

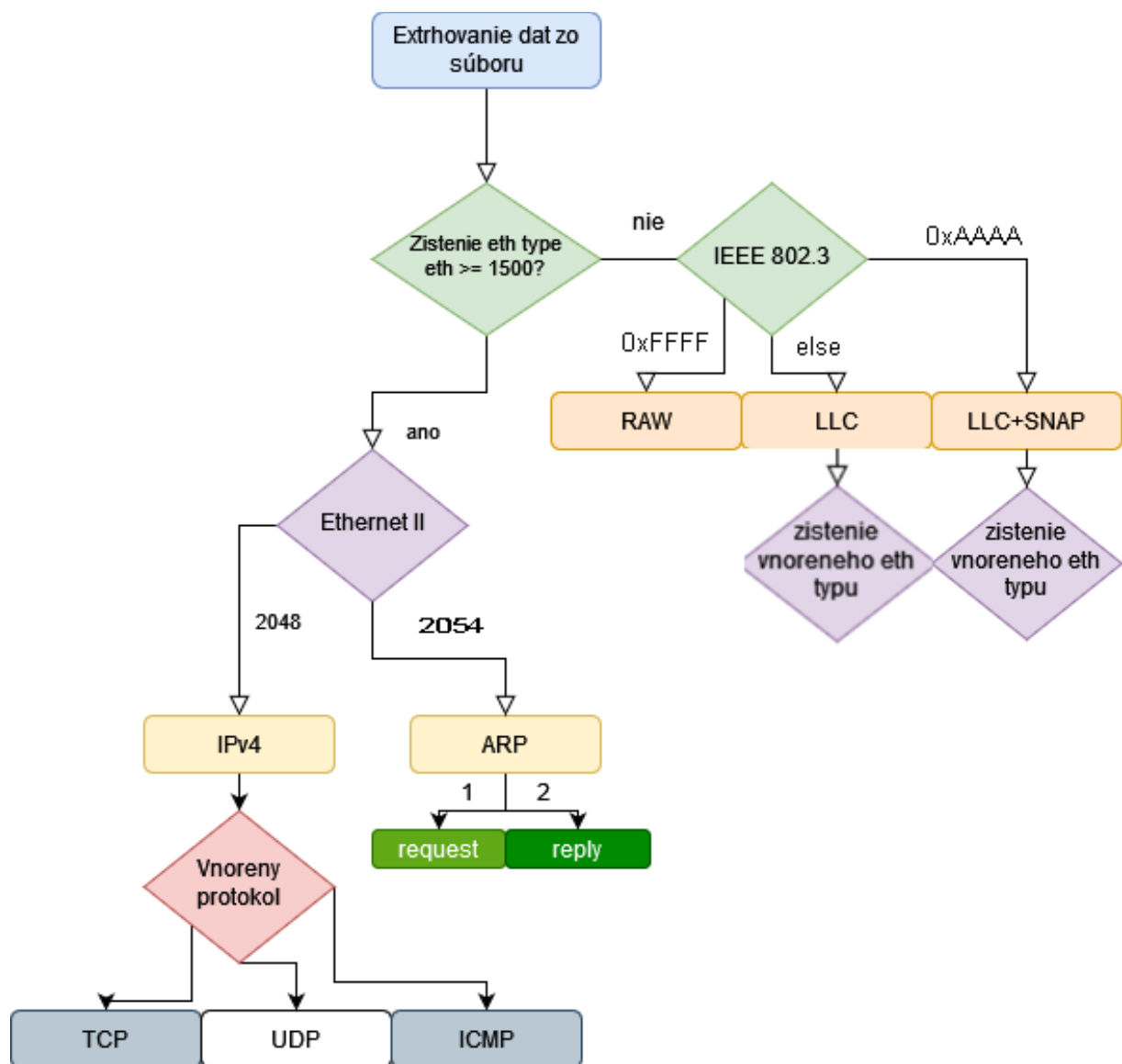
Slovenská Technická Univerzita v Bratislave
Fakulta Informatiky a Informačných Technológií

Počítačové a komunikačné siete
Analyzátor sieťovej komunikácie

Obsah

Diagram	3
Implementácia	4
Externe súbory	4
Štruktúra externého.....	4
Rozbor rámca	5
Analýza komunikácie.....	5
TCP komunikácia	5
Otvorenie a ukončenie komunikácie.....	5
UDP komunikácia.....	6
ICMP komunikácia	6
ARP komunikácia	6
Používateľské rozhranie	6
Hlavne menu.....	6
Ukázkový výstup	7
Implementačné prostredie.....	7
Zhodnotenie.....	7

