VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLÓGIÍ

# Počítačové komunikácie a siete

2. projekt - Sniffer paketov

30. apríla 2020 Terézia Sochová (xsocho14)

# Obsah

# 1 Zadanie

Cieľom projektu bolo vytvoriť packet sniffer v jazyku *c /c++*. Program zachytáva pakety na dostupnom sieťovom rozhraní, ktoré je zadané používateľom ako vstupný argument. Následne pakety filtruje na základe protokolu alebo čísla portu. Výsledkom je výpis dát paketov v hexadecimálnej a ASCII podobe. Prepínač *--udp* alebo *-u* spracúva pakety s UDP protokolom, *--tcp* alebo *-t* pakety s TCP protokolom. Podporované sú aj prepínače *–i* na výber rozhrania, *-p* pre číslo portu a *–n* určuje počet vypísaných paketov.

## 1.1 Packet Sniffer

Packet sniffer je známy aj pod názvami ako packet analyzer alebo network sniffer.

Packet sniffer je program, ktorý zachytáva dáta prechádzajúce cez sieťové rozhrania. Aplikácia zachytáva každý paket a ukladá ho pre budúce spracovanie. Ak je to potrebné dekóduje údaje a zobrazí hodnoty rôznych polí paketu.

Packet sniffer býva často používaný sieťovými alebo systémovými administrátormi, ktorí monitorujú a riešia problémy so sieťovou prevádzkou. Okrem toho je často využívaná schopnosť programu zachytiť prichádzajúci a odchádzajúci prenos vrátane hesiel užívateľských mien alebo iných citlivých materiálov, čo spôsobuje bezpečnostnú hrozbu.[1][2]

# 2 Implementácia

## 2.1 Spracovanie argumentov

Pri robení projektu som začala získaním argumentov z príkazového riadku. Implementácia tejto časti projektu sa nachádza v súbore *ipk-parser.cpp*. Kedže aplikácia podporuje krátke(*-i, -n, -p, -u, -t* ), ale rovnako aj dlhé možnosti argumentov (*--udp* a *--tcp*) pre spracovanie argumentov a ich hodnôt som používala funkciu *getopt_long()* [3].(Postupovala som podľa v zdroji uvedeného manuálu). Výhodou využitia tejto funkcie bola aj možnosť nastavenia, že dlhé argumenty neočakávajú hodnotu, a teda hodnotu premennej *has_arg* som nastavila na *no_argument*. Ak bol vstupný argument zadaný, tak sa pri ňom nastavil príznak výskytu a každý argument mohol byť v príkazovom riadku nastavený len jedenkrát. Týmto spôsobom je kontrplovaný počet zadaných argumentov. Hodnoty a príznaky argumentov sú pomocou *Packet_sniffer()* odovzdané pre ďalšie spracovanie.

## 2.2 Aktívne rozhrania

Okrem spracovania argumentov, program podporuje aj výpis aktívnych rozhraní, a to v prípadoch, že nie je zadaný žiadny argument, nebol zadaný prepínač *-i* alebo mu chýba hodnota. Pre nájdenie rozhraní bola použitá funkcia *pcap_findalldevs()* [4]. (Informácie do implementácie som čerpala z priloženého manuálu), ktorá vracia pole sieťových rozhraní. Následne u jednotlivých rozhraní kontrolujem, či sú pri nich nastavené príznaky *PCAP_IF_UP* a *PCAP_IF_RUNNING*.

## 2.3 Spracovanie paketov

Časť projektu zaoberajúca sa zachytávaním a spracovaním paketov sa nachádza v súbore *ipk-sniffer.cpp*. Na začiatku prebehne kontrola nastavených príznakov a overenie platnosti zadaného rozhrania. Potom je volaná funkcia *pcap_open_live()* [5][6]. Pre prácu s *pcap* funkciami som používala hlavne tieto dva manuály.), ktorá otvára rozhranie pre zachytávanie paketov. Nastavením parametra *promiscuous mode* na *true* sú zachytávané aj pakety určené pre iné zariadenia. Ďalšou funkciou použitou z knižnice *pcap* je *pcap_loop()* [5][6], ktorá sa stará o samotné zachytávanie paketov. Každý paket je posielaný do funkcie na filtrovanie paketov.

