Autor: Adam Šípka
AIS ID: 97065

Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 3, 842 16 Bratislava 4

POČÍTAČOVÉ A KOMUNIKAČNÉ SIETE

Dokumentácia k druhému zadaniu

Analyzátor sieťovej komunikácie

ak. rok 2019/20, zimný semester

Adam Šípka

Autor: Adam Šípka
AIS ID: 97065

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:

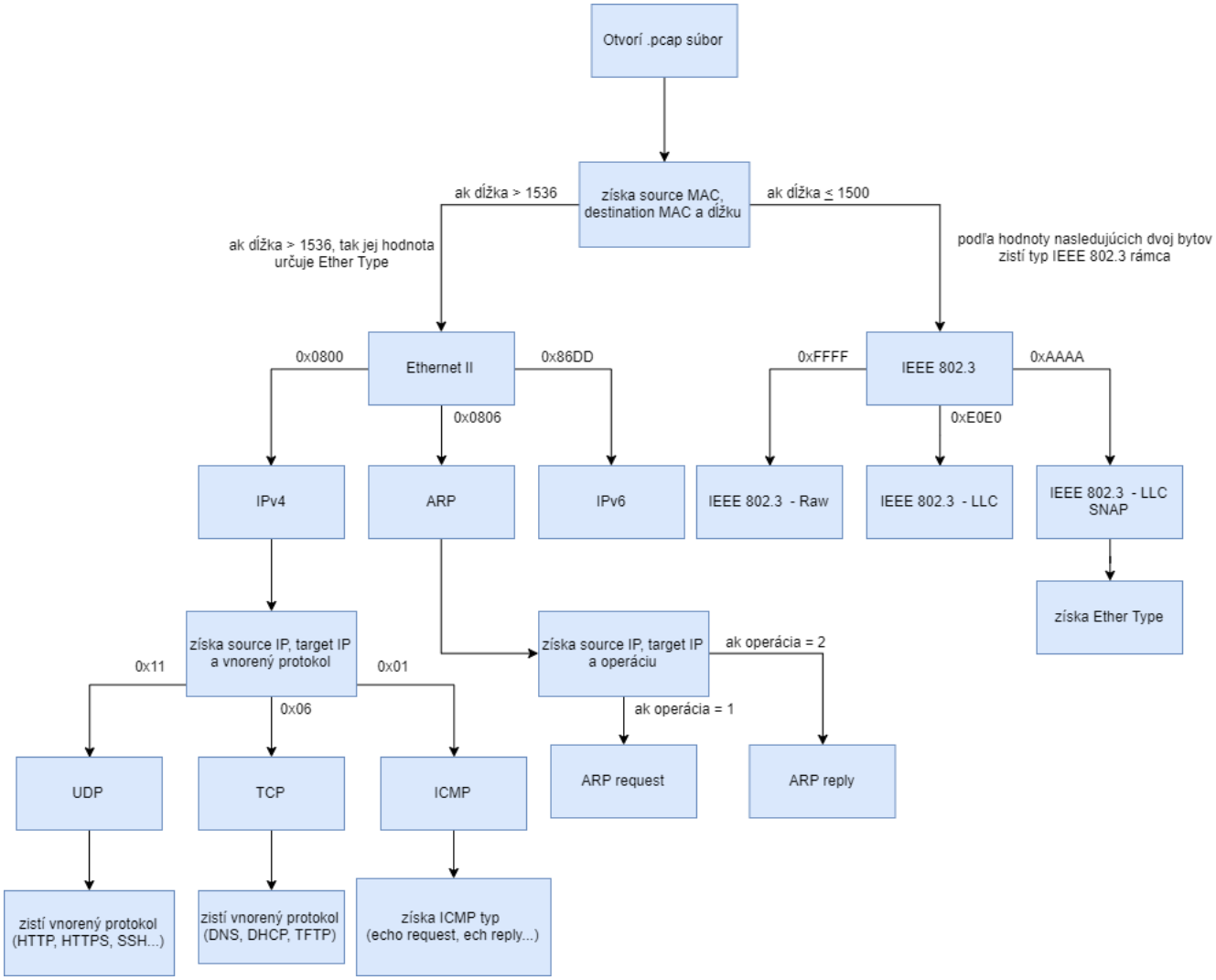
1) **Výpis všetkých rámcov v hexadecimálnom tvare** postupne tak, ako boli zaznamenané v súbore. Pre každý rámec uveďte:
    a) Poradové číslo rámca v analyzovanom súbore.
    b) Dĺžku rámca v bajtoch poskytnutú pcap API, ako aj dĺžku tohto rámca prenášaného po médiu.
    c) Typ rámca – Ethernet II, 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.

