Vysoké učení technické v Brně

Fakulta informačních technologií

Počítačové komunikace a sítě

2020/2021

Projekt 2

Varianta ZETA: Sniffer paketů

Juraj Lazorík (xlazor02)

Brno, 1. mája 2020

Obsah

1. Zadanie

Navrhnite a implementujte sieťový analyzátor v C/C++/C#, ktorý bude schopný na určitom sieťovom rozhraní zachytávať a filtrovať pakety(sniffer).

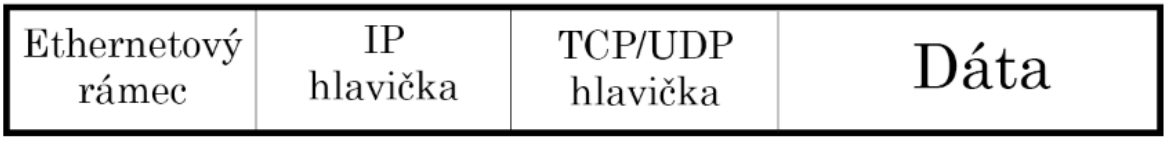
2. Úvod

Sniffer je aplikácia, ktorá zachytáva a filtruje pakety na zadanom sieťovom rozhraní. Aplikácia umožňuje užívateľovi zobraziť aktívne sieťové rozhrania a následne sledovať TCP alebo UDP pakety na vybranom rozhraní, prípadne aj na vybranom porte daného rozhrania. Výstup snifferu obsahuje čas prijatia paketu, IP adresu/príslušné doménové meno a port zdroja paketu, IP adresu/príslušné doménové meno a port cieľa paketu. Nasleduje obsah hlavičiek a obsah dát paketu v hexadecimálnej forme a v ASCII forme kde sú netlačiteľné znaky nahradené bodkou. Taktiež je možné vypísať si obsah DNS cache programu.

2.1. Paket

Paket, ktorý zachytávame sa skladá z viacero častí: IP hlavička, TCP/UDP hlavička, dáta. Na určenie, či sa jedná o IPv4 alebo IPv6 paket použijeme ethernetový rámec, ktorý uchováva informáciu o type paketu v časti next header.

Z IP hlavičky je pre nás dôležitá zdrojová/cieľová IP adresa a použitý protokol. V našom prípade nás zaujímajú len protokoly TCP a UDP. V ich hlavičkách sa nachádza zdrojový a cieľový port. Za týmito hlavičkami sa už nachádzajú dáta paketu.



Obrázok 1 – Zapúzdrenie dát

2.2. Resolving

Resolving znamená preklad IP adresy na príslušné doménové meno. To sa nachádza v DNS zázname PTR, ktorý sa využíva na spätné vyhľadávanie doménového mena. V našom projekte využívame resolving pri zobrazovaní informácií o pakete, no nastáva tu problém. Samotné získavanie doménového mena vytvára pakety medzi našou aplikáciou a DNS serverom, čo spôsobuje cyklus pri sniffovaní a tým značne znehodnocuje fungovanie aplikácie. Tento problém sme riešili dvoma spôsobmi. Prvý zo spôsobov riešenia umožňuje užívateľovi zapnúť/vypnúť funkciu resolvingu a teda možnosť sledovať pakety bez rušenia paketmi vytváraných aplikáciou. Druhé riešenie zahŕňa vytvorenie takzvanej DNS cache v programe, čím sa zamedzí cyklenie pri preklade, no aplikácia stále vytvára pakety potrebné na získanie doménového mena. Tie sú ale v značnej miere minimalizované využitím DNS cache.

3. Spustenie programu

Projekt rozbalíme z archívu tar. Následne potrebujeme vytvoriť spustiteľný súbor. Ten vytvoríme pomocou príkazu `make`, ktorý preloží zdrojový súbor `ipk-sniffer.c`. V prípade, že spustiteľný súbor chceme odstrániť, použijeme príkaz `make clean`, ktorý súbor vymaže. Program spustíme nasledovne:

```
$ ./ipk-sniffer.c [-i rozhranie] [-p port] [-t] [-u] [-n num] [-r] [-d]
```

Všetky argumenty sú voliteľné. Argumenty môžu byť zadané v ľubovoľnom poradí. Po prepínačoch `-i`, `-p` a `-n` musí nasledovať odpovedajúci argument.

3.1. Argumenty

Spustenie programu bez argumentov vypíše zoznam aktívnych rozhraní. Spustenie programu bez parametrov `-t` a `-u` zobrazí TCP aj UDP pakety zároveň.

