VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta informačních technologií

POČÍTAČOVÉ KOMUNIKACE A SÍTĚ

2021

Projekt 2 – Varianta ZETA: Sniffer paketů

Dokumentácia

Peter Rúček (xrucek00) 19. apríl 2021

Obsah

1. Úvod

Cieľom projektu bolo vytvoriť sieťový analyzátor- paket sniffer, ktorý bude na určitom sieťovom rozhraní zachytávať a filtrovať pakety. Program na štandardný výstup vypisuje zachytené pakety a v prípade úspechu vracia chybový kód 0. V prípade neúspechu vypíše chybu na štandardný chybový výstup a vráti chybový kód 1 pokiaľ nastala chyba pri práci s knižnicou `pcap.h` a chybový kód 2 pokiaľ boli zadané nesprávne argumenty programu.

2. Implementácia

2.1 Základné údaje

Program je napísaný v jazyku C++17, no je to skôr jazyk C, s drobnými prvkami C++. Vybral som si tento jazyk, lebo som si myslel, že implementácia bude jednoduchšia, ale nestalo sa tak. Samotný kód je rozdelený do dvoch modulov- `print.cpp/hpp` kde sú funkcie potrebné pre výpis paketu na `stdout` a ipk-`sniffer.cpp/hpp`, ktorý je hlavným programom, parsujú sa tu argumenty programu a pomocou knižnice `pcap.h`, ktorá bola použitá na zachytávanie paketov, sa vykonáva logika programu.

2.2 Chod programu

V tejto časti dokumentácie sa bližšie popisuje ako program funguje. Sniffer sa dá rozdeliť do troch logických častí:

1. Spracovanie argumentov
2. Zachytenie paketu
3. Výpis paketu

Na začiatku vykonávania programu sa skontrolujú argumenty programu. Musia byť zhodné s:

./ipk-sniffer [-i rozhraní | --interface rozhraní] {-p port} {[--tcp|-t] [--udp|-u] [--arp] [--icmp] } {-n num}

Pri nezadaní argumentu `-i/--interface` alebo pri zadaní iných argumentov ako tu uvedených dôjde k chybe 2 a vypíše sa na `stderr`. Pri zadaní iba argumentu `-i` bez rozhrania sa vypíšu všetky dostupné rozhrania. Ostatné argumenty slúžia na filtráciu paketov. Pri nezadaní argumentu `-n` sa vypíše iba jeden paket. Funkcia `arg_parse()` vykonáva všetky tieto kontroly a okrem toho naplňuje dáta globálnych premenných, z ktorých každá reprezentuje jeden prepínač/argument.

Zachytávanie paketu prebieha vo funkcií `main()`, kde sa najskôr nachystá zariadenie pre zachytávanie paketov, a potom až funkcia `pcap_loop()` zachytáva pakety. Táto funckia sa volá v cykle aby sa dalo jednoducho odchytiť nami zadaný počet paketov. Ako jeden z jej parametrov je adresa funkcie na spracovanie paketu. Pre nás je to funkcia `process_packet()`, kde sa odohráva všetko dôležité ohľadom práce s paketom. Jedným z jej parametrom je `buffer` kde sú uložené všetky prijaté dáta. Postupne z `buffera` dostávame viacej informácií, pomocou pretypovania na príslušnú hlavičku protokolu.

Keď už máme všetky potrebné údaje tak paket vypíšeme vo formáte:

IPv4/ IPv6 *Protokol*

čas (RFC3339) IP : port > *IP : port*, length *dĺžka* bytes

offset_vypísaných_bajtov: výpis_bajtov_hexa výpis_bajtov_ASCII

Pre prehľadnosť pred každý paket vypíšeme či je IPv4 alebo IPv6 alebo ARP a jeho protokol (pri ARP nie). Pri protokole ARP a ICMP/ ICMPv6 vypisujeme iba *IP* > *IP* bez portov a pri TCP a UDP už aj s portami. Na výpis času nám slúži `print_time_rfc3339()`, IP adresy a porty si musíme vyčítať z hlavičky a na výpis dát v požadovanom formáte voláme fukciu `print_data()`.

