eth-1, trace-26): trace-25
Idu sa analyzovat data z: trace-25.pcap

Process finished with exit code 0
|
```

Výstup po zadaní 2, arp:

-v textovom súbore vypisArpKomunikacii.txt – výpis arp rámcov postupne ako prichádzajú, ak príde request vypíše preto request a vypíše všetky rámce s rovnakou ip adresou v danom requeste, ak príde request a k tomu reply vypíše Komunikáciu a poradové číslo, všetky rovnaké requesty a za tým reply a za tým Koniec komunikacie poradové číslo komunikácie

-príklad z trace-26



```
vypisArpKomunikacii - Poznámkový blok
Súbor Úpravy Formát Zobrazit' Pomocník
|-----Komunikacia cislo: 1-----
REQUEST
zdrojova IP, cielova IP, zdrojova MAC
('10.20.30.1', '10.20.30.254', '0.16.24.51.207.68')
ramec 41
dlzka ramca poskytnuta pcap API - 42 B
dlzka ramca prenasaneho po mediu - 64 B
Ethernet II
ARP
Cielova IP: 10.20.30.254
Zdrojova IP: 10.20.30.1
Cielova MAC: 0.0.0.0.0.0
Zdrojova MAC: 0.16.24.51.207.68
FF FF FF FF FF FF 00 10 18 33 CF 44 08 06 00 01
08 00 06 04 00 01 00 10 18 33 CF 44 0A 14 1E 01
00 00 00 00 00 00 0A 14 1E FE

ramec 95
dlzka ramca poskytnuta pcap API - 42 B
dlzka ramca prenasaneho po mediu - 64 B
Ethernet II
ARP
Cielova IP: 10.20.30.254
Zdrojova IP: 10.20.30.1
Cielova MAC: 0.0.0.0.0.0
Zdrojova MAC: 0.16.24.51.207.68
FF FF FF FF FF FF 00 10 18 33 CF 44 08 06 00 01
08 00 06 04 00 01 00 10 18 33 CF 44 0A 14 1E 01
00 00 00 00 00 00 0A 14 1E FE

ramec 97
dlzka ramca poskytnuta pcap API - 42 B
dlzka ramca prenasaneho po mediu - 64 B
Ethernet II
ARP
```

a koniec 1. komunikácie je tu, keď nájde reply:

```
08 00 06 04 00 01 00 10 18 33 CF 44 0A 14 1E 01
00 00 00 00 00 00 0A 14 1E FE
```

ramec 166

dlzka ramca poskytnuta pcap API – 42 B

dlzka ramca prenasaneho po mediu – 64 B

Ethernet II

ARP

Cielova IP: 10.20.30.254

Zdrojova IP: 10.20.30.1

Cielova MAC: 0.0.0.0.0.0

Zdrojova MAC: 0.16.24.51.207.68

```
FF FF FF FF FF FF 00 10 18 33 CF 44 08 06 00 01
```

```
08 00 06 04 00 01 00 10 18 33 CF 44 0A 14 1E 01
```

```
00 00 00 00 00 00 0A 14 1E FE
```

REPLY

ramec 42

dlzka ramca poskytnuta pcap API – 60 B

dlzka ramca prenasaneho po mediu – 64 B

Ethernet II

ARP

Cielova IP: 10.20.30.1

Zdrojova IP: 10.20.30.254

Cielova MAC: 0.16.24.51.207.68

Zdrojova MAC: 0.22.71.2.36.64

```
00 10 18 33 CF 44 00 16 47 02 24 40 08 06 00 01
```

```
08 00 06 04 00 02 00 16 47 02 24 40 0A 14 1E FE
```

```
00 10 18 33 CF 44 0A 14 1E 01 00 00 00 00 00 00
```

```
00 00 00 00 00 00 00 00 00 00 00 00 00
```

-----Koniec komunikacie cislo: 1-----

d'alej sa tam už žiaden reply nenachádza, a vypisuje iba requesty

-----Koniec komunikacie cislo: 1-----

REQUEST

zdrojova IP , cielova IP, zdrojova MAC

('0.0.0.0', '10.20.30.1', '0.16.24.51.207.68')

ramec 94

dlzka ramca poskytnuta pcap API – 42 B

dlzka ramca prenasaneho po mediu – 64 B

Ethernet II

ARP

Cielova IP: 10.20.30.1

Zdrojova IP: 0.0.0.0

Cielova MAC: 0.0.0.0.0.0

Zdrojova MAC: 0.16.24.51.207.68