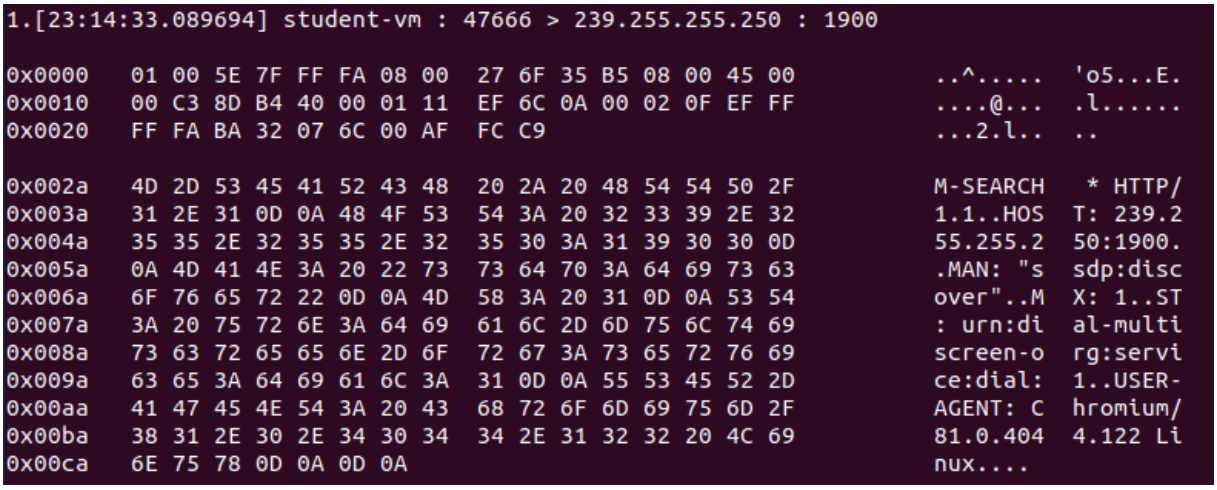
Argumenty:

- `-i rozhranie`: rozhranie, na ktorom sa budú sniffovať pakety
- `-p port`: filtrovanie paketov podľa portu `port` na danom rozhraní, bez uvedenia tohto parametru sa zobrazujú všetky porty
- `-t`: zobrazenie iba TCP paketov
- `-u`: zobrazenie iba UDP paketov
- `-n num`: určuje počet paketov `num`, ktoré sa zobrazia(v prípade záporného čísla zobrazuje pakety do ukončenia programu napr. pomocou CTRL+C), bez uvedenia sa zobrazí 1 paket
- `-r`: spustí resolving IP adries pri výpise paketov
- `-d`: zobrazí obsah DNS cache aplikácie(odporúčané použitie zároveň s arg. `-r`)

3.2. Príklad spustenia programu

- `$ ./ipk-sniffer -i enp0s3 -p 80 -t -n 5`
- `$ ./ipk-sniffer -i enp0s3 -u`
- `$ ./ipk-sniffer`
- `$ ./ipk-sniffer -n`
- `$ ./ipk-sniffer -p 443 -t -u`
- `$ ./ipk-sniffer -i enp0s3`
- `$ ./ipk-sniffer -i enp0s3 -n 5 -r -d`

3.3. Príklad výstupu



Obrázok 2 – Príklad výstupu

Popis výstupu:

- 1. – poradové číslo paketu
- [23:14:33.089694] – čas prijatia paketu v tvare HH:MM:SS.MS
- student-vm – hostname zdroja paketu
- 47666 – port zdroja paketu
- 239.255.255.250 – IP cieľa paketu
- 1900 – port cieľa paketu

Stĺpec vľavo zobrazuje počet zobrazených hexadecimálnych hodnôt. Stredné dva stĺpce zobrazujú obsah hlavičky a dát v hexadecimálnom tvare(oddelené prázdnym riadkom).
Pravé dva stĺpce zobrazujú obsah hlavičky a dát v ASCII tvare(netlačiteľné znaky sú nahradené bodkou.

4. Návrh riešenia

Program ipk-sniffer sme implementovali v jazyku C.

4.1. Spracovanie argumentov

Ako prvé, po spustení programu spracujeme argumenty. Kontrolu argumentov vykonávame za pomoci funkcie `getopt`. Každý argument následne uložíme do premennej, s ktorou budeme pracovať. V prípade chybne zadaných argumentov vypíšeme príslušnú chybovú hlášku a poskytneme vzorový príklad na spustenie.

