Slovenská technická univerzita
Fakulta informatiky a informačných technológií
Ilkovičova 2, 842 16 Bratislava 4

Zadanie 1 – Analyzátor sieťovej komunikácie
Dokumentácia

Dominik Vasko

Študijný program: INFO4
Ročník: 3
Predmet: PKS
Cvičenia: Dr. Ing. Michal Ries , utorok 16:00
Akademický rok: 2017/2018

Zadanie úlohy

Navrhnite a implementujte programový “post” analyzátor Ethernet siete, ktorý analyzuje komunikácie v sieti zaznamenané v súbore a poskytuje nasledujúce informácie o komunikáciách. Vypracované zadanie musí spĺňať nasledujúce body:

1) Výpis všetkých komunikácií, t.j. všetkých rámcov v hexadecimálnom tvare postupne tak, ako boli zaznamenané v súbore. Pre každý rámec uveďte:
- Poradové číslo rámca v analyzovanom súbore.
- Dĺžku rámca v bajtoch poskytnutú pcap API, ako aj dĺžku tohto rámca prenášaného po médiu.
- Typ rámca – Ethernet II, IEEE 802.3 (IEEE 802.3 - LLC, IEEE 802.3- LLC - SNAP, IEEE 802.3 – Raw).
- Zdrojovú a cieľovú fyzickú (MAC) adresu uzlov, medzi ktorými je rámec prenášaný. Vo výpise jednotlivé bajty rámca usporiadajte po 8, 16 alebo 32 v jednom riadku. Pre prehľadnosť výpisu je vhodné použiť neproporcionálny (monospace) font.

2) 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 jednotlivé vrstvy vykonajte len pre rámce Ethernet II a protokoly rodiny TCP/Ipv4: Na konci výpisu z bodu 1) uveďte IP adresy všetkých vysielajúcich uzlov, ako aj IP adresu uzla, ktorý sumárne odvysielal (bez ohľadu na príjemcu) najväčší počet bajtov (vypíšte). V danom súbore analyzujte zadané komunikácie:
a) HTTP komunikácie
b) HTTPS komunikácie
c) TELNET komunikácie
d) SSH komunikácie
e) FTP riadiace komunikácie
f) FTP dátové komunikácie
g) Všetky TFTP komunikácie
h) Všetky ICMP komunikácie
i) Všetky ARP dvojice (request – reply).

Vo všetkých výpisoch treba uviesť aj IP adresy a pri transportných protokoloch aj porty komunikujúcich uzlov. V prípade výpisu h) uveďte aj typ ICMP správy (pole Type v hlavičke ICMP), napr. Echo request, Echo reply, Time exceeded, a pod.

V prípade výpisu i) uveďte pri ARP-Request 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 rovnakých rámcov ARP-Request, vypíšte všetky.

Ak počet rámcov danej komunikácie je väčší ako 20, vypíšte iba 10 prvých a 10 posledných rámcov. Pri všetkých výpisoch musí byť poradové číslo rámca zhodné s číslom rámca v analyzovanom súbore.

