Vysoké Učení Technické v Brně
Fakulta informačních technologií

Počítačové komunikace a sítě 2020

Dokumentácia k projektu

Varianta 2: Sniffer

Marek Žiška (xziska03)

Brno, 03.5.2020

Obsah

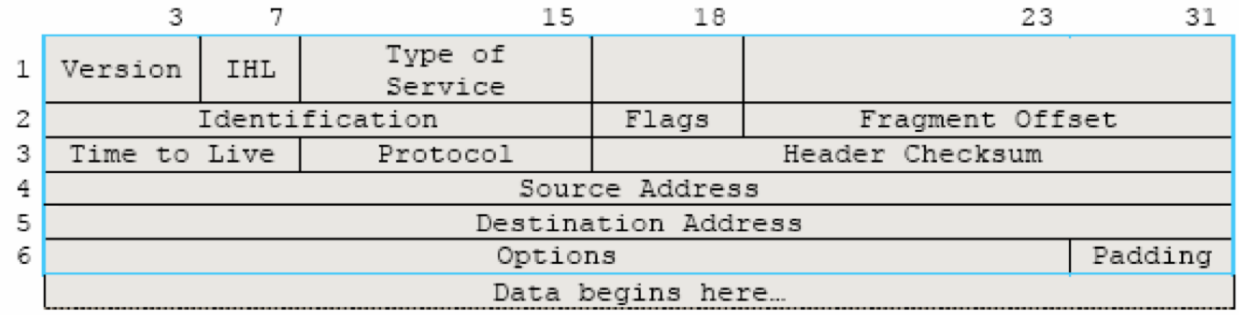
1 Úvod

Cieľom projektu bolo vytvoriť sieťový analyzátor ktorý bude na danom sieťovom rozhraní zachytávať a filtrovať pakety. Filtrovanie je špecifikované na základe argumentov programu. Filtrovať je možné na základe rozhrania(na ktorom sieťový analýzátor pracuje a zachytáva pakety), portu, protokolu paketov. Projekt bol implementovaný pomocou knižnice libpcap [1] v jazyku C. Pri implementácií som sa viedol reálnymi riešeniami ako *Wireshark* alebo *Tcpdump*.

2 Naštudované informácie

2.1 IP (Internet Protocol)

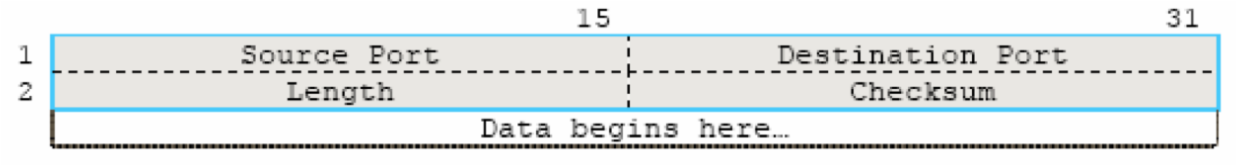
Internetový protokol je protokol sieťovej vrstvy, ktorý sa používa na smerovanie údajov od zdroja k cieľu. Každý datagram obsahuje hlavičku IP nasledovanú protokolom transportnej vrstvy, ako je napríklad tcp alebo udp. Nasledujúca tabuľka zobrazuje formát IP hlavičky.



Zaujímavými dátami pre účely projektu sú *Source Address*, *Destination Address*, *IHL* a *Protocol*. Zdrojové a cieľové adresy(*Source Address* a *Destination Address*) boli použité pre bližšiu špecifikáciu analyzovaného paketu.
Dĺžka IP hlavičky (*IHL*) ukazuje na začiatok dát. Minimálna hodnota pre správnu hlavičku je 5. Hodnoty iné ako 5 je potrebné nastaviť iba v prípade, že hlavička ip obsahuje možnosti (väčšinou sa používajú na smerovanie). Bola použitá pre správnu transformáciu paketu.
Protokol transportnej vrstvy (*Protocol*) označuje, ktorý protokol hornej vrstvy prijíma prichádzajúce pakety po dokončení spracovania IP. Môže byť tcp (6), udp (17), icmp (1) alebo akýkoľvek protokol, ktorý nasleduje po hlavičke ip [2]

