Sniffer sieťových paketov

Počítačové komunikácie a siete

2019/2020

3. mája 2020

Autor: Martin Fekete (xfeket00@stud.fit.vutbr.cz)

Obsah

1 Zadanie

Zadaním projektu bolo implementovať sieťový analyzátor schopný zachytávať a filtrovať pakety na určitom sieťovom rozhraní. Aplikácia by mala byť plne prenositeľná na operačných systémoch postavených na Unixe a musí sa dať preložiť pomocou príkazu `make`.

2 Problematika

Paket je blok prenášajúci dáta v počítačovej sieti a je alternatívou k prenosu dát bit po bite. Okrem používateľských dát obsahuje aj informácie potrebné na smerovanie paketu v sieti - napríklad zdrojový a koncový port, IP adresu zdrojového a koncového zariadenia a podobne [2].

Sniffer paketov potom slúži na odchytávanie týchto paketov v sieti, ich prípadný parsing a analýzu [3]. Existuje veľa rôznych nástrojov na sniffing paketov, medzi najznámejšie patria open-source nástroje Tcpdump alebo Wireshark, ktorý je popísaný nižšie.

3 Implemenácia

Program je implementovaný v jazyku C za použitia systémových knižníc a knižníc pre prácu so sieťovými nástrojmi.

3.1 Základné jadro programu

Na úvod sú spracované argumenty pomocou funkcie `getopt_long` a taktiež je skontrolovaná ich validita - ak bol teda zadaný nepovinný prepínač `-n` a nebol v jeho argumente špecifikovný počet požadovaných paketov, program je ukončený s chybovou hláškou. Nasleduje kontrola, či bolo zadané rozhranie a ak nie, je vypísaná programová nápoveda spolu so zoznamom rozhraní. Ďalej je nastavený filter siete podľa hodnoty nepovinných prepínačov, implicitne je nastavovaný filter `tcp or udp`. Nasleduje štvorica funkcií z knižnice `pcap` [4]:

- `pcap_open_live` : funkcia otvorí rozhranie špecifikované argumentom `-i rozhranie` a uloží jeho tzv. *handler* potrebný na ďalšie spracovávanie do pomocnej premennej. V prípade, že zariadenie neexistuje je program ukončený s chybou 1 a odpovedajúcou chybovou hláškou. Pomocou argumentu funkcie je nastavený takzvaný promiskuidný režim, ktorý je vhodný na sniffing paketov, pretože nie je zachytávaná iba kominikácia určená pre zariadenie, na ktorom je program spustený. Buffer timeout je nastavený na 100 ms (rovnaký hodnotu používa aj nástroj Wireshark), takže program čaká maximálne 100 ms, kým začne spracovávať zachytené pakety.

- `pcap_compile` a `pcap_setfilter` : dvojica funkcií ktorá je potrebná na filtrovanie paketov. Implicitne sú odchytávané pakety typu TCP a UDP, pomocou argumentov programu však je možné ďalej špecifikovať port, z ktorého budú pakety odchytávané, poprípade špecifikovať, že budú odchytávané buď pakety iba typu TCP, alebo iba typu UDP.

- `pcap_loop` : funkcia zachytáva prvých `n` paketov (číslo `n` stanoví buď používateľ prepínačom, alebo je implicitne nastavené na 1), ktoré sú ďalej poslané funkcii `process_packet`, kde sú sparsované a vypísané na štandardný výstup.

Funkcia `process_packet` najprv získa a vypíše čas paketu, následne je zistená IP verzia paketu a paket je spracovávaný na základe typu jeho transportnej vrstvy (TCP alebo UDP). Potom sú vypísané informácie o zdrojovej a cieľovej IP adrese a porte. Preklad IP adresy na FQDN neprebieha z dôvodu, že pri preklade sú vytvárané ďalšie pakety a vzniká tak teoreticky nekonečný cyklus, ktorý je ukončený buď používateľom, alebo zachytením stanoveného počtu paketov zadaných používateľom.

