

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií



Počítačová komunikácia a siete
2019/2020

Projekt č.2 – Packet Sniffer

Matej Otčenáš, xotcen01
20.4.2020

Obsah

1. Zadanie.....	3
2. Implementácia.....	3
3. Testovanie.....	5
4. Zdroje.....	9

1. Zadanie

Cieľom projektu bolo navrhnúť a implementovať sieťový analyzátor, ktorý bude schopný na určitom sieťovom rozhraní zachytávať a filtrovať TCP/UDP packety.

2. Implementácia

Projekt bol implementovaný v jazyku C++11, vďaka ktorému bolo možné použiť objektové programovanie a jednoduchšiu manipuláciu s reťazcami pomocou STL knižníc. Napriek tomu značná časť programu používa základné konštrukcie jazyka C, z dôvodu využívania knižníc pre prácu s packetmi, ktoré sú implementované taktiež v jazyku C.

I. *Spracovanie argumentov*

Vstupné argumenty programu sú spracované pomocou funkcie *getopt_long()*, ktorá umožňuje spracovať tzv. *long/short options*, ktoré program musí podporovať. Následne za pomoci konštrukcie *switch* sú jednotlivé vstupné parametre ukladané do premenných vo vytvorenom mennom priestore (*namespace*) aby sa zamedzilo používaniu globálnych premenných. Tieto premenné obsahujú napríklad číslo portu, počet packetov na výpis či rozhodovanie pre výpis TCP/UDP packetov. Tieto premenné sú ďalej v programe testované pre svoju korektnosť (napríklad číselný rozsah portu atd.).

II. *Nastavenie rozhrania*

Rozhranie, na ktorom bude packet sniffer odchytať packety musí byť explicitne zadané pomocou vstupného argumentu.

V prípade, že rozhranie špecifikované nie je, program vypíše všetky dostupné rozhrania a úspešne skončí. V tejto situácii sa nenastaví žiadne rozhranie.

Ak je programu zadané korektné aktívne rozhranie (napr. *wlp5s0*, *enp3s0*, ...), tak sa dané rozhranie nastaví pomocou funkcií knižnice *<pcap.h>* (*packet capture*). Pri konkrétne špecifikovanom porte sa taktiež použijú funkcie *pcap_compile()* a *pcap_setfilter()*, ktoré nastaví k danému rozhraniu aj

špecifikovaný port.

III. *Funkcia Callback()*

Funkcia callback je použitá v knihovnej funkcii *pcap_loop()* ako jeden z parametrov a je volaná pre každý prijatý packet, ktorý nastavené rozhranie obdrží. Telo samotnej funkcie callback spracuje prijatý packet a analyzuje jednotlivé vrstvy packetu. V najspodnejšej vrstve (*Network Access Layer*) v prípade, že sa jedná o *Ethernet*, tak sa rozhodne o tom či ďalšia vrstva (*Internet Layer*) obsahuje *IPv4* alebo *IPv6* hlavičku. Následne sa môže analyzovať o úroveň vyššia vrstva (*Transport Layer*), ktorá obsahuje protokoly (napr. *TCP*, *UDP*, *ICMP*, ...). V prípade, že sa jedná o protokol *TCP* alebo *UDP*, tak funkcia *Callback()* volá funkcie (konkrétnejšie statické metódy objektu *Sniffer*), ktoré bližšie daný protokol rozoberú.

IV. *TCP/UDP*

I. *TCP (Transmission Control Protocol)*

Daný protokol je bližšie spracovaný pomocou funkcie *print_tcp()*, ktorá zistí veľkosť *TCP* hlavičky, čo je esenciálne pre získanie dát (*payload*), ktoré ak existujú, tak sú uložené za *TCP* hlavičkou. Funkcia využíva pointrovú aritmetiku, vďaka ktorej sa môže v pamäti posúvať medzi jednotlivými vrstvami.

Pre samotný výpis na štandardný výstup volá funkcia ďalšiu objektovú metódu *print_data()*, konkrétne dvakrát, pre výpis hlavičky *TCP* a *payload* (všetko za hlavičkou).

II. *UDP (User Datagram Protocol)*

Rovnako, ako pri *TCP*, tak aj pri *UDP* sa volá funkcia *print_udp()*, ktorá takmer identicky spĺňa funkcionality funkcie *print_tcp()*. Rozdiely sú predovšetkým pri získavaní veľkosti *UDP* hlavičky, ktorá je podľa štandardu veľká 8 bytov (veľkosť *TCP* hlavičky nemá vždy fixnú veľkosť a je potrebné ju získať pomocou pointrovej aritmetiky a bitových operácií). Následne je

obdobne dvakrát volaná funkcia *print_data()* pre výpis hlavičky a payload dát na štandardný výstup.

