Zadanie 1: Analyzátor sieťovej komunikácie (EN)

Zadanie úlohy

Navrhnite a implementujte programový analyzátor Ethernet siete, ktorý analyzuje komunikácie v sieti zaznamenané v .pcap súbore a poskytuje nasledujúce informácie o komunikáciách. Kompletne vypracované zadanie spĺňa nasledujúce úlohy:

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. (tieto hodnoty nemusia byť rovnaké)
   >
   > 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) Pre IEEE 802.3 s LLC uviesť aj Service Access Point (SAP) napr. STP, CDP, IPX, SAP ...
   >
   > e) Pre IEEE 802.3 s LLC a SNAP uviesť aj PID napr. AppleTalk, CDP, DTP ...
   >
   > f) Zdrojovú a cieľovú fyzickú (MAC) adresu uzlov, medzi ktorými je rámec prenášaný.

   Ostatné **povinné** požiadavky:

   > g) Vo výpise jednotlivé ***bajty rámca usporiadajte po 16 v jednom riadku***. Každý riadok je ukončený znakom nového riadku. Pre prehľadnosť výpisu je vhodné použiť neproporcionálny (monospace) font.
   >
   > h) Výstup musí byť v ***YAML***. Odporúčame použiť knižnicu Ruamel pre Python.
   >
   > i) Odovzdanie do AIS do 2.10.2023 23:59
   >
   > j) Riešenie tejto úlohy musí byť ***prezentované na 4. cvičení***.

   Hodnotenie: **2 body**

2. **Výpis IP adries a vnorených protokol na 2-4 vrstve** pre rámce Ethernet II.

   Pre každý rámec pridajte nasledujúce informácie k výpisu z úlohy 1:

   > a) Vnorený protokol v hlavičke rámca. (ARP, IPv4, IPv6 ...)
   >
   > b) Zdrojovú a cieľovú IP adresu paketu.
   >
   > c) Pre IPv4 uviesť aj vnorený protokol. (TCP, UDP ...)
   >
   > d) Pre 4. vrstvu, tj. vo vnútri TCP a UDP, uviesť zdrojový a cieľový port komunikácie a zároveň, ak niektorý z portov patrí medzi "známe porty", tak uviesť aj názov aplikačného protokolu.

> Pozor, hlavička IP môže mať veľkosť od 20B do 60B.

Ostatné požiadavky:

> e) Čísla protokolov v rámci Ethernet II (pole Ethertype), v IP pakete (pole Protocol) a čísla portov pre transportné protokoly musia byť ***načítané z jedného alebo viacerých externých textových súborov*** (body a, c, d v úlohe 2).
>
> f) Pre ***známe protokoly a porty*** (minimálne protokoly v úlohách 1 a 2) budú ***uvedené aj ich názvy***. Program bude schopný uviesť k rámcu názov vnoreného protokolu aj po doplnení nového názvu k číslu protokolu, resp. portu do externého súboru.
>
> g) Za externý súbor sa nepovažuje súbor knižnice, ktorá je vložená do programu.

Hodnotenie: **1 bod**

3. **Na konci výpisu z úlohy 2** uveďte pre IPv4 packety nasledujúcu štatistiku:

> a) Zoznam IP adries všetkých odosielajúcich uzlov a koľko paketov odoslali.
>
> b) IP adresu uzla, ktorý sumárne odoslal (bez ohľadu na prijímateľa) najväčší počet paketov a koľko paketov odoslal, ak ich je viac, tak uviesť všetky uzly.

Poznámka

- IP adresy a počet odoslaných / prijatých paketov sa musia zhodovať s IP adresami vo výpise Wireshark -> Statistics -> IPv4 Statistics -> Source and Destination Addresses.

