SLOVENSKÁ TECHNICKÁ UNIVERZITA

Fakulta informatiky a informačných technológií

v Bratislave

Počítačové a komunikačné siete

Analyzátor sieťovej komunikácie

Adam Tomčala

Prednášajúci: prof. Ing. Ivan Kotuliak, PhD.

Cvičiaci: Ing. Pavol Helebrandt, PhD.

Čas cvičení: Štvrtok 8:0

Adam Tomčala, Zadanie 1
ID: 103164

Zadanie 1: Analyzátor sieťovej komunikácie

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). 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 byťschopný 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 súborsa 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 byťschopný 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.

Rozbor programu:

- Pri implementácií môjho riešenia som využíval Python 3.9 a knižnicu Scapy, z ktorej som použil funkciu rdpcap () na čítanie pcap súboru a funkciu raw().

- Po otvorení pcap súboru som analyzoval jednotlivé bajty súboru:

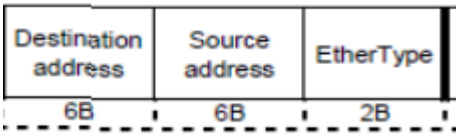
1, Analýza veľkosti paketu

- Veľkosť paketu som analyzoval pomocou funkcie len(paket), dĺžka paketu preneseného po médiu je dĺžka paketu + 4. Ak je dĺžka paketu menej ako 60B, dĺžka paketu preneseného po médiu je 64B.
- Na zistenie a zápis dĺžky paketu v mojom kóde slúži funkcia:

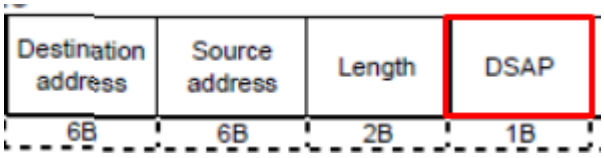
```
def lenght_of_packet(length, oFile):...
```

2, Analýza L2:

- Typ rámca určujem podľa hodnoty na pozícií EtherType (13 a 14 B)



- Ak je hodnota na tejto pozícií viac ako 1500, tak sa jedná o rámec Ethernet II
- Ak je hodnota menšia, ide o rámce IEEE 802.3
- Rámce IEEE 802.3 následne analyzujem podľa následujúceho 15 B

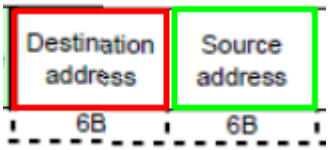


- Ak je hodnota na tomto bajte rovná 255 (“FF”), jedná sa o rámec typu IEEE 802.3 – Raw
- Ak je hodnota na tomto bajte rovná 170 (“AA”) jedná sa o ráme typu IEEE 802.3 s LLC a SNAP
- Inak sa jedná o rámec typu IEEE 802.3 s LLC
- Túto analýzu vykonávam vo funkcií:

```
def type_of_packet(pckt, index, flag, pcap, oFile):
```

```
if int(pckt[12:14].hex(), base=16) > 1500:
    oFile.write("Ethernet II\n")
    print_dst_and_src_mac(pckt, oFile)
    type_of_ether_type(pckt, index, flag, pcap, oFile)
else:
    if int(pckt[14:15].hex(), base=16) == 255:
        oFile.write("IEEE 802.3 - Raw\n")
        oFile.write("IPX\n")
    elif int(pckt[14:15].hex(), base=16) == 170:
        oFile.write("IEEE 802.3 s LLC a SNAP\n")
    else:
        oFile.write("IEEE 802.3 s LLC\n")
        type_of_802(pckt, flag, oFile)
    print_dst_and_src_mac(pckt, oFile)
```

- Pre všetky rámce zisťujem aj ich source a destination MAC adresu.