Výpis paketov na štandardný výstup zabezpečuje funkcia `print_data`, kde každý nový riadok pri výpise začína informáciou o počte dosiaľ vypísaných bajtov, nasleduje výpis 16 bajtov v hexadecimálnej podobe a ich ASCII reprezentáciou. Pre každý paket platí, že je pri výpise oddeľovaná hlavička a telo novým riadkom.

3.2 Implementačné zaujímavosti

Vzhľadom k tomu, že program má byť prenositeľný na unixových systémoch, bolo nutné vyriešiť problematiku pomenovania atribútov štruktúr definujúcich napríklad ethernet a IP hlavičku. Pri linuxových systémoch majú tieto atribúty odlišné názvy ako na iných unixových systémoch, takže je tento problém vyriešený jednoduchým makrom, ktoré zistí, na akom OS je program spustený a podľa toho nastaví makro hodnoty ukazateľov do potrebných štruktúr. Napríklad v prípade nelinuxového systému je makro zdrojového portu TCP paketu nastavené na hodnotu `tcph->th_sport`, v prípade linuxového systému je táto hodnota `tcph->source`.

Okrem podpory paketov s ethernetovou hlavičkou vrstvy dátového spoja podporuje aplikácia takisto pakety s tzv. *linux-cooked* hlavičkou, ktorá je špecifická pre rozhranie *any* na linuxových systémoch [1]. Kvôli tejto podpore je v programe vytvorená jediná globálna premenná, do ktorej je pomocou funkcie `pcap_datalink` uložená konštanta s hodnotou veľkosti danej hlavičky.

4 Preklad a spustenie

4.1 Preklad

Preklad programu prebieha podobne ako preklad štandardných Unixových programov pomocou príkazu make a je ho možné rozdeliť do nasledujúcich 2 krokov:

1. Pomocou príkazového riadku je nutné otvoriť priečinok, v ktorom je uložený zdrojový súbor `ipk-sniffer.c` a príslušný `Makefile`.

2. Spustenie príkazu `make` vytvorí spustieľný súbor `ipk-sniffer`.

V prípade nedostupnosti súboru `Makefile` je v bode 2 nutné namiesto príkazu `make` spustiť príkaz `gcc ipk-sniffer.c -lpcap -o ipk-sniffer`.

4.2 Spustenie

Spustenie programu vyzerá nasledovne:

```
./ipk-sniffer -i rozhranie [-p port] [--tcp|-t] [--udp|-u] [-n num] [--help|-h]
```

kde:

- `-i rozhranie` je rozhranie, na ktorom budú pakety odchytávané (napr. eth0).
- `-p port` je obmedzenie na port, na ktorom budú pakety odchytávané. Pri neuvedení tohto prepínača sú brané do úvahy všetky porty.
- `--tcp` alebo `-t` je obmedzenie iba na pakety typu TCP.
- `--udp` alebo `-u` je obmedzenie iba na pakety typu UDP.
- Pri neuvedení ani `--tcp|-t` ani `--udp|-u` alebo pri uvedení oboch prepínačov sú brané do úvahy pakety typu TCP aj UDP.
- `-n num` definuje počet paketov, ktorý bude odchytávaný.
- Pri spustení programu s prepínačom `--help` alebo `-h` je vypísaná nápoveda, spolu so zoznamom dostupných rozhraní na štandardný výstup, pričom sú ostatné prepínače ignorované.

Prepínač `-i rozhranie` je povinný, ostatné prepínače sú nepovinné a pri ich neuvedení je do úvahy braná implicitná hodnota. V prípade neuvedenia prepínaču `-i rozhranie` je vypísaná stručná nápoveda, ako program spustiť spolu s výpisom všetkých dostupných rozhraní (rovnako ako pri prepínači `--help`).

4.3 Príklad spustenia

V nasledujúcom prípade budú odchytávané prvé TCP pakety na rozhradní en0 (ak je na zariadení dostupné) a porte 443.