Hodnotenie: **1 bod**

4. Váš program ***rozšírte o analýzu komunikácie*** pre vybrané protokoly:

Predpríprava:

> a) Implementujte prepínač ***-p*** (ako protokol), ktorý bude nasledovaný ďalším argumentom a to skratkou protokolu braného z externého súboru, napr. *analyzator.py -p HTTP*. Ak prepínač nebude nasledovaný ďalší argumentom alebo zadaný argument bude neexistujúci protokol, tak program vypíše chybové hlásenie a vráti sa na začiatok. Ako alternatíva môže byť implementované menu, ale ***výstup musí byť zapísaný do súboru YAML***.
>
> b) Výpis každého rámca komunikácie v nasledujúcich filtroch musí spĺňať požiadavky kladené v úlohách 1 a 2 (analýza L2 a L3).

Ak je na vstupe zadaný protokol s komunikáciou so spojením (tj. nad TCP):

> c) Vypíšte ***všetky kompletné*** komunikácie aj s poradovým číslom komunikácie - obsahuje otvorenie (SYN) a ukončenie (FIN na oboch stranách alebo ukončenie FIN a RST alebo ukončenie iba s RST) spojenia. Otvorenie spojenia môže nastať dvomi spôsobmi a zatvorenie štyrmi spôsobmi.
>
> d) Vypíšte ***prvú nekompletnú*** komunikáciu, ktorá obsahuje iba otvorenie alebo iba zatvorenie spojenia.

> e) Na vstupe musíte podporovať všetky nasledujúce protokoly so spojením: **HTTP, HTTPS, TELNET, SSH, FTP radiace, FTP dátové.**

Poznámky

- Otvorenie spojenia sa štandardne deje pomocou 3-way handshake, pošlú sa spolu 3 správy, ale môže nastať prípad, že sa spolu pošlu 4 správy, pre viac informácií pozrite celú kapitolu: TCP Connection Establishment Process: The "Three-Way Handshake".
- Zatvorenie spojenia sa deje pomocou 4-way handshake, ale môžu nastať dve situácie, pozri celú kapitolu: TCP Connection Termination.
- Spojenie tiež môže byť ukončené pomocou flagu RST.
- Paket, ktorý iniciuje začiatok procesu ukončenia spojenia, môže okrem príznaku ***FIN*** mať nastavené aj iné príznaky ako napríklad ***PUSH***.

Hodnotenie: **3 body**

Ak je na vstupe zadaný protokol s komunikáciou bez spojenia (nad UDP):

> f) Pre protokol ***TFTP uveďte všetky rámce a prehľadne ich uveďte v komunikáciách***, nielen prvý rámec na UDP porte 69, ale identifikujte všetky rámce každej TFTP komunikácie a prehľadne ukážte, ktoré rámce patria do ktorej komunikácie. Za kompletnú TFTP komunikáciu považujeme takú komunikáciu, kde veľkosť posledného datagramu s dátami je menšia ako dohodnutá veľkosť bloku pri vytvorení spojenia a zároveň odosielateľ tohto paketu prijme ACK od druhej strany. Viac na TFTP General a TFTP Detailed Operation.

Hodnotenie: **1.5 boda**

Ak je na vstupe zadaný protokol ICMP:

> g) Program identifikuje všetky typy ICMP správ. Echo request a Echo reply (vrátane aj Time exceeded) rozdeľte do kompletných komunikácii na základe nasledujúceho princípu. Najprv je potrebné identifikovať dvojice IP source a IP destination, ktoré si vymieňali ICMP správy a priradiť každej dvojici ich ICMP správy. Následne, Echo request a Echo reply obsahujú v hlavičke polia Identifier a Sequence. Pole ***Identifier*** označuje číslo komunikácie v rámci dvojice IP adries a ***Sequence*** označuje poradové číslo postupnosti v rámci komunikácie. Obe polia môžu byť rovnaké pre rôzne dvojice IP source a IP destination. Z čoho vyplýva, že nová komunikácia je identifikovaná dvojicou IP adries a ICMP polom ***Identifier***. Všetky ostatné typy ICMP správ a ICMP správy Echo request/reply bez páru vypíše ako nekompletnú komunikáciu.
>
> h) Pri každom rámci ICMP uveďte aj typ ICMP správy (pole Type v hlavičke ICMP), napr. Echo request, Echo reply, Time exceeded, a pod. Pri kompletných komunikáciach vypíšte aj ICMP polia ***Identifier*** a ***Sequence***.