- Destination adresa sa nachádza na prvých 6 bajtoch paketu a source adresa na 7-12 bajte.
- V mojom kóde mi na to slúži funckia:

```
def print_dst_and_src_mac(pckt, oFile):...
```

- Ďalšiu analýzu vykonávam pre rámce typu Ethernet II a rámce typu IEEE 802.3 s LLC

3, Analýza L3:

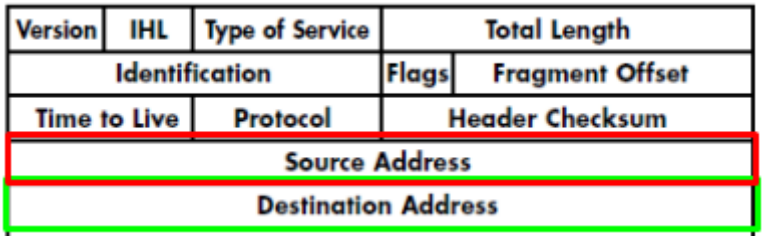
- V tretej vrstve pri rámcoch typu Ethernet II a IEEE 802.3 s LLC, podľa poľa EtherType (resp. Length) zisťujem daný protokol. Protokoly na tretej vrstve mám uložené v externom súbure file.txt. Ak sa hodnota tohto poľa zhoduje s určitým protokolom z externého súboru, vypíše sa. Inak sa vypíše “Nenašiel sa daný typ protokolu”.
- Na analýzu tretej vrstvy pre rámce typu Ethernet II mi slúži funkcia:

```
def type_of_ether_type(pckt, index, flag, pcap, oFile):
```

- Na analýzu tretej vsrtvy pre rámce typu IEEE 802.3 s LLC mi slúži funkcia:

```
def type_of_802(pckt, flag, oFile):
```

- Ďalšiu analýzu vykonávam iba pri protokole IPv4.
- Pri protokoloch IPv4 zisťujem zdrojovú IP adresu (27 – 30B) a cieľovú IP adresu (31 – 34B).



- Zdrojové IP adresy si ukladám to listu + počet paketov, ktoré dané adresy odoslali, pre úlohu číslo 3 (zisťujem aj IP adresu, ktorá odoslala najviac paketov)
- Po zistení IP adries zisťujem aj IHL (kvôli protokolom na vyšších vrstvách).
- Pre výpis IP adries mi slúži funkcia:

```
def ipv4(pckt, index, flag, pcap, oFile):...
```

- Pre výpis všetkých zdrojových IP adries a výpis IP adresy s najväčším počtom odoslaných paketov slúži funkcia:

```
def ipv4_max(oFile):...
```

- Pre ďalšie analýzy potrebujem pole Protocol. Typy IP protokolov sú uložené v externom súbore file.txt
- IPv4 protokol určujem vo funkcií:

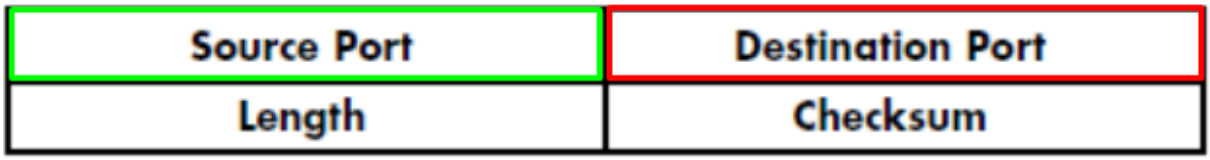
```
def find_next_protocol(pckt, index2, ihl, flag, pcap, oFile):
```

4, Analýza L4:

- Analýzu na štvrtej vrstve robím tak, že pozerám na pole Protocol v IP headeri
- Vypíšem daný typ protokulu, ak sa jedná o protokol TCP alebo UDP, vykonávam ďalšiu analýzu

- Ak sa jedná o ICMP protokol, uložím si index rámca do pomocného listu, pre neskoršiu analýzu ICMP komunikácie
- Pri analýze TCP protokolu pozerám na políčka Source port alebo Destination port, kvôli ďalšej analýze

- Rovnaký postup vykonám aj pri UDP protokole



- Pozícia Source port-u aj destination port-u závisí od IHL v IP headri.