3. Testovanie

a) Referenčný stroj

Testovanie na referenčnom stroji (Ubuntu 18.04) prebehlo pre základné vstupy bez problémov a program vystopoval packety podľa zadaných argumentov. *Boli otestované všetky varianty vstupu. Viac priložené obrázky.*

```
File Edit View Search Terminal Help
root@student-vm:/media/sf_Project2# ./ipk-sniffer -i
1. enp0s3 (No description available)
2. any (Pseudo-device that captures on all interfaces)
3. lo (No description available)
4. nflog (Linux netfilter log (NFLOG) interface)
5. nfqueue (Linux netfilter queue (NFQUEUE) interface)
6. usbmon1 (USB bus number 1)
root@student-vm:/media/sf_Project2#
```

(spustenie programu bez nastavenie rozhrania)

```
File Edit View Search Terminal Help
root@student-vm:/media/sf_Project2# ./ipk-sniffer -i enp0s3 -n 2
00:37:25.082009 student-vm: 53010 > server-99-86-243-73.vie50.r.cloudfront.net: 443
0x0000:  cf 12 01 bb 7d d0 a9 a7 01 e3 b0 cf 50 10 f7 03  ....}... ....P...
0x0010:  62 c9 00 00                                     b...

00:37:25.087604 server-99-86-243-73.vie50.r.cloudfront.net: 443 > student-vm: 53010
0x0000:  01 bb cf 12 01 e3 b0 cf 7d d0 a9 a8 50 10 ff ff  .... }...P...
0x0010:  a2 2c 00 00                                     ,...

0x0000:  00 00 00 00 00 00                                .....
root@student-vm:/media/sf_Project2#
```

(spustenie programu na určitom rozhraní s výpisom dvoch packetov)

```
File Edit View Search Terminal Help
root@student-vm:/media/sf_Project2# ./lpk-sniffer -i enp0s3
00:36:23.447303 student-vm: 52423 > sagemcom: 53

0x0000:  cc c7 00 35 00 37 ce 00          ...5.7..

0x0000:  ce e2 01 00 00 01 00 00  00 00 00 00 0e 64 31 7a  ....d1z
0x0010:  6b 7a 33 6b 34 63 63 6c  6e 76 36 0a 63 6c 6f 75  kz3k4ccl nv6.clou
0x0020:  64 66 72 6f 6e 74 03 6e  65 74 00 00 1c 00 01  df front.n et.....

root@student-vm:/media/sf_Project2#
```

(spustenie programu pre určité rozhranie bez špecifikácie vypíše akýkoľvek jeden packet)

```
File Edit View Search Terminal Help
root@student-vm:/media/sf_Project2# ./lpk-sniffer -i enp0s3 --tcp -p 443
00:39:32.588412 student-vm: 49378 > ec2-54-212-104-249.us-west-2.compute.amazonaws.com: 443

0x0000:  c0 e2 01 bb 1e d5 a6 cc  02 d8 92 8e 50 10 f9 9c  ....P...
0x0010:  ab f6 00 00          ....

root@student-vm:/media/sf_Project2#
```

(spustenie programu na porte 443 vypíše jeden TCP packet)

```
File Edit View Search Terminal Help
root@student-vm:/media/sf_Project2# ./lpk-sniffer -i enp0s3 --udp -p 53
00:38:59.961913 student-vm: 36433 > sagemcom: 53

0x0000:  8e 51 00 35 00 28 cd f1          .Q.5.(..

0x0000:  71 81 01 00 00 01 00 00  00 00 00 00 03 77 77 77  q.....www
0x0010:  06 67 6f 6f 67 6c 65 03  63 6f 6d 00 00 01 00 01  .google.com.....

root@student-vm:/media/sf_Project2#
```

(spustenie programu na porte 53 vypíše jeden UDP packet)

