SLOVENSKÁ TECHNICKÁ UNIVERZITA V BRATISLAVE

Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Analyzátor sieťovej komunikácie

Adrián Vančo
ID: 103171
Predmet: Počítačové a komunikačné siete

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**.

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

Adrián Vančo
ID: 103171
Predmet: Počítačové a komunikačné siete

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 byťporadové 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.

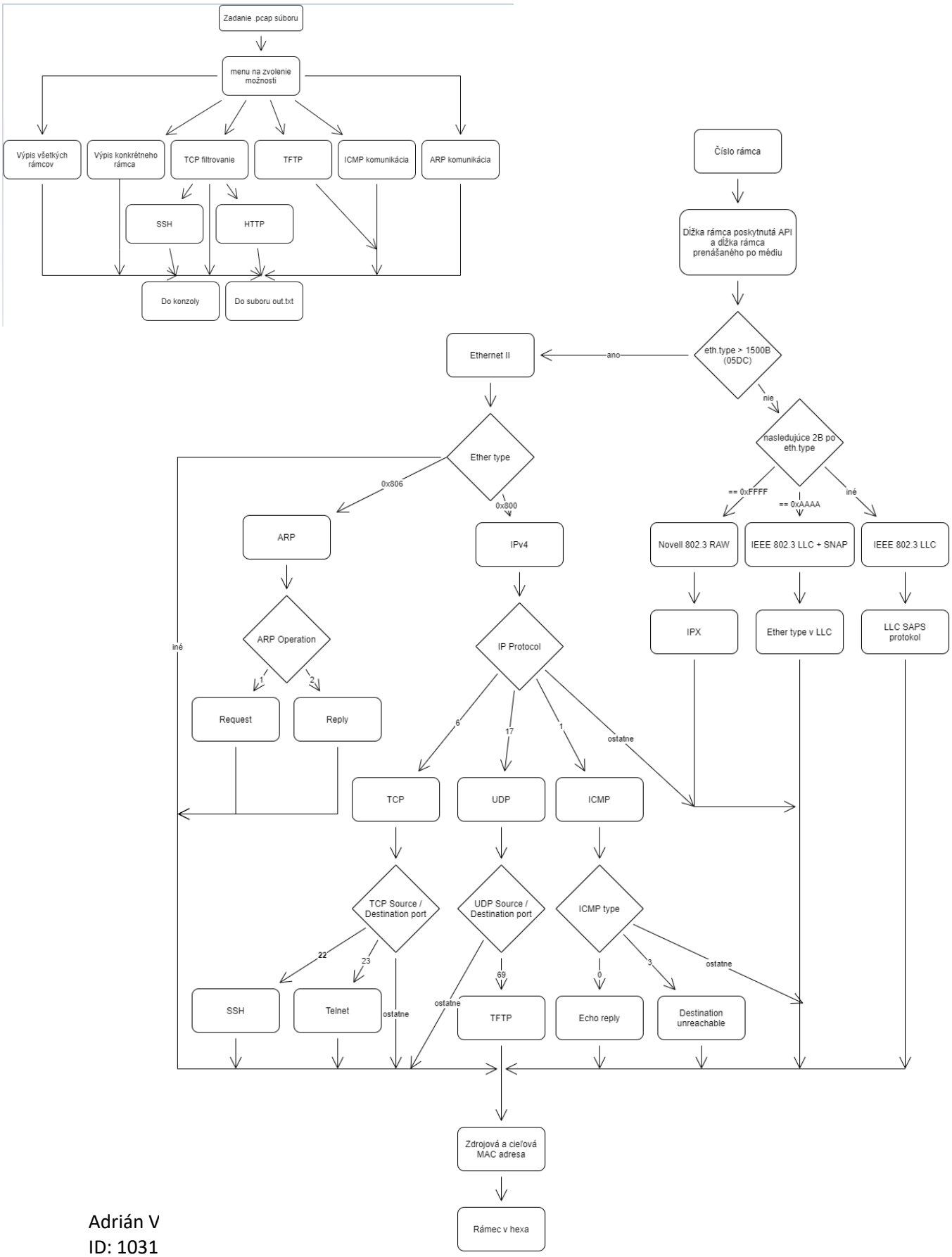
6) V procese analýzy rámcov pri identifikovaní jednotlivých polí rámca ako aj polí hlavičiek vnorených protokolov nie je povolené použiť funkcie poskytované použitým programovacím jazykom alebo knižnicou. **Celý rámec je potrebné spracovať postupne po bajtoch.**