2.2 UDP (User Datagram Protocol)

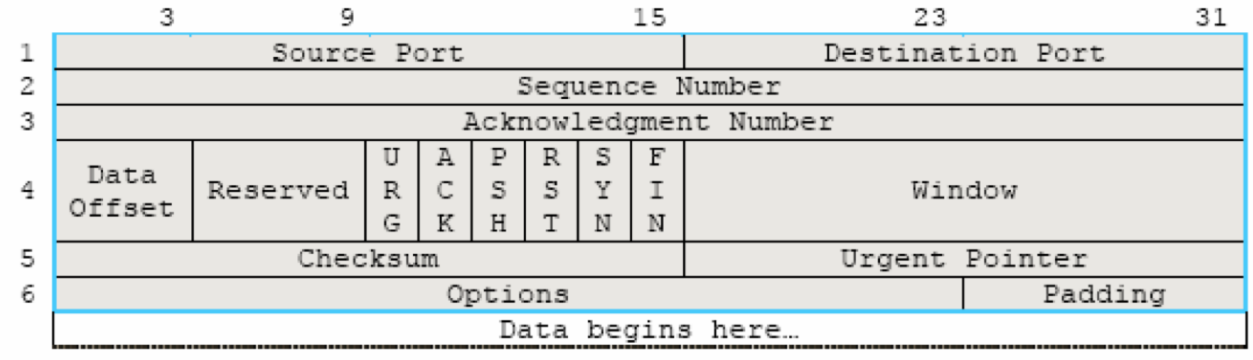
UDP je prenosový protokol pre relácie, ktoré si musia vymieňať údaje. Oba transportné protokoly, UDP a TCP, poskytujú 65535 rôznych štandardných a neštandardných zdrojových a cieľových portov. Cieľový port sa používa na pripojenie k špecifickej službe na tomto porte. Nasledujúca tabuľka zobrazuje formát UDP hlavičky.



Zaujímavými dátami pre účely projektu sú *Source Port* a *Destination Port*. Zdrojový port (*Source Port* predstavuje voliteľné pole, označuje port odosielacieho procesu a možno ho považovať za port, na ktorý by sa malo odpovedať ak nebudú dostupné žiadne ďalšie informácie.
Cieľový port (*Destination Port* na ktorom je možné kontaktovať konkrétny server.
Oba porty boli použité pre bližšiu špecifikáciu analyzovaného paketu.

2.3 TCP (Transmission Control Protocol)

TCP je najčastejšie používaný prenosový protokol, ktorý poskytuje mechanizmy na vytvorenie spoľahlivého spojenia s niektorými základnými overeniami pomocou stavov pripojenia a poradových čísel.



Zaujímavými dátami pre účely projektu sú znova *Source Port* a *Destination Port*, ako pri UDP protokole.
Zdrojom informácií bol článok *Pokročilé protokoly TCP/IP a RAW SOCKET* [3]

3 Implementácia

3.1 Spracovanie argumentov

Pre spracovanie argumentov som využil funkciu *get_opt_long*, vďaka ktorej som bol schopný podporovať nie len konkrétne špecifikovaný formát v zadaní, ale univerzálny formát krátkych jednopomlčkových argumentov ale aj dlhých dvoj(Podpora -i ako aj –i). TODOOOO Ďalej treba dopisať akee error handling situations tu mam + opisat options subor atd atd optind, optarg atd. Inšpiráciu som čerpal z manuálových stránok [4].

```
static struct option long_options[] = {
        {"i",       required_argument,          0,   'i' },
        {"p",       required_argument,          0,   'p' },
        {"port",    required_argument,          0,   'p' },
        {"t",       no_argument,          &tcp_flag,  't' },
        {"tcp",     no_argument,          &tcp_flag,  't' },
        {"u",       no_argument,          &udp_flag,  'u' },
        {"udp",     no_argument,          &udp_flag,  'u' },
        {"n",       required_argument,          0,   'n' },
        {"h",       no_argument,                0,   'h' },
        {"help",    no_argument,                0,   'h' },
        {0,                 0,                  0,    0 }
};
```

3.2 Prípad zadaného rozhrania (-i/–i)

3.2.1 Otvorenie desktriptoru nahrávacej inštancie.

Funkcia *pcap_open_live* sprístupňuje nahrávací "Handle", vďaka ktorému bude možné nahliadať na pakety v sieti. Je to štruktúra *pcap_t*, ktorá je dátového typu "opaque"[5], teda je prístupná len pomocou volaní ďalších funkcí z *libpcap*. Ako argumenty funkcia berie: 1. špecifikované načúvacie rozhranie, 2. veľkosť bufferu "Handle", 3. nastavenie promiskuitného módu, pre účely sniffera nastavené na true. 4. paket buffer timeout, nastavené na 1000ms (tak ako má tcpdump). 5. errbuf, úložisko pre prípadné chybové hlášky.

```
if(!(od = pcap_open_live(interface, BUFSIZ, 1, 1000, errbuf))){
    printf("ERROR -> got %s ", errbuf);
    exit(EXIT_FAILURE);
}
```

Ak nastane chyba, funkcia vráti *NULL* a vloží chybovú hlášku do errbuf, ktorú následne vypisujem.