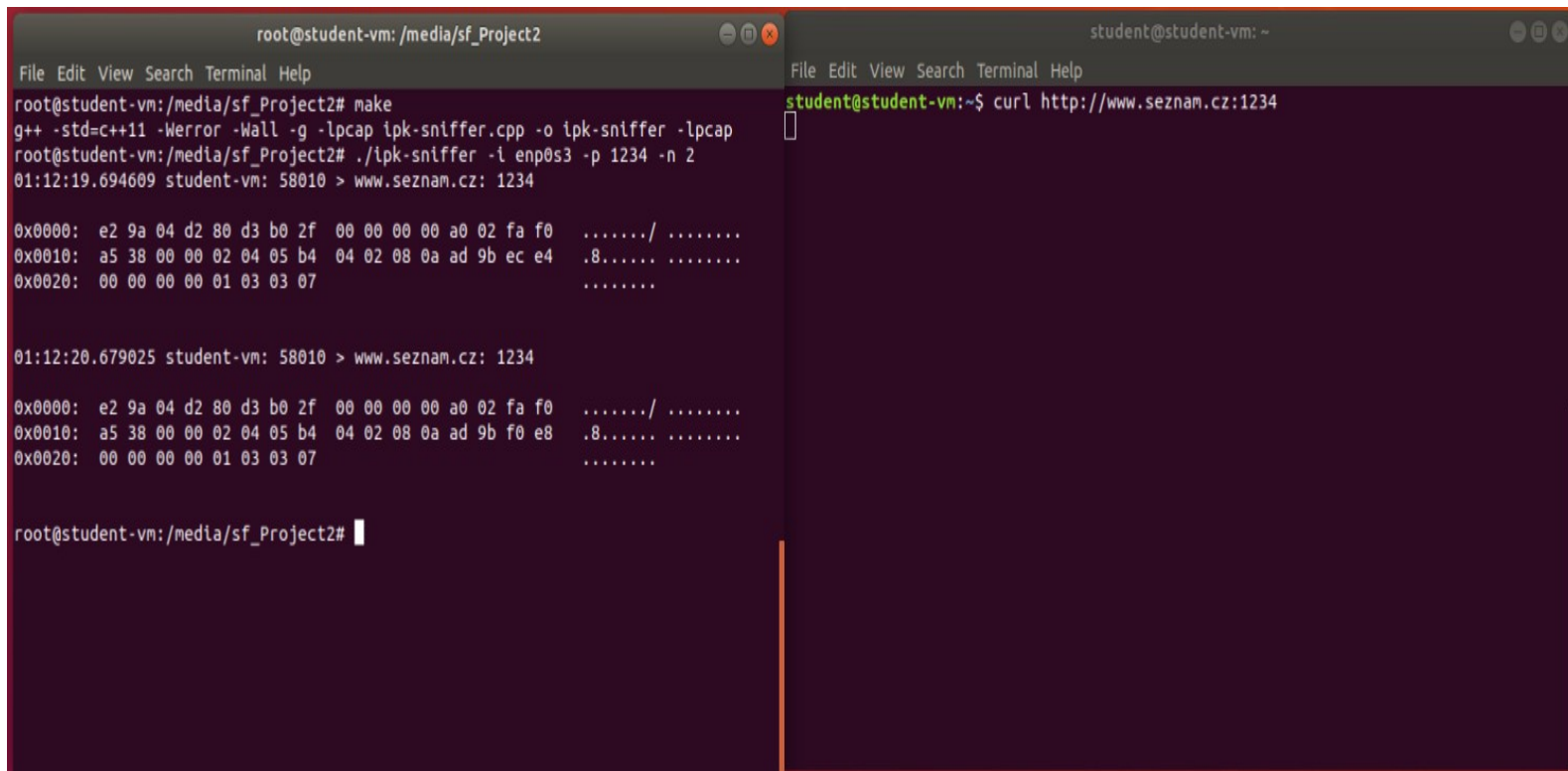
b) Vlastné prostredie Linux (Fedora 31) a Wireshark

Testovanie na linuxovom prostredí s využitím *open source* programu Wireshark, bolo použité pre kontrolu zachytávania packetov. V programe Wireshark boli zadané rovnaké filtrovacie parametre ako pre program *ipk-sniffer* a pomocou nástroja *curl*, cez ktorý sme sa pripojili na rovnaký port, boli odoslané packety, pričom ich oba programy obdržali v rovnaký čas s obsahom rovnakých hlavičiek a dát (v našom prípade je uvádzaná iba hlavička TCP/UDP protokolu a ich prípadný payload).

(Príklad testovania pomocou programu Wireshark je uvedený na obrázku po sekcií *Curl* na samostatnej stránke koli jeho veľkosti.)

c) Curl

Príkaz *curl* bol spomenutý už v predošlej sekcií, a tu je priadná iba jeho demonštrácia na referenčnom stroji. Dotazujeme sa na stránku www.seznam.cz cez port 1234 a ako výsledok môžeme vidieť packety, ktorých *FQDN* je *správny*.