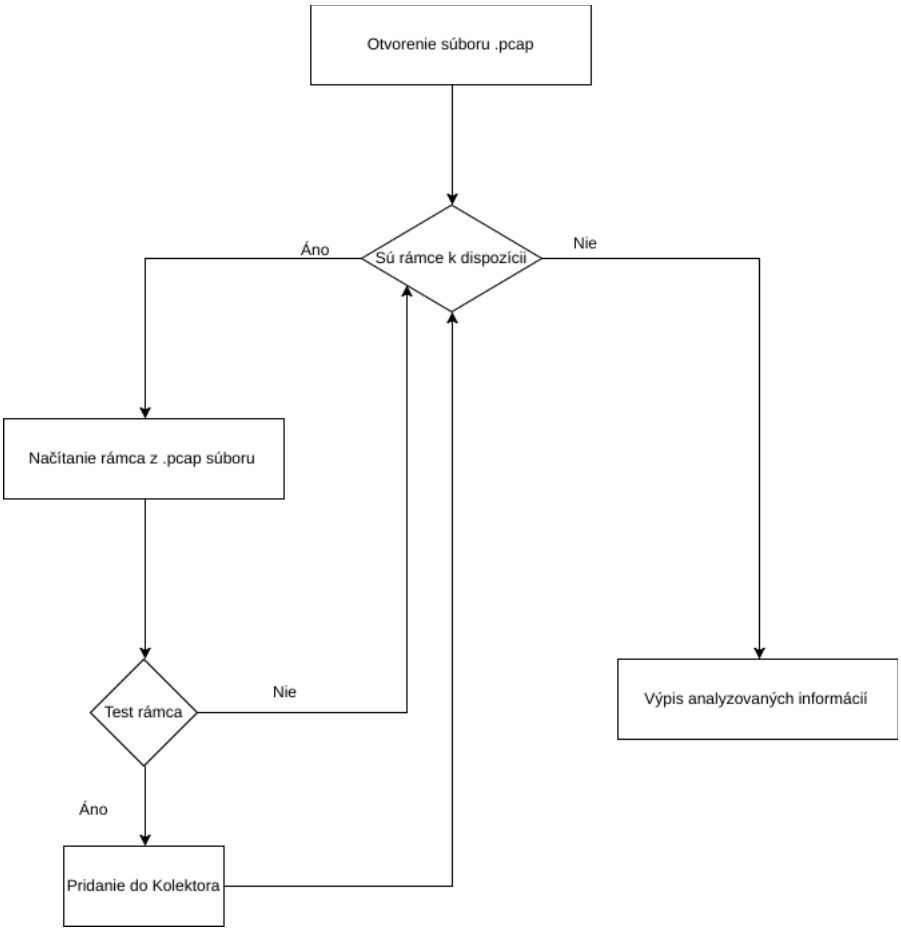
4) Program musí byť organizovaný tak, aby čísla protokolov v rámci Ethernet II a v IP pakete ako aj čísla portov v transportných protokoloch boli programom určené z externého súboru a pre známe protokoly a porty boli uvedené aj ich názvy.

5) V procese analýzy rámcov pri identifikovaní jednotlivých polí rámca ako aj polí hlavičiek vnorených protokolov nie je povolené použiť funkcie poskytované použitým programovacím jazykom. Celý rámec je potrebné spracovať postupne po bajtoch.

6) Program musí byť organizovaný tak, aby bolo možné jednoducho rozširovať jeho funkčnosť o výpis rámcov podľa ďalších požiadaviek na protokoly v bode 3) - pri doimplementovaní jednoduchej funkčnosti na cvičení.

7) Študent musí byť schopný preložiť a spustiť program v miestnosti, v ktorej má cvičenia!

Blokový návrh



Po načítaní ramca sa rámec otestuje. Po prebehnutí každého testu sa vypíšu údaje, ktoré sme získali z analýzy.

Na otvorenie súborov pcap sa používa knižnica libpcap, ktorá poskytuje funkcie pcap_open_offline, pomocou, ktorého môžeme pracovaž so súborom.

Nasledovne sa načítajú porty/protokoly z externých súborov, pomocou funkcie init_nums, všetky porty/protokoly sa načítajú do poľa, pre jednoduchšiu prácu.

```
char frameTypes[0x10000][512];
char etherTypes[0x10000][512];
char tcpProts[0x10000][512];
char udpProts[0x10000][512];
char ip4Prots[0x1000][512];
```

V ďaľšom kroku sa vo while cykle čítajú jednotlivé rámce, smerník poskytnutý funkciou pcap_next použijeme na prístup k rámcu.

```
while ((data = pcap_next(handle, ph))) {
```

