

Zadanie 1: Analyzátor sieťovej komunikácie

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.

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 prijímajúcich uzlov,
- b) IP adresu uzla, ktorý sumárne prijal (bez ohľadu na príjemcu) najväčší počet paketov a koľko paketov prijal (berte do úvahy iba IPv4 pakety).

IP adresy a počet poslaných 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 jednoduchšej 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.

Riešenie

Implementačné prostredie

Riešenie zadania je vypracované v programovacom jazyku Python 3 a na vytvorenie používateľského rozhrania bol použitý nástroj tkinter.

Na otvorenie a čítanie z .pcap súborov bol použitý nástroj scapy (<https://scapy.readthedocs.io/en/latest/introduction.html>) a jeho trieda PcapReader, ktorá umožňuje vytvoriť objekt, ktorý postupne generuje rámce zo súboru.

```
def show_all(self):  
    try:  
        file_reader = sc.PcapReader(self.filename.get())
```

Následne bol použitý atribút *wirelen* na získanie dĺžky rámca podľa API a funkcia *raw()* na extrahovanie poľa bytov z rámca, ktoré program následne analyzuje.

```
for api_frame in file_reader:  
    i += 1  
    frame = Frame(i, raw(api_frame), api_frame.wirelen, dicts)
```

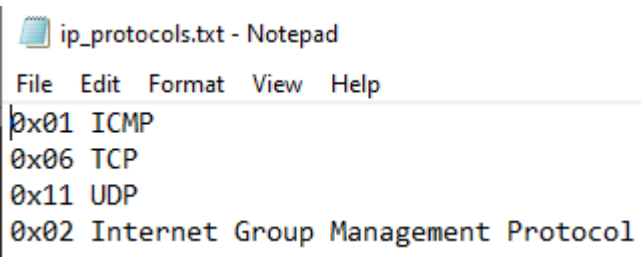
Externé súbory

Samotné rámce sa vytvárajú za pomoci slovníkov, načítaných z externých súborov a uložených do objektu triedy *Protocols*.

```
# Class for keeping all the dictionaries for protocols and ports
class Protocols:
    def __init__(self):
        self.ethertypes = self.make_dict("external_files/ethertypes.txt")
        self.lsapns = self.make_dict("external_files/lsapns.txt")
        self.tcp_ports = self.make_dict("external_files/tcp_ports.txt")
        self.udp_ports = self.make_dict("external_files/udp_ports.txt")
        self.ip_protocols = self.make_dict("external_files/ip_protocols.txt")
```

Referencia na tento objekt je posúvaná všetkým funkciám, ktoré stavajú rámce alebo prekladajú bajty do názvov protokolov pred výpisom. Samotných súborov je päť, jeden pre každý druh protokolu, či typu (pre 2. vrstvu). V súboroch sú uvedené iba hexadecimálne hodnoty.

Príklad súboru:



```
ip_protocols.txt - Notepad
File Edit Format View Help
0x01 ICMP
0x06 TCP
0x11 UDP
0x02 Internet Group Management Protocol
```

Mechanizmus analýzy

Analýza prebieha prvotným postavením rámca do objektu triedy **NetFrame** a následného prečítania a preloženia potrebných bajtov z tohoto objektu ostatnými funkciami v súbore *Main.py*.

Trieda **NetFrame** obsahuje informácie z **druhej vrstvy** rámca (MAC adresy, dĺžka rámca, protokol na 2. a 3. vrstve atď.) a referencie na objekty triedy **Layer3** a **Layer4**, ktoré obsahujú informácie z príslušných vrstiev, v závislosti od použitých protokolov.

Príklad - konštruktor triedy Layer3:

```
# Encapsulates layer 3 of the frame
class Layer3:
    def __init__(self, bytes, type):
        if type == "ARP":
            self.op = bytes[6:8]
            self.smac = bytes[8:14]
            self.sip = bytes[14:18]
            self.dmac = bytes[18:24]
            self.dip = bytes[24:28]
        elif type == "IPV4":
            self.ihl = bytes[0:1]
            self.layer4_prot = bytes[9:10]
            self.sip = bytes[12:16]
            self.dip = bytes[16:20]
```