2) Pre rámce typu **Ethernet II a IEEE 802.3 vypíšte vnorený protokol**.

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

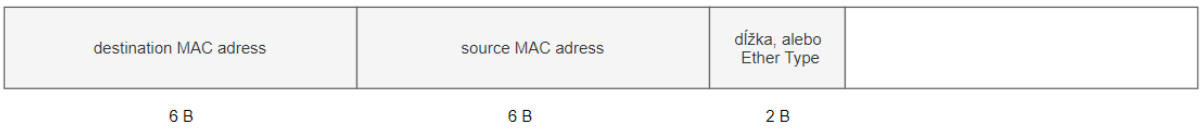
4) V danom súbore analyzujte komunikácie pre zadané protokoly:
    a) HTTP
    b) HTTPS
    c) TELNET
    d) SSH
    e) FTP riadiace
    f) FTP dátové
    g) TFTP, **uveďte všetky rámce komunikácie**, nielen prvý rámec na UDP port 69
    h) ICMP, uveďte aj typ ICMP správy (pole Type v hlavičke ICMP), 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.

Autor: Adam Šípka
AIS ID: 97065



Priebeh analýzy .pcap súboru

Po otvorení súboru a načítaní údajov do bytového poľa môžeme začať s analyzovaním vybraného .pcap súboru. Ako prvé zistíme MAC adresy, pretože tie sa vždy nachádzajú na rovnakej pozícii pri každom type rámca. Určíme ich vo funkcii ***get_mac_addr()***, kde ako argument dostaneme pole bytov. Odtrhneme z neho prvých 14 bytov, kde sa nachádzajú MAC adresy a informácia o dĺžke rámca. Podľa nej zistíme, či sa jedná o Ethernet II, alebo IEEE 802.3. Obe MAC adresy majú veľkosť 6 bytov a dĺžka má 2. Ak je jej hodnota väčšia ako 1536, tak sa jedná o rámec typu Ethernet II, ak je menšia, tak ide o IEEE 802.3. Keď máme Ethernet II, tak sa na daných dvoch bytoch nachádza Ether Type, ktorý určíme pomocou ich hodnoty. Na to slúži funkcia ***get_ether_type***(), kde z externého súboru priradíme danej premennej hodnotu, ak nájdeme zhodu.



Rovnakým spôsobom získame aj informáciu o tom, či sa jedná o IEEE 802.3 raw, alebo IEEE 802.3 s LLC, či IEEE 802.3 s LLC a SNAP. Ak sa jedná o SNAP, tak tiež zisťujeme Ether Type rovnakým spôsobom, ako u Ethernet II. Po tomto je analýza pre IEEE 802.3 ukončená, keďže pri Ethernet II je

