Zadanie 1

PKS: PcapShark

Matej Rástocký
GitHub projektu: https://github.com/uyohn/pcapshark

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.

Vo výpise jednotlivé bajty rámca usporiadajte po 16 alebo 32 v jednom riadku. Pre prehľadnosť
výpisu je vhodné použiť neproporcionálny (monospace) font.

2) Pre rámce typu Ethernet II a IEEE 802.3 vypíšte vnorený protokol. Študent musí vedieť vysvetliť,
aké informácie sú uvedené v jednotlivých rámcoch Ethernet II, t.j. vnáranie protokolov ako aj
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 prijímajúcich uzlov,
b) IP adresu uzla, ktorý sumárne prijal (bez ohľadu na príjemcu) najväčší počet paketov
a koľko paketov prijal (berte do úvahy iba IPv4 pakety).

IP adresy a počet poslaných paketov sa musia zhodovať s IP adresami vo výpise Wireshark ->

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.

Vo všetkých výpisoch treba uviesť aj IP adresy a pri transportných protokoloch TCP a UDP aj porty komunikujúcich uzlov.
V prípadoch komunikácií so spojením vypíšte iba jednu kompletnú komunikáciu - obsahuje

otvorenie (SYN) a ukončenie (FIN na oboch stranách alebo ukončenie FIN a RST alebo ukončenie
iba s RST) spojenia a aj prvú nekompletnú komunikáciu, ktorá obsahuje iba otvorenie spojenia.
Pri výpisoch vyznačte, ktorá komunikácia je kompletná.
Ak počet rámcov komunikácie niektorého z protokolov z bodu 4 je väčší ako 20, vypíšte iba 10
prvých a 10 posledných rámcov tejto komunikácie. (Pozor: toto sa nevzťahuje na bod 1, program
musí byť schopný vypísať všetky rámce zo súboru podľa bodu 1.) Pri všetkých výpisoch musí byť
poradové číslo rámca zhodné s číslom rámca v analyzovanom súbore.

5) Program musí byť organizovaný tak, aby čísla protokolov v rámci Ethernet II (pole Ethertype),
IEEE 802.3 (polia DSAP a SSAP), v IP pakete (pole Protocol), ako aj čísla portov v transportných
protokoloch boli programom načítané z jedného alebo viacerých externých textových súborov.
Pre známe protokoly a porty (minimálne protokoly v bodoch 1) a 4) budú uvedené aj ich názvy.
Program bude schopný uviesť k rámcu názov vnoreného protokolu po doplnení názvu k číslu protokolu, resp. portu do externého súboru. Za externý súbor sa nepovažuje súbor knižnice, ktorá
je vložená do programu.

6) 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.

7) Program musí byť organizovaný tak, aby bolo možné jednoducho rozširovať jeho funkčnosť
výpisu rámcov pri doimplementovaní jednoduchej funkčnosti na cvičení.

8) 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.

Blokový návrh

Riešenie som implementoval v jedinom .c súbore, ktorý ale čerpá informácie o protokoloch z externých súborov. Proces analyzovania packetov z .pcap súboru prebieha nasledovne:

1. z parametru programu načíta cestu k súboru na analýzu a pokúsi sa ho otvoriť:

```c
int main (int argc, char **argv) {
    // open capture
    if (argv[1] == NULL) {
        printf("supply pcap file path\n");
        return -1;
    }

    pcap_t *handle = open_capfile(argv[1]);

    if (handle == NULL)
        return -1;
```

2. z externých súborov načíta informácie o protokoloch so štruktúr:

```c
typedef struct protocol {
    int n;
    int nstop;
    char *name;
} protocol;
```

```c
    // load subprotocols
    ethernetII_protocols = load_protocols("source/ethernetII_protocols.txt");
    eth802_3_protocols = load_protocols("source/802-3_protocols.txt");
    ipv4_protocols = load_protocols("source/ipv4_protocols.txt");
    tcp_protocols = load_protocols("source/tcp_protocols.txt");
    udp_protocols = load_protocols("source/udp_protocols.txt");
```

3. pripraví štruktúry na IPv4 štatistiky
4. spustí analýzu každého packetu zvlášť:

```c
    // start packet processing loop
    if ( pcap_loop(handle, 0, packetHandler, NULL) < 0 ) {
        printf( "pcap_loop() failed: %s\n", pcap_geterr(handle) );
        return -2;
    }
```