The image shows two terminal windows side-by-side. The left window is titled 'root@student-vm: /media/sf_Project2' and shows the compilation and execution of 'ipk-sniffer'. The right window is titled 'student@student-vm: ~' and shows the execution of 'curl' to fetch data from 'http://www.seznam.cz:1234'. Both windows display hex and ASCII packet captures.

```
root@student-vm: /media/sf_Project2
File Edit View Search Terminal Help
root@student-vm:/media/sf_Project2# make
g++ -std=c++11 -Werror -Wall -g -lpcap ipk-sniffer.cpp -o ipk-sniffer -lpcap
root@student-vm:/media/sf_Project2# ./ipk-sniffer -i enp0s3 -p 1234 -n 2
01:12:19.694609 student-vm: 58010 > www.seznam.cz: 1234

0x0000: e2 9a 04 d2 80 d3 b0 2f 00 00 00 00 a0 02 fa f0 ...../ .....
0x0010: a5 38 00 00 02 04 05 b4 04 02 08 0a ad 9b ec e4 .8..... .....
0x0020: 00 00 00 00 01 03 03 07 .....

01:12:20.679025 student-vm: 58010 > www.seznam.cz: 1234

0x0000: e2 9a 04 d2 80 d3 b0 2f 00 00 00 00 a0 02 fa f0 ...../ .....
0x0010: a5 38 00 00 02 04 05 b4 04 02 08 0a ad 9b f0 e8 .8..... .....
0x0020: 00 00 00 00 01 03 03 07 .....

root@student-vm:/media/sf_Project2#
```

```
student@student-vm: ~
File Edit View Search Terminal Help
student@student-vm:~$ curl http://www.seznam.cz:1234
```

0x0000: d5 06 04 d2 21 56 b4 27 00 00 00 00 a0 02 fa f0!V.'
0x0010: ce c8 00 00 02 04 05 b4 04 02 08 0a 51 e2 23 60Q.#'
0x0020: 00 00 00 00 01 03 03 07

00:56:11.807220 sagemcom.ip: 54534 > www.seznam.cz: 1234

0x0000: d5 06 04 d2 21 56 b4 27 00 00 00 00 a0 02 fa f0!V.'
0x0010: 8e c8 00 00 02 04 05 b4 04 02 08 0a 51 e2 63 60Q.c'
0x0020: 00 00 00 00 01 03 03 07

00:56:43.550383 sagemcom.ip: 54534 > www.seznam.cz: 1234

0x0000: d5 06 04 d2 21 56 b4 27 00 00 00 00 a0 02 fa f0!V.'
0x0010: 10 c8 00 00 02 04 05 b4 04 02 08 0a 51 e2 e1 60Q..'
0x0020: 00 00 00 00 01 03 03 07

[root@sagemcom Project2]#

matotc@sagemcom: ~ -- curl http://www.seznam.cz:1234
File Edit View Search Terminal Help
[matotc@sagemcom ~]\$ curl http://www.seznam.cz:1234
-

tcp.port == 1234

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	192.168.1.10	77.75.74.176	TCP	74	54534 → 1234 [SYN] Seq=0 Win=64240
43	16.384026	192.168.1.10	77.75.74.176	TCP	74	[TCP Retransmission] 54534 → 1234
163	48.640238	192.168.1.10	77.75.74.176	TCP	74	[TCP Retransmission] 54534 → 1234

Frame 1: 74 bytes on wire (592 bits), 74 bytes captured (592 bits)
Ethernet II, Src: IntelCor_ad:5a:ac (c0:b6:f9:ad:5a:ac), Dst: Sagemcom_b2:ef:28 (d8:7d:7f:b2:ef:28)
Internet Protocol Version 4, Src: 192.168.1.10, Dst: 77.75.74.176
Transmission Control Protocol, Src Port: 54534, Dst Port: 1234, Seq: 0, Len: 0

0000 d8 7d 7f b2 ef 28 c0 b6 f9 ad 5a ac 08 00 45 00 .}...(. .Z...E.
0010 00 3c 3b 03 40 00 40 06 a6 0b c0 a8 01 0a 4d 4b .<;@.MK
0020 4a b0 d5 06 04 d2 21 56 b4 27 00 00 00 a0 02 J.....!V.'
0030 fa f0 ce c8 00 00 02 04 05 b4 04 02 08 0a 51 e2Q.
0040 23 60 00 00 00 00 01 03 03 07 #`.....

wlp5s0: <live capture in progress> Packets: 167 · Displayed: 3 (1.8%) Profile: Default

8

4. Zdroje

- <https://www.tcpdump.org/pcap.html>
- <https://tools.ietf.org/html/rfc793#page-15>
- <https://tools.ietf.org/html/rfc791>
- <https://tools.ietf.org/html/rfc768>
- <https://github.com/hokiespurs/velodyne-copter/wiki/PCAP-format>
- <https://www.devdungeon.com/content/using-libpcap-c>
- <https://elf11.github.io/2017/01/22/libpcap-in-C.html>
- http://web.deu.edu.tr/doc/oreily/networking/firewall/ch06_03.html