Diagram



Implementácia

Implementácia je rozdelená do viacerých častí. Hlavnou časťou je funkcia **parse_packet()**, ktorá slúži na samotné analyzovanie rámcov. Rámce sa analyzujú podľa požiadavky používateľa. Funkcia **get_data_frame()**, ktorá vracia údaje z rámca v potrebnom rozsahu, táto funkcia uľahčuje prácu keďže nevzniká redundantný kód. Ďalšou dôležitou funkciou je **inside_protocol()**, ako z názvu vyplýva získavam tu všetky potrebné vnorené údaje.

V tom zadání riešime 4 typy rámcov:

- Ethernet II
- IEEE 802.3 LLC & SNAP
- IEEE 802.3 LLC
- IEEE 802.3 RAW

Externe súbory

Pri implementovaní som používal externé súbory, ktoré mi pomáhali pri načítaní protokolov, well-known portov

Štruktúra externého

```
0x1  ICMP
0x2  IGMP
0x6  TCP
0x9  IGRP
0x11 UDP
0x2F GRE
0x32 ESP
0x33 AH
0x39 SKIP
0x58 EIGRP
0x59 OSPF
0x67 PIM
0x73 L2TP
```

Pri vytváraní súborov som použil následné formátovanie. V programe si súbory načítavam do dictionary kde prvá hodnota predstavuje kľúč a druhá hodnotu , s ktorou ďalej budem pracovať.

Rozbor rámca

Pri rozbere rámca postupujem postupne, takže v prvom rade pozerám mac adresu, ktorá sa nachádza na 0-11 byte, tu si to rozdelím na cieľovú a zdrojovú mac adresu. Následne sa analyzovalo o aký ethernet typ sa jedná, to sa zisťovalo podľa 12-13 bytu, v prípade že hodnota je iná ako ≥ 1500 tak vtedy už vieme že to je dĺžka čiže sa jedná o IEEE.



Ďalšiu dôležitú časť som analyzoval ip adresy, ktoré sa nachádzajú na 26-29 byte pre zdrojovú adresu a 30-33 byte pre cieľovú adresu. Následne pozerám na 23 byte, ktorý mi hovorí o tom, aký vnorený protokol obsahuje. Najdôležitejšia časť pri rozbere je zistenie, aká veľká je hlavička. To zisťujem pomocou IHL, ktoré sa nachádza na 14 byte. Tu sa nachádzajú údaje o tom, aká IP verzia je použitá a veľkosť hlavičky. Túto hodnotu si však musím rozdeliť a to riešim vo funkcii **get_header_length()**, kde si zistím aká veľká je hlavička a následne vrátim rámec posunutý o veľkosť hlavičky.

Analýza komunikácie

TCP komunikácia

Pri TCP komunikácii rozlišujem tieto protokoly: http, https, ssh, telnet, ftp - riadiace a ftp – dátové. Na začiatku si ich roztriedim podľa používateľskej voľby. Dané rámce následne párujem podľa toho, či zdrojová adresa je cieľová adresa, zdrojový port je cieľový toho druhého. Na základe tohto viem povedať, že dané rámce medzi sebou komunikujú.

Otvorenie a ukončenie komunikácie

Bolo potrebné zistiť, či je komunikácia začatá a ukončená správne a či je začatá ale nie ukončená alebo naopak. Na to, aby komunikácia bola správne otvorená musí obsahovať tieto TCP flagy (tcp flagy sa nachádzajú na 13B za ip hlavičkou):

SYN, SYN ACK, ACK

Ak splní tieto podmienky môžem prehlásiť, že komunikácia bola správne otvorená.

Následne pri ukončení môžem očakávať 3 možnosti:

1. Three-Way handshake

- posledné 3 rámce od konca musia obsahovať tieto flagy a. ACK, FIN ACK, FIN ACK

2. Four-Way handshake

- posledné 4 rámce od konca musia obsahovať tieto flagy a. ACK, FIN ACK, ACK, FIN ACK

3. RST

- posledný rámec obsahuje RST alebo RST ACK

UDP komunikácia

Pre toto zadanie som riešil jedine **TFTP** protokol. Tak isto ako pri TCP komunikácií som ich pároval podľa portu a ip adresy. Jediný rozdiel bol v tom, že komunikácia začína portom 69, a následne som pozeral ako sa porty menia, ak už nenájdem ďalšie tak viem že komunikácia je ukončená a idem pozerať ďalšiu.