5. na záver vypíše IPv4 štatistiky a uvoľní alokovanú pamäť

Každý packet parsujem do štruktúry, ktorá následne drží všetky relevantné informácie až kým ich nevypíšem. Po výpise funkcia skončí a opakuje sa s ďalším packetom.

```c
typedef struct pkt {
    // meta
    int order;
    int len;
    int real_len;
    char *eth_type_string;

    // pointers to data
    uint8_t *dst_mac;
    uint8_t *src_mac;
    uint16_t *eth_type;
    uint16_t *log_header;
} pkt;
```

```c
void packetHandler (u_char *userData, const struct pcap_pkthdr *pkthdr, const u_char *packet) {

    // parse frame into pkt struct

    pkt current;

    current.order      = frame_no++;
    current.len        = pkthdr->len;
    current.dst_mac    = (uint8_t *)  (packet);
    current.src_mac    = (uint8_t *)  (packet + MAC_SIZE);
    current.eth_type   = (uint16_t *) (packet + 2 * MAC_SIZE);
    current.log_header = (uint16_t *) (packet + 2 * MAC_SIZE + 2);


    // print packet info
    print_pkt(current);
}
```

Analýza protokolov

Na jednotlivých vrstvách analyzujem protokoly nasledovne:

- pre každý packet sa snažím nájsť jeho typ. Ak tento typ nájdem (v externom súbore), vypíšem relevantné informácie. Pre frames typu **Ethernet II** špeciálne kontrolujem, či náhodou neobsahujú **IPv4** packet.
- Ak sa jedná o **IPv4** packet, uložím si jeho informácie do štruktúry s IPv4 štatistikou a skúsim zistiť, aký vnorený protokol tento packet prenáša (opäť pomocou externého súboru).
- Ak prenáša **TCP** alebo **UDP**, zistím na akom porte (zdrojovom a cieľovom) táto komunikácia prebiehala, a podľa well-known ports odhadnem na akom protokole táto komunikácia bežala.

Štruktúra externých súborov

Externé súbory sú písané vo formáte:

M-N#Popis
M#Popis

kde M a N sú čísla v **hexadecimálnom** formáte. Kedže niektoré protokoly operujú na rozsahu portov, je možnosť uviesť M-N (M = počiatočný port, N = koncový port).

```
0000-05DC#IEEE802.3 Length Field
0101-01FF#Experimental
0200#XEROX PUP (see 0A00)
0201#PUP Addr Trans (see 0A01)
0400#Nixdorf
0600#XEROX NS IDP
0660#DLOG
```