5, Analýza L5:

- Na analýzu TCP protokolu v mojom kóde slúži funkcia:

```
def find_tcp_ports(pckt, index2, ihl, flag, oFile):
```

- Funkcia vypíše typ zdrojového a cieľového portu
- Ak narazí na port 80 HTTP, 443 HTTPS, 23 TELNET, 20 FTP-data, 21 FTP – control alebo 22 SSH uloží sa číslo rámca pre ďalšiu analýzu týchto typov TCP komunikácií (pre každý port je individuálny list)

- Na analýzu UDP protokolu v mojom kóde slúži funkcia:

```
def udp_port(pckt, index2, ihl, flag, pcap, oFile):...
```

- Funkcia vypíše typ zdrojového a cieľového portu

- Ak narazí na port 69 uloží sa číslo rámca pre ďalšiu analýzu TFTP komunikácie

Analýza komunikácií:

1, Analýza TCP komunikácií

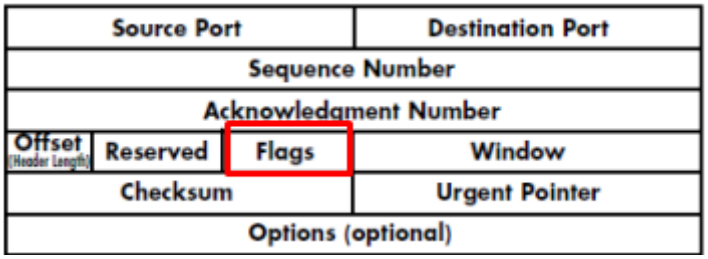
- Na analyzovanie TCP komunikácií slúži funkcia:

```
def tcp_comunication(file, array, port, name, ofile_com):...
```

- Pri analyzovaní TCP komunikácií si zistím porty daného paketu a hľadám také pakety, ktoré majú tieto porty rovnaké (môže byť aj obrátene) –> jedna komunikácia
- Ďalej analyzujem či je komunikácia je kompletná alebo nekompletná
- Za kompletnú komunikáciu považuje takú komunikáciu, ktorá je správne otvorená aj uzatvorená
- Za nekompletnú komunikáciu považuje takú komunikáciu, ktorá je správne otvorená, ale nie uzatvorené
- Na overenie otvorenia komunikácie mi slúži funkcia:

```
def verify_opening(packet1, packet2, packet3, file):...
```

- Za správne otvorenie komunikácie považujem, ak má prvý rámec flag SYN = 1, druhý rámec má flagy SYN a ACK = 1 a tretí rámec má flag ACK = 1



- Na overenie správneho uzatvorenia komunikácie slúži funkcia:

```
def verify_ending(packet1, packet2, packet3, packet4, file):...
```

- Za správne uzavretie komunikacie považujem, považujem následné flagy posledných rámcov:
  - FIN, FIN a ACK, FIN, FIN a ACK
  - FIN, ACK, FIN, ACK
  - FIN, RST

- RST, ACK
- FIN, RST, ACK

2, Analýza ICMP komunikácie:

- Na analýzu ICMP komunikácie mi slúži funkcia:

```
def icmp_comunication(file, array, com_num, ofile_com):
```

- Analýza ICMP komunikácie spočíva v porovnávaní IP adries v rámcoch
- Ak narazím na iné dvojice IP adries, začína nová komunikácia
- Ak je ICMP správa napríklad Time Exceeded, pozerám sa na IP adresy na Bajtoch : 14 + ihl + 24 – 24 + ihl 28 (Cieľová IP adresa) a 14 + ihl + 20 – 14 + ihl + 24 (Zdrojová IP adresa)

3, Analýza ARP komunikácie:

- Na analýzu ARP komunikácie mi slúži funkcia:

```
def arp_comunication(file, array, ofile_com):...
```

- Zisťujem si cieľovú a zdrojovu IP adresu a zdrojovú MAC adresu.
- Následne porovnávam tieto údaje s ostatnými rámcami a hľadám dvojice Request – Reply