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

Adrián Vančo
ID: 103171
Predmet: Počítačové a komunikačné siete

Blokový návrh (koncepcia) fungovania riešenia

Postup programu **Postup spracovania jedného rámca**



Adrián V
ID: 1031
Predmet. Počítačové a komunikačné siete

Základné informácie

Program vyhotovený v jazyku Python vo verzii 3.8.6
Program je spustiteľný ako .exe
Použité knižnice:

- scapy
- os
- sys

Program používa textové súbory, ktoré musia byť v rovnakom priečinku ako program:

- ether_types.txt
- icmp_types.txt
- ip_protocols.txt
- llc_saps.txt
- tcp_ports.txt
- udp_ports.txt

V každom riadku je najprv hexadecimálna hodnota a následne názov.

Program môže výstup napísať do konzole alebo do súboru s názvom out.txt, ktorý si sám vytvorí. Preto pozor ak by ste už mali podobný priečinok, prepíše vám ho.

Objekty:

```
class udp:
    def __init__(self, frame_number, source_ip, destination_ip, source_port, destination_port):
        self.frames_number = []
        self.frames_number.append(frame_number)
        self.source_ip = source_ip
        self.destination_ip = destination_ip
        self.source_port = source_port
        self.destination_port = destination_port
```

```
class arp:
    def __init__(self, optional, frame_number, sender_ip, sender_mac, target_ip, status):
        self.optional = optional
        self.frames_number = []
        self.frames_number.append(frame_number)
        self.sender_ip = sender_ip
        self.sender_mac = sender_mac
        self.target_ip = target_ip
        self.status = status
```

Adrián Vančo
ID: 103171
Predmet: Počítačové a komunikačné siete

Mechanizmus analyzovania protokolov

1. Načítanie dát

Ako prvé si program vypýta súbor funkciou **load_data()** pre analýzu, funkcia si skontroluje či zadaný súbor existuje a je správneho typu, ak nie očakáva súbor znova pripadne môžeme ukončiť program napísaním „exit“, ale ak súbor existuje pomocou **scapy** ho načítam.

```
#nacitanie dat zo suboru
def load_data():
    filename = input("Zadaj cestu k súboru alebo \"exit\" pre ukončenie programu: ")

    while not os.path.exists(filename) or not filename.endswith('.pcap'):
        if filename == 'exit':
            exit()
        print("Zlá cesta alebo zly typ súboru.")
        filename = input("Zadaj cestu k súboru alebo \"exit\" pre ukončenie programu: ")
        os.path.exists(filename)

    return scapy.rdpcap(filename)
```

Ďalej program cez funkciu **load_nested_dictionary()** načíta slovník, v ktorom sú povnárané slovníky pre jednotlivé **.txt** súbory pre analýzu protokolov.

```
#funkcia na nacitanie slovnika, ktory obsahuje protokoly a porty ako slovniky
def load_nested_dictionary():
    nested_dictionary ={}
    ether_types = {}
    llc_saps = {}
    ip_protocols = {}
    tcp_ports = {}
    udp_ports = {}
    icmp_types = {}

    load_dictionary(ether_types, "ether_types.txt")
    load_dictionary(llc_saps, "llc_saps.txt")
    load_dictionary(ip_protocols, "ip_protocols.txt")
    load_dictionary(tcp_ports, "tcp_ports.txt")
    load_dictionary(udp_ports, "udp_ports.txt")
    load_dictionary(icmp_types, "icmp_types.txt")

    nested_dictionary['ether_types'] = ether_types
    nested_dictionary['llc_saps'] = llc_saps
    nested_dictionary['ip_protocols'] = ip_protocols
    nested_dictionary['tcp_ports'] = tcp_ports
    nested_dictionary['udp_ports'] = udp_ports
    nested_dictionary['icmp_types'] = icmp_types

    return nested_dictionary
```