Hodnotenie: **1.5 boda**

Ak je na vstupe zadaný protokol ARP:

> i) Vypíšte 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 niekoľko rámcov ARP-Request na rovnakú IP adresu, najprv identifikujte všetky ARP dvojice a vypíšte ich do jednej kompletnej komunikácie bez ohľadu na zdrojovú adresu ARP-Requestu. Následne všetky ARP requesty bez ARP reply vypíšte v jednej nekompletnej komunikácii. Rovnako, ak identifikuje viac ARP reply ako ARP request správ na rovnakú IP, tak všetky ARP reply bez ARP request vypíšte v jednej nekompletnej komunikácii. Ostatné typy ARP správ ignorujeme v rámci filtra.

Hodnotenie: **1 bod**

Ak je IP paket fragmentovaný:

> j) Ak veľkosť IP paketu presiahne MTU, tak paket je rozdelený do niekoľko menších paketov tzv. fragmentov pred odoslaním a následne po prijatí všetkých fragmentov na strane príjemcu je opäť poskladaná celá správa. Pre ICMP filter, identifikujte všetky fragmentované IP pakety a vypíšte pre každý takýto paket všetky rámce s jeho fragmentami v správnom poradí. Pre spájanie fragmentov študujte polia ***Identification***, ***Flags*** a ***Fragment Offset*** v IP hlavičke a uveďte ich aj pri paketoch v takej komunikácii, ktorá obsahuje fragmentované pakety ako ***id, flags_mf a frag_offset***, viac detailov TU. Úloha je iba rošírenie úlohy ICMP filter, čiže protokol na vstupe je rovnaký.

Hodnotenie: **1 bod**

5. Súčasťou riešenia je aj ***dokumentácia***:

> a) Vyžaduje sa prehľadnosť a zrozumiteľnosť odovzdanej dokumentácie ako aj kvalita spracovania celkového riešenia. Za túto časť získa plný bodový zisk študent, ktorý má v dokumentácií uvedené všetky podstatné informácie o fungovaní jeho programu vrátane diagramu spracovávania *.pcap súborov a popis jednotlivých častí zdrojového kódu (knižnice, triedy, metódy, ...).
>
> b) Musí ***obsahovať*** najmä:
>
> - Úvodnú stranu,
> - Diagram (activity, flowchart) spracovávania (koncepcia) a fungovania riešenia,
> - Navrhnutý mechanizmus analyzovania protokolov na jednotlivých vrstvách,
> - Príklad štruktúry externých súborov pre určenie protokolov a portov,
> - Opísané používateľské rozhranie,
> - Voľbu implementačného prostredia,
> - Zhodnotenie a prípadné možnosti rozšírenia.

Hodnotenie: **1.5 body**

Minimálne požiadavky na akceptovanie odovzdaného zadania:

- 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.

- 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()*
- **Výstup** z každej úlohy **musí byť v** súbore **YAML** (.yaml) a v kompatibilnom formáte s YAML (pomôcka, dostanete schémy na testovanie svojich výstupov).
- 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.** Napríklad použitie funkcionality *libpcap* na priamy výpis konkrétnych polí rámca (napr. ih->saddr) bude mať za následok nulové hodnotenie celého zadania.
- Program musí byť **organizovaný tak**, **aby bolo možné** jednoducho **rozširovať** jeho funkcionalitu výpisu rámcov **pri doimplementovaní** jednoduchej úlohy na cvičení.
- Poradové číslo rámca vo výpise programu musí byť zhodné s číslom rámca v analyzovanom súbore (overenie Wireshark).
- Pri **finálnom odovzdaní**, každý rámec vo všetkých výpisoch musí **spĺňať** všetky **požiadavky v úlohách 1 a 2.**
- 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.
- Na prvom cvičení, ktoré nasleduje po uzavretí miesta odovzdania v AISe, musí študent priamo na cvičení doimplementovať do funkčného programu ďalšiu prídavnú funkcionalitu podľa zadania cvičiaceho. Doimplementácia iba rozširuje funkcionalitu analyzátora, nemôže poškodiť alebo znefunkčniť už existujúcu funkcionalitu v analyzátore.
- Dokumentáciu a zdrojový kód implementácie študent odovzdáva v elektronickom tvare do AISu v určenom termíne.
    - Pri celkovom hodnotení sa bude prihliadať aj na efektívnosť vášho programu a jednoduchosť interakcie s ním.
- Body za dokumentáciu budú udelené iba ak bude predvedené plne funkčné riešenie (splnené aspoň minimálne požiadavky) na prvý pokus, bez nutnosti reštartovať program, robiť úpravy v kóde, atď...
- Odovzdané finálne zadanie musí **prejsť úspešne cez plagiátorskú kontrolu**.