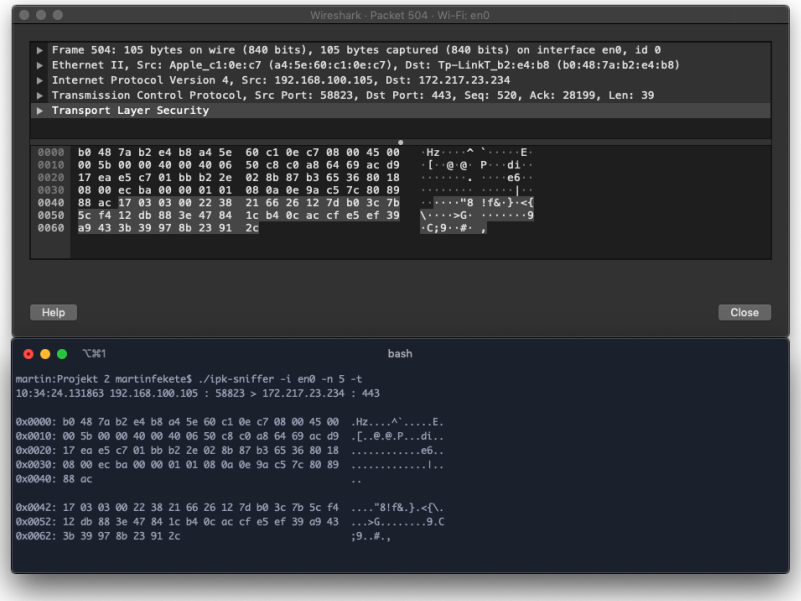
```
./ipk-sniffer -i en0 -p 443 --tcp -n 2
```

Príklad výstupu programu pri spustení s týmito parametrami:

```
10:38:30.187354 192.168.100.105 : 58843 > 35.167.182.111 : 443

0x0000: b0 48 7a b2 e4 b8 a4 5e 60 c1 0e c7 08 00 45 00  .Hz....^‘.....E.
0x0010: 00 28 1b e1 00 00 40 06 5f c7 c0 a8 64 69 23 a7  .(....@._...di#.
0x0020: b6 6f e5 db 01 bb a3 1e e8 82 b7 ce 87 b0 50 10  .o............P.
0x0030: 08 00 f5 f4 00 00                                ......


10:38:30.378720 35.167.182.111 : 443 > 192.168.100.105 : 58843

0x0000: a4 5e 60 c1 0e c7 b0 48 7a b2 e4 b8 08 00 45 00  .^‘....Hz.....E.
0x0010: 00 34 03 de 40 00 dc 06 9b bd 23 a7 b6 6f c0 a8  .4..@.....#..o..
0x0020: 64 69 01 bb e5 db b7 ce 87 b0 a3 1e e8 83 80 10  di..............
0x0030: 00 79 e5 e0 00 00 01 01 08 0a ba 6e b8 1e 0e 9e  .y.........n....
0x0040: 5d 57                                            ]W
```

5 Testovanie

Funkčnosť programu bola testovaná na operačných systémoch MacOS 10.14 a Ubuntu 18. Výstup programu bol porovnávaný s výstupom populárneho protokolového analyzéru Wireshark. Na obrázku nižšie je takéto porovnanie ukázané.



V prvej fáze testovania boli porovnávané samotné počty vypísaných bajtov spolu s ich ASCII reprezentáciou. Následne bolo odchytávaných viacero paketov a kontrolované, či sedí ako ich obsah, tak aj poradie zachytenia.

Testovanie paketov typu IPv4 bolo viacmenej bezproblémové, IPv6 pakety boli kvôli obmedzeným podmienkam testované menej rozsiahlo a boli testované pomocou programu `curl`.

Literatúra

[1] Link-layer header types. https://www.tcpdump.org/linktypes.html.

[2] Network packet. https://en.wikipedia.org/wiki/Network_packet.

[3] Packet analyzer. https://en.wikipedia.org/wiki/Packet_analyzer.

[4] G. Malkin. Programming with pcap. https://www.tcpdump.org/pcap.html.