```
FF FF FF FF FF FF 00 10 18 33 CF 44 08 06 00 01
```

```
08 00 06 04 00 01 00 10 18 33 CF 44 00 00 00 00
```

```
00 00 00 00 00 00 0A 14 1E 01
```

ramec 96

dlzka ramca poskytnuta pcap API – 42 B

dlzka ramca prenasaneho po mediu – 64 B

Ethernet II

ARP

Cielova IP: 10.20.30.1

Zdrojova IP: 0.0.0.0

Cielova MAC: 0.0.0.0.0.0

Zdrojova MAC: 0.16.24.51.207.68

```
FF FF FF FF FF FF 00 10 18 33 CF 44 08 06 00 01
```

```
08 00 06 04 00 01 00 10 18 33 CF 44 00 00 00 00
```

```
00 00 00 00 00 00 0A 14 1E 01
```

<

Ak som si vybrala 2 a chcem icmp:

```
main x
C:\Users\Anna\PycharmProjects\PKS\venv\Scripts\python.exe C:/Users/Anna/PycharmProjects/PKS/main.py
Zadaj, co chces urobiť -> 1: vypise bod1,2,3; 2: vypis komunikacie - vypise bod 4a,4h,4i ---> 2
Zadaj konkretnu komunikáciu(http, icmp, arp): icmp
Zadaj názov pcap suboru(eth-1, trace-26): trace-26
Idu sa analyzovať data z: trace-26.pcap

Process finished with exit code 0
```

Výstup po zadání 2, icmp:

-v textovom súbore vypisIcmpKomunikacii.txt – výpis všetkých icmp rámcov, ak je počet icmp menší ako 20, inak vypíše prvých 10 a posledných 10

-príklad z trace-26

-vypísalo 20 rámcov – prvých 10 a posledných 10 - 43, 44, 45, 46, 47, 48, 49, 50, 52, 53, 79, 80, 84, 85, 86, 87, 89, 90, 91, 92

-prvý je 43

```
ramec 43
dĺžka ramca poskytnutá pcap API - 1514 B
dĺžka ramca prenasaneho po mediu - 1518 B
Ethernet II
zdrojová MAC adresa: 00 10 18 33 CF 44
cieľová MAC adresa: 00 16 47 02 24 40
IPv4
zdrojová IP adresa: 10.20.30.1
cieľová IP adresa: 10.20.30.254
ICMP
ECHO
00 16 47 02 24 40 00 10 18 33 CF 44 08 00 45 00
05 DC 04 4D 20 00 80 01 00 00 0A 14 1E 01 0A 14
1E FE 08 00 F1 41 00 01 00 BF 61 62 63 64 65 66
67 68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 76
77 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F
70 71 72 73 74 75 76 77 61 62 63 64 65 66 67 68
69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 76 77 61
62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70 71
72 73 74 75 76 77 61 62 63 64 65 66 67 68 69 6A
6B 6C 6D 6E 6F 70 71 72 73 74 75 76 77 61 62 63
64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70 71 72 73
74 75 76 77 61 62 63 64 65 66 67 68 69 6A 6B 6C
6D 6E 6F 70 71 72 73 74 75 76 77 61 62 63 64 65
66 67 68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75
76 77 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E
```

-na prelome desiateho od začiatku a desiateho od konca sú rámce 53 a 79

```
ramec 53
dĺžka ramca poskytnutá pcap API - 1122 B
dĺžka ramca prenasaneho po mediu - 1126 B
Ethernet II
zdrojová MAC adresa: 00 10 18 33 CF 44
cieľová MAC adresa: 00 16 47 02 24 40
IPv4
zdrojová IP adresa: 10.20.30.1
cieľová IP adresa: 10.20.30.254
ICMP
00 16 47 02 24 40 00 10 18 33 CF 44 08 00 45 00
04 54 04 4F 00 B9 80 01 00 00 0A 14 1E 01 0A 14
1E FE 61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E
6F 70 71 72 73 74 75 76 77 61 62 63 64 65 66 67
68 69 6A 6B 6C 6D 6E 6F 70 71 72 73 74 75 76 77
61 62 63 64 65 66 67 68 69 6A 6B 6C 6D 6E 6F 70
```