- Zadanie, ktoré **nespĺňa** ktorúkoľvek z **minimálnych požiadaviek** vyššie **alebo** nespĺňa **minimálne body** za jednotlivé časti zadania podľa hodnotiacej tabuľky, bude hodnotené **0 bodmi**.

Odovzdanie finálneho zadania do AIS:

- Termín: **16.10.2023 23:59**
- Odovzdáva sa jeden **.ZIP** súbor s názvom <ais_login>.zip napr. xpacket.zip
- ZIP súbor obsahuje nasledujúcu štruktúru:
  - Adresár **Documentation,** v ktorom je dokumentácia v **PDF** formáte.
  - Adresár **Protocols** , v ktorom budú vaše súbory so zadefinovanými portami a názvami protokolov.
  - Ďalej v ZIP súbore bude už iba váš súbor s kódom a vaše vlastné napísané knižnice/moduly. Neodovzdávať štandardné knižnice alebo tie, ktoré je možné inštalovať cez pip.
    - Napr. v pythone to bude súbor main.py a vaše vlastné napísané moduly, ktoré budete importovať.
    - Napr. v C to bude main.c a vaše vlastné includnuté súbory *.c* a *.h*.
- Ukážka štruktúry odovzdaného ZIP súboru:

```
-   Documentation
    -   documentation.pdf
-   Protocols
    -   l2.txt
    -   l3.txt
-   main.py
-   IcmpFilter.py
-   tcpFilter.py
```

Hodnotiaca tabuľka

Číslo úlohy	Názov úlohy	Max bodov	Min bodov
1	Výpis všetkých rámcov v hexadecimálnom tvare	2	1
2	Výpis IP adries a vnorených protokol na 2-4 vrstve	1	5
3	Pre IPv4 packety štatistika	1	

Číslo úlohy	Názov úlohy	Max bodov	Min bodov
4 (c-e)	Analýza protokolov s komunikáciou so spojením	3	
4 (f)	Analýza protokolov s komunikáciou bez spojenia	1.5	
4 (g-h)	Analýza ICMP	1.5	
4 (i)	Analýza ARP	1	
4 (j)	IP fragmentácia	1	
5	Dokumentácia	1.5	0.5
6	Efektívnosť	0.5	-
7	Doimplementácia	1	0.5
	Spolu:	15	7

Príklad možného formátovania externých súborov

```
#Ethertypes
0x0800 IPv4
0x0806 ARP
0x86dd IPv6
#LSAPs
0x42 STP
0xaa SNAP
0xe0 IPX
#IP Protocol numbers
0x01 1 ICMP
0x06 6 TCP
0x11 17 UDP
#TCP ports
0x0015 22 SSH
0x0050 80 HTTP
#UDP ports
0x0035 53 DNS
0x0045 69 TFTP
```

Ukážky výstupu riešenia

Ukážky výstupu riešenia sú súčasťou validátora na overenie správneho formátu vášho výstupu. Obsah rámcov nezodpovedá reálnej komunikácii. Podobne, uvedené IP adresy v desiatkovo-bodkovej notácii nezodpovedajú reálnym hodnotám v rámci.