**Počítačové a komunikačné siete**

**Cvičenia**

**Ak. Rok 2019/2020, zimný semester**

Zadanie 2: **Analyzátor sieťovej komunikácie**

Autor: **Denis Piovarči**

Dátum: **1.12.2019**

Úlohou zadania bolo navrhnúť a implementovať programový „post“ analyzátor Ethernet siete, ktorý analyzuje komunikácie v sieti zaznamenané v .pcap súbore a poskytne informácie o komunikáciách.

Postupne opíšem moje riešenie zadania a logiku použitú v implementácií funkcií.

1. ***Opis používateľského rozhrania***

Užívateľ si môže hneď po spustení programu vybrať, ktorú funkcionalitu chce vypísať. Má na výber z viacerých možností, pričom každá možnosť reprezentuje jednu podúlohu zadania 1. V programe je to vyriešené jednoduchým switch-om. Každý case spustí jednu funkciu, ktorá vykonáva požadovanú úlohu. Následne ako si užívateľ vybral funkciu, musí zadať prístupovú cestu k input súbor, čiže súboru s príponou .pcap. Po zadaní cesty nasleduje výpis do textového súboru, ktorý sa nachádza v priečinku output. Každá funkcia má vlastný súbor pre výpis.

1. ***Načítavanie hodnôt z externého súboru***

V tejto podkapitole sa budem venovať logike, ako som načítaval konštanty z externého súboru. V priečinku „konstanty“ je niekoľko textových súborov, v ktorých sa nachádzajú konštanty aj s ich menom. Na začiatku každej funkcie si volám funkciu s názvom void open\_xy(), ktorá mi otvorí požadovaný textový súbor, načíta konštantu – hodnotu, a zároveň názov protokolu či portu do globálnych premenných definovaných na začiatku programu. Následne súbor zavrie.

1. ***Hlavné funkcie***

Ďalej by som rád opísal logiku v hlavných funkciách programu.

* *Void vsetko()*

Funkcia všetko sa spustí po stlačení klávesy ‘x‘ v menu. Táto funkcia slúži na výpis všetkých rámcov súboru aj s opisom. Ku každému rámcu je vypísané aj číslo rámca, dĺžka rámca poskytnutá v pcap API alebo prenášané po médiu, zdrojová a cieľová MAC adresa, určenie či ide o Ethernet II alebo IEEE 802.3. Taktiež program dokáže zistiť vnorený protokol na 2.-4. vrstve OSI modelu.

Ako prvé som otvoril pcap súbor. Načítal som si jeho hodnoty do štruktúry a taktiež hodnoty konštánt z externých textových súborov. Nasleduje výpis poradového čísla rámca, zistenie dĺžky rámca pomocou prepínača ->len. Následne zisťujem či sa jedná o Ethernet alebo 802.3 + vnorené protokoly. Ak sa jedná o Ethernet, tak pomocou štruktúry si ukladám IP adresy uzla, ktoré po výpise všetkých rámcov taktiež vypíšem. Tiež si odkladám aj dĺžku rámca, aby som mohol zistiť, ktorý uzol odvysielal najväčší počet bajtov. Následne zisťujem zdrojovú MAC adresu, ktorú popisujú 6. až 11 B, cieľovú MAC adresu (prvých 6 B) a nasleduje výpis celého rámca. Po výpise rámca sú vypísané zdrojové IP adresy, a IP adresa, ktorá odvysielala najväčší počet bajtov.

* *Funkcie pre výpis TCP portov*

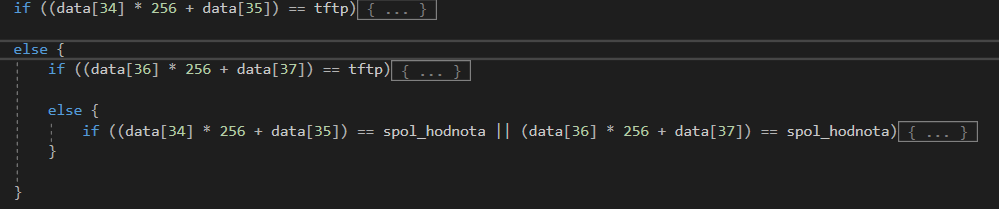
Tieto funkcie opíšem dokopy, pretože sú viacmennej identické. Líšia sa len konštantou portu. Jedná sa o funkcie výpisu http, https, telnet, ssh, ftp control, ftp data. Funkcionalitu opíšem na funkcií pre výpis http komunikácie.

Po otvorení pcap súboru a načítaní konštánt zisťujem pre každý rámec či sa jedná o Ethernet II, IPv4, TCP a potom teda daný port. Všetky prevzaté hodnoty z kódu sú prevedené do desiatkovej číselnej sústavy. Každú z hľadaných vecí som našiel na presnom mieste bajtu (bajtov) podľa ťaháku.

Po podmienkach vyobrazených nasledujú výpisy rovnaké ako vo funkcií všetko. Ich logika je rovnaká, preto ich nebudem opätovne opisovať. Navyše sú pridané výpisy IPv4, TCP a zdrojový či cieľový port. Čísla portov sú taktiež prevedené na int hodnoty a porovnávané s konštantami. Miesto ich nálezu v „hexaguláši“ som tiež zistil z ťaháku.

Výpis pozostáva iba z prvej úplnej komunikácie tj. obsahuje otvorenie komunikácie medzi dvoma IP adresami SYN, a ukončenie na konci pomocou 2xFIN, FIN+RST alebo iba RST. funkcie je rozdelená na dve polovice. V prvej hľadám všetky komunikácie, ktoré pcap súbor obsahuje. V druhej nájdem spomedzi uložených komunikácií prvú úplnú, a tú vypíšem.