Pri filtrovaní paketov používam štruktúry pre hlavičky vrstiev paketu definované v knižnici *netinet* [7](pre túto časť implementácie som sa inšpirovala manuálom a príkladmi z daného zdroja), ako prvá je to *struct ip* [8] pre naplnenie ip hlavičky a *struct ether_header* pre jej správnu veľkosť. Z nej získavam informácie o zdrojovej a cieľovej ip adrese a o protokole. Následne porovnávam hodnotu protokolu s číslo *6* [9] (TCP protokol) a s číslom *17* [9], čo predstavuje UDP protokol. Ak je paket jednou z týchto dvoch možností posúva sa na ďalšie spracovanie, ak nie tak je zahodený a prichádza na rad nasledujúci paket.

V dvoch rozdielnych funkciách sa napĺňajú štruktúry: *tcphdr* [10] a *udphdr* [11] pre *tcp* a *udp* hlavičky. Z nich sú získavané hodnoty portov. Ak je vstupný argument *-p* nastavený, tak je jeho hodnota porovnávaná s hodnotami cieľového a zdrojového portu. Pri zhode je paket posielaný do funkcie vypisovania dát inak sa prechádza na spracovanie dalšieho paketu. Program je ukončený ak je dosiahnuta hodnota *-n* argumentu.

## 2.4 Čas zachytenia paketu

Hodnota času je získaná z *pcap* hlavičkovej štruktúry daného paketu, konkrétne z *tv_usec* a *tv_sec*. Hodnota v premenných udáva čas, ktorý uplynul od začiatku roku 1970. Zrozumiteľnejší čas pre ľudský rozum je získaný sú funkciami z knižnice *time.h*. [12](Implemetáciu tejto časti som robila podľa priloženého manuálu).

## 2.5 Výpis dát

Funkcia príma paket s dátami a jeho veľkosť. Kým nie je dosiahnutá veľkosť paketu, tak sa vypíše *hexadecimálna* hodnota znaku. Pre jednoduchšie čítanie je po 8 znakoch vypísaná dvojitá medzera. Na konci riadku sa nachádza prepis *hexadecimálnych* hodnôt na *ASCII* znaky. Každý riadok obsahuje aj informáciu o už vypísaných bajtoch začínajúc hodnotou *0x0000*. Pred každým vypísaných paketom je uvedený riadok s informáciami o čase zachytenia paketu portoch a adresách. Ak má adresa pridelené doménové meno, tak je vypísané miesto nej. Toto má na starosti *gethostbyaddr()* [13] funkcia.

# 3 Testovanie

## 3.1 Spôsob spustenia

Ku spusteniu programu je nutné zadať argument *–i [rozhranie]*. Ostatné argumenty sú nepovinné. Ak nie je zadan7 argument *–n [číslo]* , ktorý určuje počet zobrazených paketov, hodnota je implicitne nastavená na 1. Dôležité je spomenúť, že ak nie je vybraný filter protokolu sú zachytávané aj upd aj tcp pakety.

Príklady:

*./ipk-sniffer -i eth0*

*./ipk-sniffer -i eth0 -n 10*

*./ipk-sniffer -i eth0 -t -p 80*

*./ipk-sniffer -i eth0 --udp --tcp -p 80 -n 12*

## 3.2 Testovanie programu

Testovanie programu prebehlo na dvoch linuxových distribúciach: *Debian* a *Ubuntu,* na ktorých je projekt preložiteľný. Program som spúšťala pre tri rôzne rozhrania : *eth0, lo* a *enp0s3.*

Na generovanie prichádzajúcich a odchádzajúcich dát som spúšťala rôzne stránky vo webovom prehliadači.

Správne hodnoty a správny tok programu som kontrolovala pomocou vypisovania premenných a ich príznakov. Vypisovala som najmä hodnotu premennej *ip_p* (protokol) nachádzajúcich sa v štruktúre *ip* a porovnávala s očakávanými hodnotami. Rovnakým spôsobom som testovala aj použitie argumentu *-p*. Kontrolovala som či sa porty pri zachytených paketoch rovnajú *53* (UDP) alebo *443* (TCP).