```
66 67
ramec 79
dĺžka ramca poskytnutá pcap API - 1514 B
dĺžka ramca prenasaneho po mediu - 1518 B
Ethernet II
zdrojová MAC adresa: 00 16 47 02 24 40
cieľová MAC adresa: 00 10 18 33 CF 44
IPv4
zdrojová IP adresa: 10.20.30.254
cieľová IP adresa: 10.20.30.1
ICMP
ECHO-REPLY
00 10 18 33 CF 44 00 16 47 02 24 40 08 00 45 00
05 DC 04 55 20 00 FF 01 40 A5 0A 14 1E FE 0A 14
1E FE 08 00 F1 41 00 01 00 BF 61 62 63 64 65 66
```

-a posledný je 92

```
ramec 92
dĺžka ramca poskytnutá pcap API - 1122 B
dĺžka ramca prenasaneho po mediu - 1126 B
Ethernet II
zdrojová MAC adresa: 00 16 47 02 24 40
cieľová MAC adresa: 00 10 18 33 CF 44
IPv4
zdrojová IP adresa: 10.20.30.254
cieľová IP adresa: 10.20.30.1
ICMP
00 10 18 33 CF 44 00 16 47 02 24 40 08 00 45 00
04 54 04 4F 00 B9 80 01 00 00 0A 14 1E 01 0A 14
```

Ak som si vybrala 2 a chcem http:

```
main x
C:\Users\Anna\PycharmProjects\PKS\venv\Scripts\python.exe C:/Users/Anna/PycharmProjects/PKS/main.py
Zadaj, co chces urobiť -> 1: vypise bod1,2,3; 2: vypis komunikacie - vypise bod 4a,4h,4i ---> 2
Zadaj konkretnu komunikáciu(http, icmp, arp): http
Zadaj názov pcap suboru(eth-1, trace-26): trace-26
Idu sa analyzovať data z: trace-26.pcap

Process finished with exit code 0
```

Výstup po zadání 2, http:

-v textovom súbore komunikacia.txt – výpis prvej komunikácie, ktorú nájde

-príklad z trace-26

-začiatok je rámec 77 a koniec je rámec 155

```
Subor  Upravy  Formát  Zobrazit  Pomocník
Komunikácia c.1
zdrojova IP adresa: 147.175.98.238
cieľova IP adresa: 147.175.99.42
zdrojový port: 1114
cieľový port : 80

rámec 77
dĺžka rámca poskytnutá pcap API - 62 B
dĺžka rámca prenasaného po mediu - 66 B
Ethernet II
zdrojova MAC adresa: 00 00 C0 D7 80 C2
cieľova MAC adresa: 00 04 76 A4 E4 8C
IPv4
zdrojova IP adresa: 147.175.98.238
cieľova IP adresa: 147.175.99.42
zdrojový port: 1114
cieľový port : 80
00 04 76 A4 E4 8C 00 00 C0 D7 80 C2 08 00 45 00
00 30 0A 0A 40 00 80 06 03 47 93 AF 62 EE 93 AF
63 2A 04 5A 00 50 79 44 45 AB 00 00 00 00 70 02
40 00 92 0E 00 00 02 04 05 B4 01 01 04 02

rámec 155
dĺžka rámca poskytnutá pcap API - 54 B
dĺžka rámca prenasaného po mediu - 64 B
Ethernet II
zdrojova MAC adresa: 00 00 C0 D7 80 C2
cieľova MAC adresa: 00 04 76 A4 E4 8C
IPv4
zdrojova IP adresa: 147.175.98.238
cieľova IP adresa: 147.175.99.42
zdrojový port: 1114
cieľový port : 80
00 04 76 A4 E4 8C 00 00 C0 D7 80 C2 08 00 45 00
00 28 0A 35 40 00 80 06 03 24 93 AF 62 EE 93 AF
63 2A 04 5A 00 50 79 44 4A AD 4E 72 EC 62 50 04
00 00 BE F8 00 00
```

6. IMPLEMENTAČNÉ PROSTREDIE

Vybrala som si programovací jazyk Python v programe Pycharm (verzia 2020 2.2). Python som si zvolila pre funkcie, ktoré ponúka ako `split()`, `hexstr()`, a podobne, ktoré sa mi pri tomto zadání zišli. Oproti C mi to prišlo jednoduchšie ako si to dávať dávať do štruktúr.

Na začiatku treba jedine nainštalovať scapy (File -> Settings -> Project: main.py -> python interpreter -> add -> v hľadani napísať scapy a pridať), aby to bolo v pycharm spustiteľné.