4, Analýza TFTP komunikácie:

- Na analýzu TFTP komunikácie mi slúži funkcia:

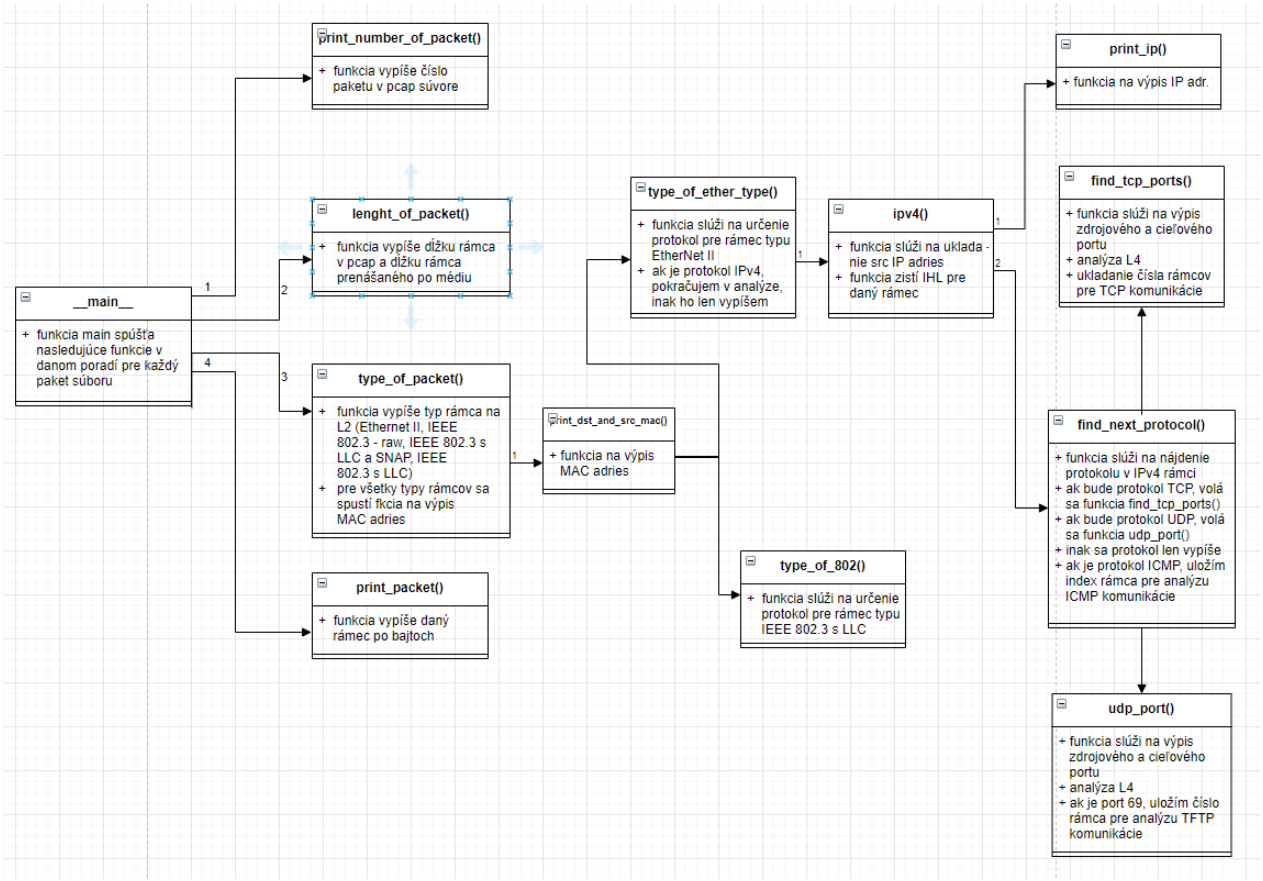
```
def tftp_comunication(file, array, ofile_com):...
```

```
Ak najdem 69 je to nova komunikacia
```