4.2. Sniffing paketov

V prípade, ak uživateľ nezadá žiadny argument zobrazíme všetky dostupné rozhrania. Tie získame volaním funkcie `pcap_findalldevs`. Pri zadaní správnych parametrov si pripravíme rozhranie na sniffovanie v promiskuitnom móde pomocou `pcap_open_live`. Nasleduje spracovanie filtra podľa argumentov. Filtrovací výraz následne predáme funkcii `pcap_compile`, ktorá spracuje daný výraz tak aby sme ho mohli použiť vo funkcii

`pcap_setfilter`. V tomto momente je všetko pripravené a môžeme zahájiť sniffovanie v reálnom čase s funkciou `pcap_loop`, ktorá následne volá funckiu `my_packet_handler` a tá zabezpečí spracovanie informácií z každého paketu.

Pri spracovaní paketov využívame tieto štruktúry v tomto poradí:

1. `ether_header` – Z tejto štruktúry zisťujeme, či sa jedná o IPv4 alebo IPv6 paket a to v položke `ether_type`.
2. `iphdr/ip6_hdr` – Ako už z názvu vyplýva, jedná sa o štruktúry na uloženie IPv4/IPv6 hlavičky. V nich je pre nás zaujímavá položka `protocol` na určenie protokolu paketu. Ďalej potrebujeme položky `saddr/ip6_src` a `daddr/ip6_dst` pre získanie zdrojovej a cieľovej IP adresy.
3. `tcphdr/udphdr` – Táto štruktúra uchováva informácie o zdrojovom a cieľovom porte a to v položkách `source` a `dest`.

4.3. Výpis paketov

Pakety vypisujeme pomocou jednoduchých `for` cyklov v hexadecimálnom a ASCII tvare. Pri prevode hodnôt do ASCII vypisujeme iba znaky v rozmedzí hodnôt <32,127) a teda znaky, ktoré sú tlačiteľné. Ostatné nahrádzame bodkou.

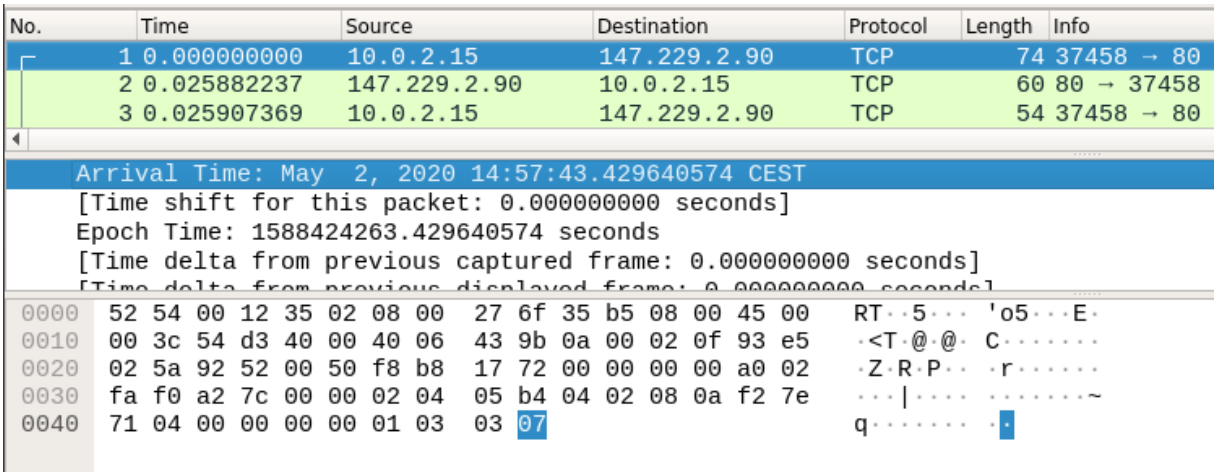
4.4. DNS resolving

Preklad IP adresy na doménové meno vykonávame pomocou funkcie `getnameinfo`. Túto funkciu ale voláme len v prípade, ak sa daná IP adresa nenachádza v DNS cache aplikácie. Ak funkcia nájde doménové meno k IP, uložíme tento údaj do tejto cache. Cache pozostáva z dvoch polí `ipCacheArray` a `hostnameCacheArray` a uchováva IP a jej hostname.

5. Testovanie

Program bol testovaný na virtuálnom stroji, ktorý bol vytvorený z poskytnutého referenčného Linux obrazu. Na porovnanie výsledkov budeme používať nástroj Wireshark.

Pakety vyvolané príkazom „curl www.vutbr.cz“.