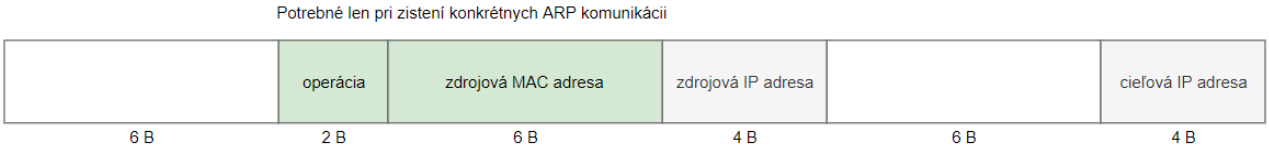
viacej informácii ku ktorým sa môžeme dostať. Ak sme pri Ethernet II získali Ether Type IPv6, tak s analýzou rámca končíme, lebo to nebolo úlohou zadania. Ďalšie dve možnosti ku ktorým sme sa mohli dostať sú IPv4 a ARP. Pri oboch zistíme najprv IP adresy, ale keďže rozloženie ich headerov je odlišné, tak pre každé mám implementovanú osobitnú funkciu.

```
1   #Ethertypes
2   0x800   IPv4
3   0x806   ARP
4   0x86dd  IPv6
5
6   FFFF IEEE 802.3 - raw
7   AAAA IEEE 802.2 - LLC SNAP
8   E0E0 IEEE 802.2 - LLC
9
```

Ukážka externého .txt súboru

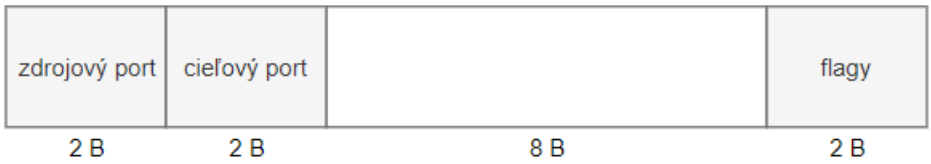
Ak sa jedná o IPv4, tak sa spustí funkcia ***get_ip_addr_ipv4()***, kde okrem zdrojovej a cieľovej IP adresy zistíme aj protokol na tejto vrstve. Okrem toho pridáme zdrojovú adresu do listu, v ktorom sa nachádzajú všetky zdrojové IP adresy, ak v ňom ešte nie je. Okrem toho v ďalšom liste na rovnakom indexe zvýšime hodnotu o jedna. Tento list určuje počet odoslaných rámcov pre každú IP adresu pri Ether Type IPv4. Keďže pri zisťovaní MAC adries sme okrem zistených informácii vrátili aj byty skrátené o 14, tak teraz 'odtrhávaní' bytov nemusíme odrátať chýbajúcich 14, ale trháme opäť od nuly. Celkovo odoberieme 20 bytov, ale len na 9 z nich sú potrebné údaje a ostatné byty nedekódujeme. Na 9 bytoch, ktorých hodnotu chceme zistiť sa nachádza zdrojová a cieľová IP adresa a protokol (TCP, UDP, alebo ICMP).

Pri ARP sa spustí funkcia ***get_ip_addr_arp()***, v nej taktiež zisťujeme IP adresy a typ ARP operácie, ktorá určuje či sa jedná o ARP Reply, alebo ARP Request. Znova ako argument dostaneme byty skrátené o 14, v ktorých sa nachádzali MAC adresy. V tomto prípade odtrhneme 28 bytov, z ktorých nám je reálne potrebných 10. (4 – zdrojová IP, 4 cieľová IP, 2 operácia). Zvyšné byty obsahujú napríklad MAC adresy, ale tie už poznáme, takže znova zisťovať ich nemusíme. Ak chceme vyfiltrovať ARP komunikáciu, tak ich dekódovať môžeme, ušetríme tým počet argumentov vo funkcii, čo sprehľadní kód. Len v tomto prípade využijeme operáciu, ktorú sme zistili. Ak sa operácia rovná 1, jedná sa o ARP request, ak 2, ARP Reply. Komunikácie pri filtrovaní nezgrupujem, rámce vypisujem v takom poradí, v akom sa nachádzajú v .pcap súbore.