Používateľské rozhranie

Pôvodne som plánoval vytvoriť užívateľské rozhranie pomocou knižnice **ncurses**, a strávil som nemalé množstvo času štúdiom tejto knižnice a vytváraním prvej verzie rozhrania. Avšak čas sa krátil a bol som nútený od tohto plánu upustiť. Program nakoniec neobsahuje žiadne UI - pri spúšťaní mu ako argument poskytneme cestu k .pcap súboru a program vypíše všetky packety ktoré sa v ňom nachádzajú, a ku každému z nich vypíše relevantné informácie:

```
Frame 16: 493 bytes on wire (3944 bits), 489 bytes captured (3912 bits)
Ethernet II: Internet Protocol version 4 (IPv4)
ttl: 55, TCP (Transmission Control Protocol): subprotocol not found
Src port: 80, Dst port: 1058
Src ip addr: 128.119.245.12, Dst ip addr: 192.168.1.105
Src mac: 00:06:25:da:af:73, Dst mac: 00:d0:59:a9:3d:68

0000   00 d0 59 a9 3d 68 00 06   25 da af 73 08 00 45 40   01 db 8f 32 40 00 37 06   7b 15 80 77 f5 0c c0 a8        ..Y.=h..%..s..E@...2@.7.{..w....
0020   01 69 00 50 04 22 ac a5   50 d0 65 14 9c 1f 50 18   1b 28 49 75 00 00 3c 68   33 3e 41 6d 65 6e 64 6d        .i.P."..P.e...P..(Iu..<h3>Amendm
0040   65 6e 74 20 49 58 3c 2f   68 33 3e 3c 2f 73 74 72   6f 6e 67 3e 3c 2f 61 3e   0a 0a 3c 70 3e 3c 2f 70        ent.IX</h3></strong></a>..<p></p
0060   3e 3c 70 3e 54 68 65 20   65 6e 75 6d 65 72 61 74   69 6f 6e 20 69 6e 20 74   68 65 20 43 6f 6e 73 74        ><p>The.enumeration.in.the.Const
0080   69 74 75 74 69 6f 6e 2c   20 6f 66 20 63 65 72 74   61 69 6e 20 72 69 67 68   74 73 2c 20 73 68 61 6c        itution,.of.certain.rights,.shal
00a0   6c 0a 6e 6f 74 20 62 65   20 63 6f 6e 73 74 72 75   65 64 20 74 6f 20 64 65   6e 79 20 6f 72 20 64 69        l.not.be.construed.to.deny.or.di
00c0   73 70 61 72 61 67 65 20   6f 74 68 65 72 73 20 72   65 74 61 69 6e 65 64 20   62 79 20 74 68 65 20 70        sparage.others.retained.by.the.p
00e0   65 6f 70 6c 65 2e 0a 0a   3c 2f 70 3e 3c 70 3e 3c   61 20 6e 61 6d 65 3d 22   31 30 22 3e 3c 73 74 72        eople...</p><p><a.name="10"><str
0100   6f 6e 67 3e 3c 68 33 3e   41 6d 65 6e 64 6d 65 6e   74 20 58 3c 2f 68 33 3e   3c 2f 73 74 72 6f 6e 67        ong><h3>Amendment.X</h3></strong
0120   3e 3c 2f 61 3e 0a 0a 3c   70 3e 3c 2f 70 3e 0a 3c   70 3e 54 68 65 20 70 6f   77 65 72 73 20 6e 6f 74        ></a>..<p></p>.<p>The.powers.not
0140   20 64 65 6c 65 67 61 74   65 64 20 74 6f 20 74 68   65 20 55 6e 69 74 65 64   20 53 74 61 74 65 73 20        .delegated.to.the.United.States.
0160   62 79 20 74 68 65 20 43   6f 6e 73 74 69 74 75 74   69 6f 6e 2c 20 6e 6f 72   20 70 72 6f 68 69 62 69        by.the.Constitution,.nor.prohibi
0180   74 65 64 20 0a 20 20 62   79 20 69 74 20 74 6f 20   74 68 65 20 73 74 61 74   65 73 2c 20 61 72 65 20        ted....by.it.to.the.states,.are.
01a0   72 65 73 65 72 76 65 64   20 74 6f 20 74 68 65 20   73 74 61 74 65 73 20 72   65 73 70 65 63 74 69 76        reserved.to.the.states.respectiv
01c0   65 6c 79 2c 20 6f 72 20   74 6f 20 74 68 65 20 70   65 6f 70 6c 65 2e 3c 2f   70 3e 0a 3c 2f 62 6f 64        ely,.or.to.the.people.</p>.</bod
01e0   79 3e 3c 2f 68 74 6d 6c   3e                                                                                y></html>
```

- číslo rámca
- dĺžku rámca
- typ packetu
    - ak ethII tak aj vnorený protokol
        - ak IPv4 tak aj vnorený protokol
        - ak TCP alebo UDP tak aj vnorený protokol
- zdrojový a cieľový port
- zdrojovú a cieľovú IP
- zdrojovú a cieľovú MAC

- “hexdump” rámca

Implementačné prostredie

Riešenie som sa rozhodol implementovať v jazyku C kvôlu jeho dobrej schopnosti pracovať s low-level dátami po bitoch a jeho vysokej efektivite. Program bol písaný na linuxe, preto som mal prístup k header súboru **endian.h** (obsahuje funkcie na vysokoefektívne prevádzanie big/little endian na kódovanie použité na aktuálnom systéme - 10x rýchlejšie ako bitové shifty) a knižnici **libpcap**. Kompiloval som to celé pomocou **gcc** kompilátora použitím flagu **-lpcap** ktorý oznámi kompilátoru aby linkol pcap knižnicu.