Po úspešnom načítaní sa rámec, resp. Smerník na rámec zapúzdrí do štruktúry DATA. Táto štruktúra je potrebná pre správne fungovanie, testovania.

```
typedef struct data {
        int len;
        int num;
        FRAME raw;
        struct data *next;
} DATA;
```

Na testovanie rámcov potrebujeme objekty COLLECTOR, tento objekt si musí programátor sám vyrobyť funkciou create_collector. Tieto objekty pre jednoduchšiu manipuláciu sú súčasťou pola COLLECTOR_SET. Samotný objekt COLLECTOR obsahuje testy, úložiso pre vyhovujúce rámce, spôsob výpisu a pridávania rámco, tieto časti COLLECTOR objektu sú implementované ako funkcie, resp. Smerníky na funkcie, okrem úložiska, ktoré je smerníkom na zvolenú štruktúru dát.

```
typedef struct collector {
        char name[250];
        size_t size;
        void *data;
        DATA *tail;
        bool (*test)(const DATA*);
        void (*add)(struct collector*, DATA*);
        void (*print)(const struct collector*);
        void (*destructor)(struct collector*);
} COLLECTOR;
```

Ak test, patriaci COLLECTOR objektu je úspešní potom COLLECTOR zozbiera daný rámec.

```
for (int i = 0; i < cn-1; ++i)
        if (cs[i]->test(d)) {
                cs[i]->add(cs[i], d);
                break;
        }
```

Po prebehnuťí všetkých testov sa pristupuje k print-ovaniu rámco z COLLECTOR_SET-u.

Posledným krokom je použitie funkcie print_ip_stat, ktorá vypíše info o IP adresách uzlov a určí uzol, ktorý poslal najviac bytov.

Navrhnutý mechanizmus analyzovania protokolov na jednotlivých vrstvách

Analyzovanie rámcov majú za úlohu objekty COLLECTOR, ich funkcia test obsahuje logiku testovania jednotlivých rámcov. Test musí byť poskytnutý pri vytváraní COLLECTOR objektu.

```
collector_set[0] = new_collector("http", test_http, add_list, print_tcp_list, destruct);
```

Počas testovanie sa preiteruje cez test každého COLLECTOR objektu.

Tieto testy určujú, aký protokol/port sa používa na jednotlivých vrstvách. Príklad testu:

```
bool test_arp(const DATA *d)
{
        int prot = d->raw.length[0] << 8 | d->raw.length[1];
        if (prot == get_ether_prot_num("arp"))
                return true;
        return false;
}
```

Navrhnutý postup pri analýze komunikácie so spojením (otváranie, ukončovanie spojení)

Program nenadväcuje žiadne spojenia, všetky vstupy sú načítavané zo súborov .pcap.

Príklad obsahu externých súborov pre určenie protokolov a portov

ipv4_prot.txt

80 http
8080 http
8008 http
20 ftp_data
21 ftp_com
22 ssh
23 telnet
443 https

1. číslo protokolu v desiatkovej sústave
2. meno protokolu/portu, musí byť rovnaké ako v programe, pri inicializovaní objektu COLLECTOR.

Používateľské rozhranie

Program beží v príkazovom riadku použitie:
./packet_analyzer.out <súbor pcap>
Po spustení sa vypíšu všetky rámce a nasledovňe rámce podla protokolov.

Voľba implementačného prostredia

Program bol písaný a kompilovaný na GNU/Linux(Ubuntu) operačnom systéme, dôvodom bola jednoduchosť linkovania a zaobstarania knižníc. Napriek tomu je program skompilovateľný pod ľubovoľným operačných systémom s gcc a libpcap knižnicou.

Program bol kompilovaný kompilátorom GCC ver. 6.3.0 20170406, pomocou týchto prepínačov -Wall -Wextra -Werror -pedantic -pipe -O3 -lpcap -march=native -std=c99, tým pádom, že bol program písaní v štandardnom C jazyku, nemalo by byť problémom prekompilovať program vo Visual Studio alebo CodeBlocks