- Pri TFTP komunikácií si zisťujem zdrojový a cieľový port
- Ak sa jeden z nich rovná portu 69, považujem to za začiatok komunikácie
- Následne pracujem s tým portom, ktorý nie je 69 a porovnávam ho s portami ďalšieho protokolu
- Ak sa jeden z portov rovná tomuto portu, pridám protokol do komunikácie
- Ak nájdem nový protokol s portom 69, považujem to za koniec komunikácie a začiatok novej

Adam Tomčala, Zadanie 1
ID: 103164

Blokokový diagram pre analýzu rámcov:



Používateľské rozhranie:

- Po spustení programu je potrebné zadať názov pcap súboru, ktorý chceme analyzovať
- Po zadaní názvu súboru, program automaticky vykoná analýzu všetkých rámcov v pcap súbore, analýzu zapíše to externého súboru ofile.txt
- Po zanalyzovaní rámcov požívateľ zadá číslo, podľa toho aký typ komunikácie bude chcieť zanalyzovať

```
Pre vypis HTTP komunikacie stlacte 1.
Pre vypis HTTPS komunikacie stlacte 2.
Pre vypis TELNET komunikacie stlacte 3.
Pre vypis FTP - control komunikacie stlacte 4.
Pre vypis FTP - data komunikacie stlacte 5.
Pre vypis SSH komunikacie stlacte 6.
Pre vypis TFTP komunikacie stlacte 7.
Pre vypis ICMP komunikacie stlacte 8.
Pre vypis ARP komunikacie stlacte 9.
Pre ukoncenie stlacte 0.
```

  - Zanalyzované komunikácie sa zapíšu do externého súboru ofilecom.txt
  - Pre ukončenie programu stlačte 0

Príklad štruktúry externého súboru:

- File.txt:

```
#Ethertypes
0x0800 IPv4
0x0806 ARP
0x86dd IPv6
0x0801 X.75 Internet
0x0805 X.25 Level 3
0x8035 ARP
0x809b Appletalk
0x80f3 AppleTalk AARP (Kinetics)
0x8100 IEEE 802.1Q VLAN-tagged frames
0x8137 Novell IPX
0x86dd IPv6
0x880b PPP
0x8847 MPLS
0x8848 MPLS with upstream-assigned label
0x8863 PPPoE Discovery Stage
0x8864 PPPoE Session Stage
#LSAPs
0x00 Null SAP
0x02 LLC Sublayer Management / Individual
```

```
0x03 LLC Sublayer Management / Group
0x06 IP (DoD Internet Protocol)
0x0e PROWAY (IEC 955) Network Management, Maintenance and
Installation
0x42 STP
0x4e MMS
0x5e ISI IP
0x7e X.25 PLP (ISO 8208)
0x8e PROWAY (IEC 955) Active Station List Maintenance
0xe0 IPX
0xfe ISO Network Layer Protocols
#IP Protocol numbers
0x01 1 ICMP
0x06 6 TCP
0x11 17 UDP
#TCP ports
0x0016 22 SSH
0x0050 80 HTTP
0x01bb 443 HTTPS
0x0017 23 TELNET
0x0014 20 FTP - data
0x0015 21 FTP - control
#UDP ports
0x0045 TFTP
#
```

- V tomto textovom súbore sa nachádzajú jednotlivé druhy protokolov a portov na rôznych vrstvách

- Icmp_ports.txt:

```
0x00 Echo Reply
0x03 Destination Unreachable
0x04 Source Quench
0x05 Redirect
0x08 Echo Request
0x09 Router Advertisement
0x0a Router Selection
0x0b Time Exceeded
0x0c Parameter Problem
0x0d Timestamp
0x0e Timestamp Reply
0x0f Information Request
0x10 Information Reply
0x11 Address Mask Request
0x12 Address Mask Reply
0x1e Traceroute
```

- V tomto súbore sa nachádzajú ICMP porty