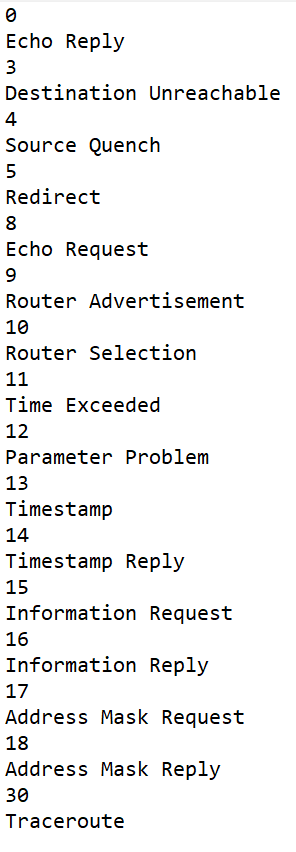
* Funkcia port\_tftp()

Táto funkcia je už o niečo komplikovanejšia. Rozdiel oproti minulým funkciám spočíva v tom, že, port TFTP sa nachádza v UDP, preto je podmienka trocha iná. Adresa portu sa nachádza na 34. a 35 bajte alebo na 36. a 37. bajte, pričom číslujem vrátane nuly. Nasleduje podmienka, ktorá zisťuje či sa na jednom z týchto dvoch možností nachádza číslo port TFTP. Ak áno, vypíšem. Bajty hore uvedené, kde sa nenachádza konštanta portu TFTP, si uložím do premennej s názvom spol\_hodnota. Nasledujúce rámce, kde sa na daných bajtoch nachádza táto spoločné hodnota, vypíšem. Túto logiku odhaľuje Obrázok č. 1.

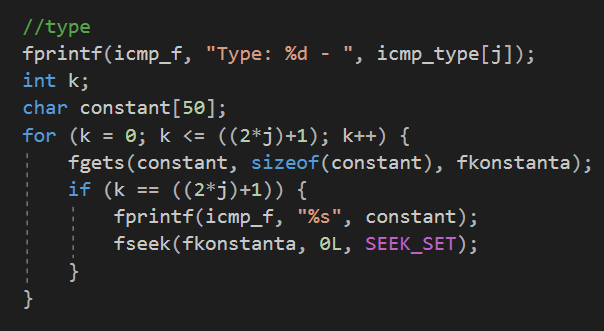
Obrázok č. 1

* Funckcia protokol\_icmp()

Funkcia protokol\_icmp má za úlohu vypisovať všetky rámce typu icmp. ICMP je protokol IPv4. Funkciu zisťuje a vypisuje všetky veci ako predošlé funkcie až na porty, ktoré sú nahradené typom. Typ sa nachádza na 34 bajte. Všetky typy sú uložené v externom textovom súbore icmp\_type.txt. V tomto súbore sa nachádzajú konštanty a pod nimi je uvedený názov (Viď. Obrázok č. 4). Po zistení konštanty otvorím a načítam súbor, presuniem sa na jej pozíciu a o riadok nižšie. Názov vypíšem. Algoritmus je zobrazený na Obrázku č. 2.

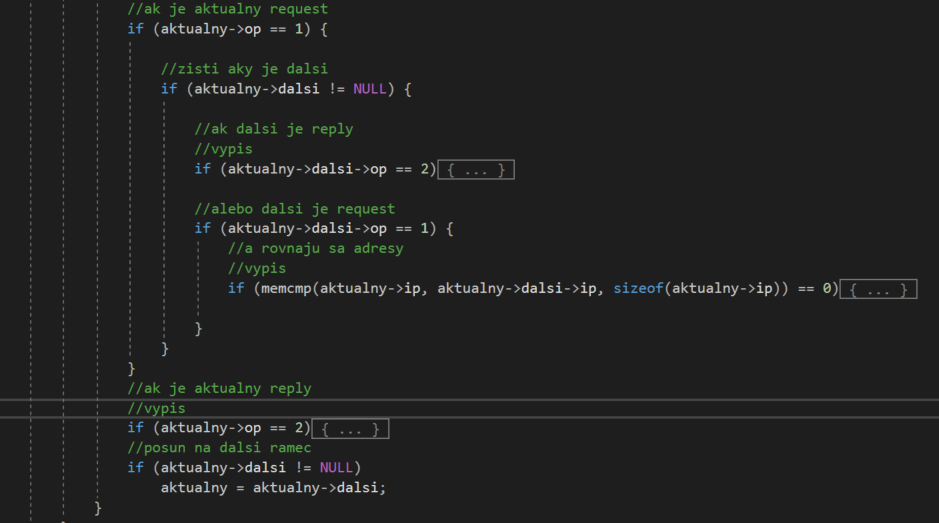


Obrázok č. 2

  
Obrázok č. 3

* Funkcia komunikacia\_arp()

Funkcia slúži na výpis arp dvojíc komunikácie. Logika tejto funkcie je nasledovná. Po otvorení súboru pcap a načítaní konštánt som zadefinoval štruktúru, do ktorej zapisujem všetky rámce obsahujúce arp. Po načítaní IP adries a typu, odznova začnem prehľadávať súbor pcap a začnem s výpisom. Komunikácia vždy začína typom arp-request. Po request nasleduje buď ďalší request, ktorý má zhodnú IP adresu (konkrétnejšie cieľovú IP adresu) alebo reply s taktiež zhodnou IP adresou ale už zdrojovou IP adresou. Ak nie sú splnené podmienky pre plnohodnotnú komunikáciu, výpis rámca sa neuskutoční. Tomu nasvedčujú aj moje podmienky, ktoré nebudem rozoberať do detailov. Na Obrázku č. 4 je podľa komentárov očividné, o čom daná podmienka hovorí.

Obrázok č. 4