Postup stavby rámca po vrstvách je zobrazený v priloženom **blokovom diagrame** a je inicializovaný konštruktorom triedy **NetFrame**, ktorý volá funkciu *build()*.

```
class NetFrame:
    def __init__(self, index, bytes, api_len, protocols):
        self.index = index
        self.bytes = bytes
        self.api_len = api_len
        self.real_len = self.set_real_len()
        self.dmac = bytes[0:6]
        self.smac = bytes[6:12]
        self.layer2_type = None
        self.layer3_protocol = None
        self.layer3 = None
        self.layer4 = None
        self.build(bytes, protocols)
```

Takto postavený rámec potom obsahuje všetky potrebné informácie na spracovanie ostatnými funkciami. Medzi tie patria hlavne výstupné funkcie (*print_frame_info()*), ktoré postupne po vrstvách preložia a naformátujú text pred výstupom.

Opis dôležitých tried a funkcií

Trieda **NetFrame**

Súbor: FrameStructure/*NetFrame.py*

Argumenty: poradové číslo rámca, bajty rámca, dĺžka rámca podľa API, objekt *Protocols* so slovníkmi protokolov

Čo robí: vo funkcii *build()* postaví rámec a drží analyzované informácie o ňom enkapsulované vo vrstvách. Stavba rámca je zobrazená v priloženom **blokovom diagrame**.

Funkcia **show_frames()**

Súbor: Main.py

Argumenty: *PcapReader* objekt s rámcami, objekt *Protocols* so slovníkmi protokolov, výstupné okno

Čo robí: Iteruje cez všetky rámce v .pcap súbore a postaví z nich objekt *NetFrame*, následne nad týmto objektom volá *print_frame_info()* na výpis na výstup. Potom pre IPv4 rámce vypíše IP adresy podľa zadania.

Trieda **ArpPair**

Súbor: CommunicationStreams/*ArpPair.py*

Argumenty: rámec s prvou ARP požiadavkou

Čo robí: Je zodpovedná za jednu ARP komunikáciu t.j. dvojicu ARP požiadavka a ARP odpoveď, v resp. všetky ARP požiadavky pred príslušnou ARP odpoveďou.

Funkcia **arp_pairs()**

Súbor: Main.py

Argumenty: *PcapReader* objekt s rámcami, objekt *Protocols* so slovníkmi protokolov, výstupné okno

Čo robí: Iteruje cez všetky rámce v .pcap súbore a postaví z nich objekt *NetFrame*, následne ak je rámec ARP, tak volá pomocné funkcie, ktoré rámce spárujú a ukončené páry vypíšu, potom vypíše zvyšné neukončené páry a iné ARP rámce.

Trieda **TftpComm**

Súbor: CommunicationStreams/TftpComm.py

Argumenty: prvý rámec s portom 69

Čo robí: Je zodpovedná za jednu TFTP komunikáciu tj. prvý rámec na port 69, následne zistí oba dohodnuté porty a drží všetky rámce patriace do jednej komunikácie.

Funkcia **tftp_comm()**

Súbor: Main.py

Argumenty: *PcapReader* objekt s rámcami, objekt *Protocols* so slovníkmi protokolov, výstupné okno

Čo robí: Iteruje cez všetky rámce v .pcap súbore a postaví z nich objekt *NetFrame*, následne ak je rámec TFTP (má jeden port 69), tak vytvorí nový TftpComm objekt, pre všetky rámce potom kontroluje, či nepatria do jednej z vytvorených komunikácií.

Trieda **TcpComm**

Súbor: CommunicationStreams/TcpComm.py

Argumenty: prvý TCP rámec s flagom iba SYN

Čo robí: Je zodpovedná za jednu TCP komunikáciu tj. prvý rámec s flagom SYN a ostatné rámce na rovnakých portoch, drží informáciu o priebehu otvorenia (*handshake_stage*) a ukončenia (*end_stage*) komunikácie .