3.2.2 Kontrola kompatibility s hlavičkou rozhrania

Funkcia *pcap_datalink*.

```
if(link_type != DLT_EN10MB) {
    fprintf(stderr, "Device %s doesn't provide Ethernet headers - not supported\n", interface)
    ;
    exit(EXIT_FAILURE);
}
```

Ak nastane chyba, funkcia vráti *NULL* a vloží chybovú hlášku do errbuf, ktorú následne vypisujem.

3.2.3 Filtrovanie

Funkcia *pcap_compile*. *pcap_setfilter*.

```
1
```

Ak nastane chyba, funkcie vrátia hodnotu -1. V takom prípade ukončujem program s filtrovacou chybou.

3.2.4 Spracovávanie paketov

Funkcia *pcap_loop* načítava *num_of_packets* paketov z načúvacieho rozhrania a vykonáva funkciu *process_packet* ktorá spracováva pakety. Argumenty tejto funkcie sú definované tak ako v manuálových stránkach. [6]

```
1 pcap_loop(od, num_of_packets, process_packet, NULL);
2 pcap_close(od);
3
```

Vo funkcií *process_packet* spracovávam štruktúru *pkthdr*, v ktorej je uložená časová značka paketu. Táto časová značka je uložená v štruktúre *timeval*, ktorú spracovávam pomocou funkcie *localtime* z *time.h*, ktorú následne formátujem na formát zo zadania.

Nasleduje viacero pretypovaní paketu na štruktúry ako IP/UDP/TCP hlavička. Pre správne pretypovanie som použil zdroj z manuálových stránok *libpcap* [7]

Variable	Location (in bytes)
sniff_ethernet	X
sniff_ip	X + SIZE_ETHERNET
sniff_tcp	X + SIZE_ETHERNET + {IP header length}
payload	X + SIZE_ETHERNET + {IP header length} + {TCP header length}

Ako som už spomínal v naštudovanej literatúre, z UDP a TCP hlavičiek spracovávam zdrojové a cieľové porty. S funkciou *ntohs* porty formátujem do bytového poradia hostiteľa. Z IP hlavičky spracovávam zdrojové a cieľové ip adresy, ktoré sa pokúšam preložiť na doménové adresy pomocou *gethostbyaddr*. Ak sa adresu nepodarí preložiť do doménovej adresy, s funkciou *inet_ntoa* ju formátujem z bytového poradia do správneho ip formátu.

3.2.5 Výpis paketov

Pred výpisom samotného paketu, vypisujem spracované porty a adresy do požadovaného formátu. Výpis paketu prebieha vo funkcií *print_packet*. Formát výpisu

je rozdelený do stĺpcov. v Prvom stĺpci je hexadecimálna hodnota počtu vypísaných bytov. V Druhom stĺpci je hexadecimálny výpis bytov paketu a v treťom totožný ale teraz ascii výpis.

3.3 Prípad nezadaného rozhrania (-i/–i)

V tomto prípade využívam funkciu *pcap_findalldevs*, ktorá získa zoznam aktívnych rozhraní, ktorý je následne vypísaný a potom uvoľnený s *pcap_freealdevs*.

4 Testovanie

Testovanie prebiehalo najmä vďaka *Wireshark* a *tcpdump* v ktorom je možné podobné nastavovanie filtrov a zobrazovanie paketov ako v projekte. Teda bolo možné porovnávanie výstupov môjho programu s týmito pre mňa referenčnými programami.

5 Použité zdroje

Odkazy

[1] Libpcap. "Portable C/C++ library for network traffic capture." In: (2019). Official site : https://www.tcpdump.org/pcap.htmland Man page: https://linux.die.net/man/3/pcap.

[2] Internet Assigned Numbers Authority. *List of IP protocols*. (IANA. 2016-06-22). URL: http://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml.

[3] www.tenouk.com. *Advanced TCP/IP and RAW SOCKET*. URL: https://www.tenouk.com/download/pdf/Module43.pdf. (accessed: 01.09.2016).

[4] GNU. *Parsing Long Options with getopt_long*. URL: https://www.gnu.org/software/libc/manual/html_node/Getopt-Long-Options.html#Getopt-Long-Options.

[5] Wikipedia. *Opaque data type*. URL: https://en.wikipedia.org/wiki/Opaque_data_type.

[6] Libpcap. *Process packets from a live capture or savefile*. URL: https://www.tcpdump.org/manpages/pcap_loop.3pcap.html.

[7] Correct typecasting a specification of each portion of the packet. *The Tcpdump Group*. (2010-2020). URL: https://www.tcpdump.org/pcap.html.