Podľa zisteného protokolu na transportnej vrstve postupujme ďalej. Ak sa jedná o TCP, alebo UDP, tak postupujeme ináč ako pri ICMP, keďže v tom prípade musíme zistiť ICMP typ. Vtedy dekódujeme len 1 byte, v ktorom je informácia o type uložená. Daný typ taktiež získame pomocou externého .txt

súboru. Pri TCP a UDP zas zisťujeme vnorený protokol. Ten zistíme so získaných portov, keďže jeden z nich by mal mať hodnotu nejakého TCP, alebo UDP protokolu. Pri UDP zisťujeme len zdrojový a cieľový port, pri TCP zisťujeme aj hodnoty flagov, kde sa nachádzajú informácie, ktoré budú potrebné pri filtrovaní TCP komunikácii. Avšak nezisťujeme všetky, ale len SYN, ACK, FIN a RST, keďže tie sú potrebné na zistenie toho, či komunikácia začína, alebo končí. Tu tiež komunikácie nezgrupujem, ale len vypíšem číslo komunikácie a flagy, ak sa v danom rámci nejaké nachádzajú. Komunikácie určujem podľa portov, ak majú zdrojový a cieľový port stále dve rovnaké hodnoty, tak sa jedná o jednu komunikáciu, ak sa hodnota jedného, alebo oboch zmení, tak už ide o druhú.



Pri vyfiltrovaní TFTP som musel postupovať iným spôsobom, ako pri ostatných protokoloch na aplikačnej vrstve, keďže tam je len prvý rámec odoslaný na port 69. Ak bol takýto rámec odoslaný, uložil som si druhý port a zisťoval, či sa nezhoduje s portom následne odoslaných rámcov. Ak hej, vedel som, že sa jedná o začatú TFTP komunikáciu. Vtedy som oba porty uložil a porovnával ich, či sa zhodujú s ďalšími. Ak áno, vedel som, že sa rovná stále o tú istú komunikáciu. V prípade, že ďalší rámec bol znova odoslaný na port 69, tak už šlo o novú komunikáciu.

Voľba implementačného prostredia, zoznam použitých knižníc a ich využitie

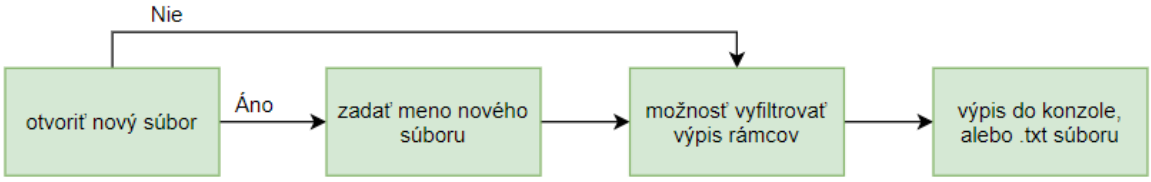
Na implementovanie tohto projektu som použil programovací jazyk Python, keďže má k dispozícii veľké množstvo knižníc a zabudovaných funkcii, ktoré zjednodušili celkovú tvorbu programu a sprehľadnili ho.

Využité knižnice:

- *scapy* – využité na otvorenie .pcap súboru a načítanie jednotlivých rámcov do poľa
- *os* – využité na zistenie existencie .pcap súboru
- *struct* – použité na vytiahnutie potrebných informácii z poľa bytov

Používateľské rozhranie

Je realizované pomocou konzoly, kde máme možnosť si vybrať nasledovnú akciu.



Záver

Mnou vytvorený projekt dokáže analyzovať poskytnuté .pcap súbory a zistiť z nich údaje, ako napríklad MAC adresy, IP adresy, cieľové a zdrojové porty a podobne. Taktiež vie zistiť celkovú veľkosť rámca a IP adresy, z ktorých sa rámce posielali. Okrem toho vie zistiť do ktorej komunikácie rámec patrí a podobne. Výsledné informácie vypisuje buď do textového súboru, alebo do konzole.