Funkcia **tcp_comm()**

Súbor: Main.py

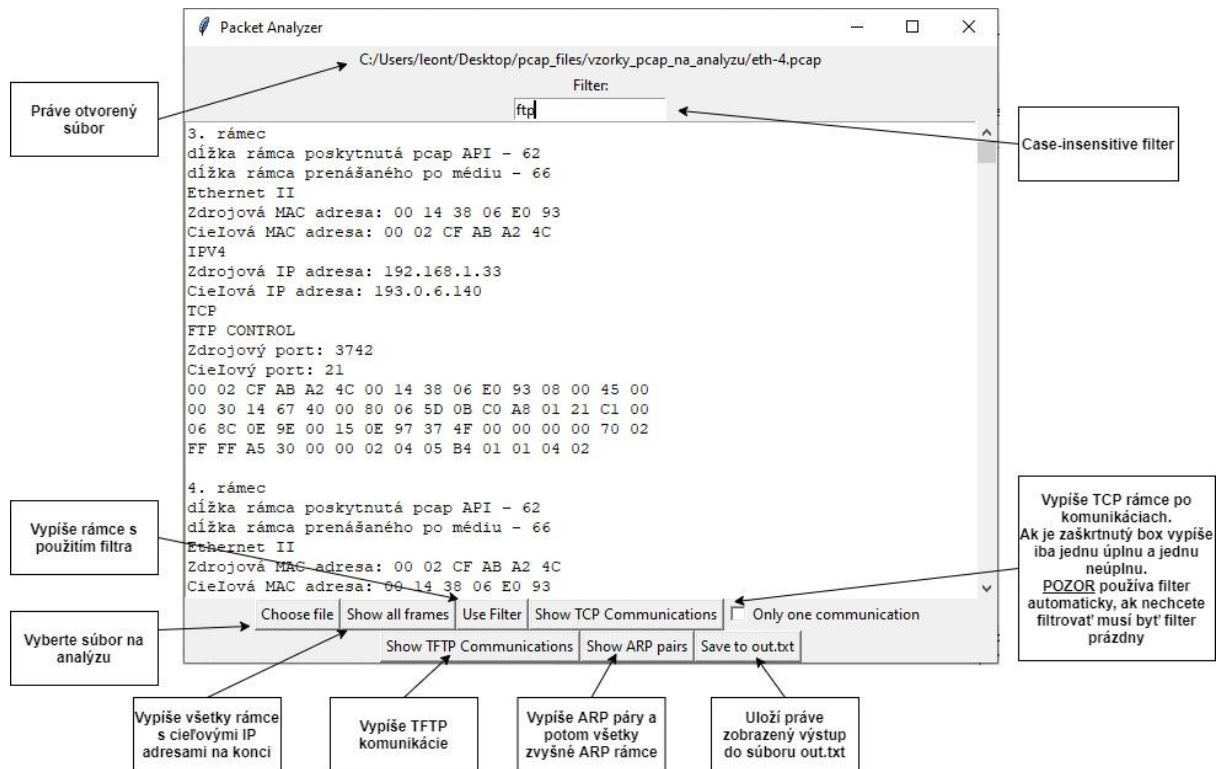
Argumenty: *PcapReader* objekt s rámcami, objekt *Protocols* so slovníkmi protokolov, výstupné okno, filter a či má vypísať všetky komunikácie alebo len jednu úplnú a jednu neúplnú.

Čo robí: Iteruje cez všetky rámce v .pcap súbore a postaví z nich objekt *NetFrame*, následne ak je rámec TCP a zhoduje sa s filtrom (ak je filter použitý), tak vytvorí nový TcpComm objekt, pre všetky rámce potom kontroluje, či nepatria do jednej z vytvorených komunikácií a zvyšné rámce (nepatriace nikam a zároveň bez príznaku SYN) tiež odkladá do iného listu.

Ako pracovať s GUI

Program sa spúšťa štandardne cez command line (spustiť treba súbor GUI.py) :

```
$ python GUI.py
```



Obrázok je aj priložený k dokumentácii pre lepšiu čitateľnosť.

1. Stlačte tlačidlo "Choose file" a vyberte súbor na analýzu.
2. Vyberte jednu z dostupných funkcionalít.
3. Nakoniec môžete výpis uložiť do súboru out.txt

Poznámky k filtru:

1. Filtruje len Ethernet II rámce a na vyšších vrstvách len IPv4 -> TCP/UDP/ICMP.
2. Filtruje aj podľa 3 aj podľa 4 vrstvy aj podľa známych portov (Aj "ipv4" aj "udp" aj "dns" bude fungovať).
3. Po zadaní "ftp" vypíše aj FTP DATA aj FTP CONTROL, selektovať sa dá zadaním konkrétne "ftp data" / "ftp control".
4. Po zadaní "tftp" automaticky spustí výpis TFTP komunikácií, inak berie ako TFTP iba komunikácie na porte 69 (napríklad "udp" vypíše iba tie)