ICMP komunikácia

V tejto komunikácii párujem na základe ip adres a mac adres. Či je komunikácia kompletná určujem iba podľa **Echo Reply**, **Echo Request** a **Time exceed**. Toto považujem za úplnú komunikáciu. Tento bod by sa dal rozšíriť o kontrolovanie **ID** a **icmp_flags**, ale tým som sa nezapodieval.

ARP komunikácia

Riešenie je rovnaké ako pri TCP komunikácií okrem toho, že rozlišujem, či sa jedná o ARP request alebo ARP reply. Ak nájdem k ARP requestu reply, tak komunikáciu uzatváram. Pre ostatné ARP packety, ktoré nemajú dvojicu, tak ich vypíšem ako nekompletné komunikácie.

Používateľské rozhranie

Používateľské rozhranie je self-explaining a používateľovi by malo byť jasne čo treba spraviť. Najprv zadá cestu k súboru, ktorý chce analyzovať a následne si vyberie jednu z akcií, ktorú chce vykonať.

Hlavné menu

Cesta k súboru by mala vyzerat nasledovne: D:\Python projects\pks22\vzorky_pcap_na_analyzu\trace-26.pcap
D:\Python projects\pks22\vzorky_pcap_na_analyzu\eth-1.pcap

Analyzátor menu

ALL - vypis vsetkych packetov
HTTP - vypis HTTP komunikacii
HTTPS - vypis HTTPS komunikacii
TELNET - vypis TELNET komunikacii
SSH - vypis SSH komunikacii
FTP-CONTROL - vypis FTP-CONTROL komunikacii
FTP-DATA - vypis FTP-DATA komunikacii
ICMP - vypis ICMP komunikacii
ARP - vypis ARP komunikacii
C - zmenit subor
END - ukoncenie

Vyber si vstup

Ukázkový výstup

name: PKS2022/23

pcap_name: eth-1.pcap

packets:

```
- frame_number: 1
  len_frame_pcap: 54
  len_frame_medium: 64
  frame_type: ETHERNET II
  src_mac: B4:B5:2F:74:CB:AE
  dst_mac: 00:02:CF:AB:A2:4C
  ether_type: IPv4
  src_ip: 192.168.1.33
  dst_ip: 147.175.1.55
  protocol: TCP
  src_port: 50032
  dst_port: 80
  app_protocol: HTTP
  hexa_frame: |
    00 02 cf ab a2 4c b4 b5 2f 74 cb ae 08 00 45 00
    00 28 0f 76 40 00 80 06 00 00 c0 a8 01 21 93 af
    01 37 c3 70 00 50 b0 6b 32 16 7b 24 63 25 50 10
    01 02 56 ca 00 00
```

Tento ukázkový výstup predstavuje výpis všetkých rámcov bez použitia filtra. Zobrazuje prvý rámec zo súboru zapísaného v YAML.

Implementačné prostredie

Na implementáciu som si zvolil jazyk Python. Dôvod bol ľahká práca s dictionary, ktoré som v projekte využíval na ukladanie údajov. Taktiež kvôli tomu, že sa s ním ľahko pracuje oproti iným jazykom. V mojom projekte som využil 2 knižnice:

- Scapy – na spracovanie .pcap súboru
- Binascii – na konvertovanie dát z bytes na ascii hodnoty
- ruamel.yaml – na formátovanie výstupu 😊

Zhodnotenie

Moje riešenie dokáže riešiť všetky požiadavky zadania. Vie zistiť či sa jedná o TCP,UDP alebo ARP. Vypísať, či je komunikácia ukončená správne alebo nie. Jediná časť, ktorá nie je úplne kompletná je ICMP analýza kde neudávam ID a flagy. Najhoršia časť a zároveň najťažšia bolo **YAML** formátovanie, ktoré zabralo významnú časť roboty v kombinácii s neustále pribúdajúcimi požiadavkami bolo toto zadanie neprimerane bodovo hodnotené.