Pri testovaní výpisu, som svoj výpis kontrolovala s výstupom v softvéri *Wireshark* a snažila som sa k nemu, čo najviac priblížiť. Na overenie aktívnych rozhraní som použila príkaz *tcpdump* s prepínačom *-D*.

## 3.3 Wireshark

*Wireshark* [14] je jeden z najrozšírenejších a svetovo najpoužívanejších softvérov na zachytávanie a analýzu paketov. Preto som sa ho aj ja rozhodla použiť. Na začiatku som si vybrala rozhranie (*Wi-Fi*) a sledovala som komunikáciu. Použila som aj filtrovací riadok na filtrovanie pre mňa dôležitých *TCP* a *UDP* paketov. Sledovala som výpisy dát, ktoré mi dali lepšiu predstavu o tom, ako má vyzerať výstup môjho programu, a podľa akých parametrov mám pakety zachytávať [15].

# 4 Obmedzenia

Pri záverečnom testovaní som neodhalila žiadne závažné obmedzenia základnej časti projektu.

# 5 Bibliografia


[1] PALLOVI, Asrodia a Patel HEMLATA. Network Traffic Analysis Using Packet Sniffer [online]. [cit. 2020-05-01]. ISSN 2248-9622. Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.417.840&rep=rep1&type=pdf.

[2] Packet analyzer [online]. [cit. 2020-05-01]. Dostupné z: https://en.wikipedia.org/wiki/Packet_analyzer

[3] Getopt_long(3) - Linux man page [online]. [cit. 2020-05-01]. Dostupné z: https://linux.die.net/man/3/getopt_long

[4] Manpage of PCAP_FINDALLDEVS [online]. 2018 [cit. 2020-05-01]. Dostupné z: https://www.tcpdump.org/manpages/pcap_findalldevs.3pcap.html

[5] CARSTENS, Tim. Manpage of PCAP_FINDALLDEVS [online]. 2018 [cit. 2020-05-01]. Dostupné z: https://www.tcpdump.org/pcap.html

[6] JACOBSON, Van, Craig LERES a Steven MCCANNE. OpenBSD manual page serve [online]. 2019 [cit. 2020-05-01]. Dostupné z: https://man.openbsd.org/pcap_open_offline.3

[7] Advanced TCP/IP and RAW SOCKET [online]. [cit. 2020-05-01]. Dostupné z: https://www.tenouk.com/download/pdf/Module43.pdf

[8] Source to netinet/ip.h [online]. [cit. 2020-05-01]. Dostupné z: https://unix.superglobalmegacorp.com/Net2/newsrc/netinet/ip.h.html

[9] Protocol Numbers [online]. [cit. 2020-05-01]. Dostupné z: https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml?fbclid=IwAR2OIASPT801KSYgDE83uSecyk05hG9ahTCSgkB-vQNymNSydcelHWG4OUE

[10] Source to netinet/tcp.h [online]. [cit. 2020-05-01]. Dostupné z: https://unix.superglobalmegacorp.com/BSD4.4/newsrc/netinet/tcp.h.html

[11] Source to netinet/udp.h [online]. [cit. 2020-05-01]. Dostupné z: https://unix.superglobalmegacorp.com/Net2/newsrc/netinet/udp.h.html

[12] <ctime> (time.h) [online]. [cit. 2020-05-01]. Dostupné z: http://www.cplusplus.com/reference/ctime/

[13] BSD Library Functions Manual [online]. [cit. 2020-05-01]. Dostupné z: http://www.manpagez.com/man/3/gethostbyaddr/

[14] Wireshark [online]. [cit. 2020-05-01]. Dostupné z: https://www.wireshark.org/?fbclid=IwAR2QEySWZid7PCn5aq3SL_U5oyOeav6eYoLJRi_-SHR8S3jy3gng086HCtc

[15] HOFFMAN, Chris. How to Use Wireshark to Capture, Filter and Inspect Packets [online]. 2017 [cit. 2020-05-01]. Dostupné z: https://www.howtogeek.com/104278/how-to-use-wireshark-to-capture-filter-and-inspect-packets/?fbclid=IwAR2bQSom4PtcNUZVV_KF34HbZX-y6DcxmwKuIUymyDrgzkx8Sd79rh-QF_g