2.3 Odlišnosti

V tejto sekcií sa zmienime o vlastných riešeniach problémov, ktoré neboli presne popísané v zadaní alebo sme si sami doplnili:

- pred každý paket vypíšeme typ paketu
- keď užívateľ nezadá žiaden prepínač z množiny (--tcp/ --udp/ --arp / --icmp) zachytávajú sa všetky pakety, nie len tieto 4
- u ICMP a ARP sa vypisujú iba IP adresy bez portov

3. Testovanie

Práca s počítačovými sieťami je náročná, nie je nikdy zaručený rovnaký výsledok, preto aj testovanie je výzvou. Najskôr sme pracovali iba s UDP a TCP paketmi, tým že sme refreshovali webové stránky webového prehliadača. Na protokoly ICMP a ARP alebo IPv6 sme spúšťali rôzne iné programy ako ping, curl, arpsend... Výstupy nášho sniffera sme porovnávali s programom Wireshark, ktorý je na túto prácu ideálny. Po dokončení projektu, náš sniffer pracuje rovnako ako Wireshark, vypíše dáta a IP adresy rovnako, akurát náš sieťový analyzátor je omnoho pomalší.

4. Zhrnutie

Tohoto projektu som sa veľmi obával, pretože som danej problematike veľmi nerozumel. Teraz však môžem povedať, že ma práca na projekte veľmi bavila (tie siete nie sú také hrozné ako sa na prvý pohľad zdá) a veľmi veľa som sa naučil (ako protokoly fungujú, čo to je paket, ako ho odchytiť, čo to je hlavička…). Náš program funguje aj pre IPv4 aj pre IPv6 aj ARP, zachytáva všetky druhy paketov a vypíše paket na `stdout`. Jediným problémom je, že program je dosť pomalý, dlho mu trvá dokým vypíše dané pakety. Zachytí ich rýchlo, rovnako ako Wireshark, ale potom treba chvíľu počkať až ich vypíše.

5. Bibliografia

Zdroje, z ktorých som čerpal pri vypracovávaní projektu :

Wikipedia: List of IP protocol numbers. [online], rev. 24. marec 2021, [vid. 19. apríl 2021]. Dostupné z : https://en.wikipedia.org/wiki/List_of_IP_protocol_numbers

Wikipedia: IPv4. [online], rev. 18. apríl 2020 [vid. 19. apríl 2021]. Dostupné z : https://sk.wikipedia.org/wiki/IPv4

Wikipedia: IPv6. [online], rev. 13. február 2019 [vid. 19. apríl 2021]. Dostupné z : https://sk.wikipedia.org/wiki/IPv6

PLUMMER, D. : An Ethernet Address Resolution Protocol. [online], rev. november 1982 [vid. 19. apríl 2021] Dostupné z : https://tools.ietf.org/html/rfc826

CARSTENS, T.: Programming with pcap. [online], rev. 2002 [vid. 19. apríl 2021] Dostupné z : https://www.tcpdump.org/pcap.html

Silver Moon : How to code a Packet Sniffer in C with Libpcap on Linux. [online], rev. 31. júl 2020 [vid. 19. apríl 2021] Dostupné z : https://www.binarytides.com/packet-sniffer-code-c-libpcap-linux-sockets/

Oracle: inet_ntoa(3SOCKET). [online], rev. júl 2014 [vid. 19. apríl 2021] Dostupné z : https://docs.oracle.com/cd/E36784_01/html/E36875/inet-ntoa-3socket.html

yagu99 : Identify ARP and Broadcast Packets with packet sniffer. [online], rev. 22. september 2012 [vid. 19. apríl 2021] Dostupné z : https://www.codeproject.com/Questions/463912/Identify-ARP-and-Broadcast-Packets-with-packet-sni

jedisct1 : rfc3339.c . [online], rev. 20. december 2016 [vid. 19. apríl 2021] Dostupné z : https://gist.github.com/jedisct1/b7812ae9b4850e0053a21c922ed3e9dc/revisions

ntohs(3) - Linux man page. [online] [vid. 19. apríl 2021] Dostupné z : https://linux.die.net/man/3/ntohs

pcap(3) - Linux man page. [online] [vid. 19. apríl 2021] Dostupné z : https://linux.die.net/man/3/pcap

getopt_long(3) - Linux man page. [online] [vid. 19. apríl 2021] Dostupné z : https://linux.die.net/man/3/getopt_long