Adrián Vančo
ID: 103171
Predmet: Počítačové a komunikačné siete

2. Menu

Následne program ponúkne možnosť zapísania do súboru, možnosti analýzy pomocou funkcie **menu()** a na základe zadanej voľby program analyzuje všetky rámce, jeden rámec,....

```
#funkcia na zvolenie co sa ma vykonat
def menu(pcap, nested_dictionary):
    while True:
        file_option = int(input('Zapisat do suboru 1 - áno / 0 - nie: '))
        print()
        print("Dostupné možnosti:")
        print("     1 - pre výpis všetkých rámcov")
        print("     2 - pre vypis konkrétneho rámca")
        print("     3 - pre vypis HTTP komunikácie")
        print("     4 - pre vypis HTTPS komunikácie")
        print("     5 - pre vypis TELNET komunikácie")
        print("     6 - pre vypis SSH komunikácie")
        print("     7 - pre vypis FTP riadiace komunikácie")
        print("     8 - pre vypis FTP dátové komunikácie")
        print("     9 - pre vypis TFTP komunikácie")
        print("     10 - pre vypis ICMP komunikácie")
        print("     11 - pre vypis ARP komunikácie")
        print("     0 - pre ukončenie")

        menu_option = int(input('Zadaj možnosť: '))


    if file_option == 1:
        orig_stdout = sys.stdout
        f = open('out.txt', 'w')
        sys.stdout = f


    if menu_option == 1:
        all_frames(pcap, nested_dictionary)
    elif menu_option == 2:
        frame_number = int(input('Zadaj číslo rámca: '))
        one_frame(pcap, frame_number, nested_dictionary)
    elif menu_option == 3:
        tcp_analization(pcap, nested_dictionary, 'HTTP')
    elif menu_option == 4:
        tcp_analization(pcap, nested_dictionary, 'HTTPS')
    elif menu_option == 5:
        tcp_analization(pcap, nested_dictionary, 'TELNET')
    elif menu_option == 6:
        tcp_analization(pcap, nested_dictionary, 'SSH')
    elif menu_option == 7:
        tcp_analization(pcap, nested_dictionary, 'FTP riadiace')
    elif menu_option == 8:
        tcp_analization(pcap, nested_dictionary, 'FTP datove')
    elif menu_option == 9:
        udp_analization(pcap, nested_dictionary)
    elif menu_option == 10:
        icmp_analization(pcap, nested_dictionary)
    elif menu_option == 11:
        arp_analization(pcap, nested_dictionary)
    elif menu_option == 0:
        exit()
    else:
        print("Zadal si zlú možnosť.")

    if file_option == 1:
        sys.stdout = orig_stdout
        f.close()
```

3. Funkcie

Pre analýzu všetkých rámcov mám funkciu **all_frames()**, ktorá pomocou cyklu prejde všetky rámce a pre každý zavolá funkciu **to_terminal()**.

```
#funkcia na vypis vsetkych ramcov z pcap suboru
def all_frames(data):
    #print(len(data),type(data))
    for frame_number in range(0,len(data)):
        raw = scapy.raw(data[frame_number])
        #print(raw.hex(),"\n")
        to_terminal(raw.hex(),frame_number + 1)
```

Pre analýzu konkrétneho rámca je tu funkcia **one_frame()**

```
#funkcia na vypis konkretneho ramca z pcap suboru
def one_frame(data, frame_number, nested_dictionary):
    if frame_number in range(1,len(data) + 1):
        print()
        raw = scapy.raw(data[frame_number - 1])
        to_terminal(raw.hex(),frame_number, nested_dictionary)
    else:
        print("Zadal si zle číslo rámca.")
```

tcp_analization() hlavná funkcia pre analýzu tcp paketov, podľa dostupných možnosti od možnosti 3 až po 8 vrátane.
udp_analization() hlavná funkcia pre analýzu TFTP komunikácie
icmp_analization() hlavná funkcia pre analýzu ICMP komunikácie
arp_analization() hlavná funkcia pre analýzu ARP komunikácie

Ďalej **to_terminal()** je hlavná funkcia, ktorá zanalyzuje rámec s pomocou ďalších funkcií ako sú **frame_len_print()**, **frame_type_print()**, **mac_addr_src_dst_print()** a **frame_print()**.

```
#vypis informacii o ramci do terminalu
def to_terminal(frame,frame_number):
    print("Ramec",frame_number,":")

    frame_len_print(int(len(frame)/2))

    frame_type_print(int(frame[24:28],16), int(frame[28:32],16))

    mac_addr_src_dst_print(frame[0:24])

    print()
    print("Zdrojova IP adresa:")
    print("Cielova IP adresa:")
    print()
    frame_print(frame)
    print()
    print()
```