Obrázok 3 – Výstup nástroja Wireshark

```
student@student-vm:~/Desktop/IPK2 s IPv6 working$ sudo ./ipk-sniffer -i enp0s3 -n 3
1.[14:57:43.429640] 10.0.2.15 : 37458 > 147.229.2.90 : 80

0x0000   52 54 00 12 35 02 08 00  27 6F 35 B5 08 00 45 00          RT..5...  'o5...E.
0x0010   00 3C 54 D3 40 00 40 06  43 9B 0A 00 02 0F 93 E5          .<T.@.@.  C.......
0x0020   02 5A 92 52 00 50 F8 B8  17 72 00 00 00 00 A0 02          .Z.R.P..  .r......
0x0030   FA F0 A2 7C 00 00 02 04  05 B4 04 02 08 0A F2 7E          ...|....  .......~
0x0040   71 04 00 00 00 00 01 03  03 07                           q.......  ..


############################################################

2.[14:57:43.455522] 147.229.2.90 : 80 > 10.0.2.15 : 37458

0x0000   08 00 27 6F 35 B5 52 54  00 12 35 02 08 00 45 00          ..'o5.RT  ..5...E.
0x0010   00 2C 01 E3 00 00 40 06  D6 9B 93 E5 02 5A 0A 00          .,....@.  .....Z..
0x0020   02 0F 00 50 92 52 00 D3  EA 01 F8 B8 17 73 60 12          ...P.R..  .....s`.
0x0030   FF FF 68 25 00 00 02 04  05 B4                           ..h%....  ..

0x003a   00 00                                                    ..

############################################################

3.[14:57:43.455547] 10.0.2.15 : 37458 > 147.229.2.90 : 80

0x0000   52 54 00 12 35 02 08 00  27 6F 35 B5 08 00 45 00          RT..5...  'o5...E.
0x0010   00 28 54 D4 40 00 40 06  43 AE 0A 00 02 0F 93 E5          .(T.@.@.  C.......
0x0020   02 5A 92 52 00 50 F8 B8  17 73 00 D3 EA 02 50 10          .Z.R.P..  .s....P.
0x0030   FA F0 A2 68 00 00                                        ...h..


############################################################
```

Obrázok 4 – Výstup aplikácie ipk-sniffer

Kvôli prehľadnosti sme nezobrazili všetky odoslané pakety a ich obsah, no ako môžeme vidieť na prvom ukážkovom pakete, informácie sa zhodujú a všetko funguje ako má. Následne si ukážeme paket s protokolom UDP:

Pakety vyvolané príkazom „nc -u www.vutbr.cz 53“.

No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000000	10.0.2.15	8.8.8.8	DNS	83	Standa
2	0.027459365	8.8.8.8	10.0.2.15	DNS	99	Standa

```
    Arrival Time: May  2, 2020 15:27:50.673960904 CEST
    [Time shift for this packet: 0.000000000 seconds]
    Epoch Time: 1588426070.673960904 seconds
    [Time delta from previous captured frame: 0.000000000 seconds]
0000  52 54 00 12 35 02 08 00  27 6f 35 b5 08 00 45 00   RT··5··· 'o5···E·
0010  00 45 46 16 40 00 40 11  d8 73 0a 00 02 0f 08 08   ·EF·@·@· ·s······
0020  08 08 a6 39 00 35 00 31  1c 61 98 79 01 00 00 01   ···9·5·1 ·a·y····
0030  00 00 00 00 00 01 03 77  77 77 05 76 75 74 62 72   ·······w ww·vutbr
0040  02 63 7a 00 00 01 00 01  00 00 29 02 00 00 00 00   ·cz····· ··)·····
0050  00 00 00                                          ···
```

Obrázok 5 – Výstup nástroja Wireshark

```
student@student-vm:~/Desktop/IPK2 s IPv6 working$ sudo ./ipk-sniffer -i enp0s3
1.[15:27:50.673960] 10.0.2.15 : 42553 > 8.8.8.8 : 53

0x0000   52 54 00 12 35 02 08 00  27 6F 35 B5 08 00 45 00          RT..5...  'o5...E.
0x0010   00 45 46 16 40 00 40 11  D8 73 0A 00 02 0F 08 08          .EF.@.@.  .s......
0x0020   08 08 A6 39 00 35 00 31  1C 61                          ...9.5.1  .a

0x002a   98 79 01 00 00 01 00 00  00 00 00 01 03 77 77 77          .y......  .....www
0x003a   05 76 75 74 62 72 02 63  7A 00 00 01 00 01 00 00          .vutbr.c  z.......
0x004a   29 02 00 00 00 00 00 00  00                             ).......  .

############################################################
```

Obrázok 6 – Výstup aplikácie ipk-sniffer

Opäť nezobrazujeme všetky pakety pre prehľadnosť. Protokol DNS v tomto prípade spadá pod UDP, takže všetko funguje správne a rovnako aj obsah je totožný. Pre úplnosť otestujeme aj IPv6 paket.

Pakety vyvolané príkazom „curl -g -6 "http://[::1]:80/"“.

```
No.  Time          Source  Destination  Protocol  Length  Info
     1 0.000000000 ::1     ::1          TCP         94    35634 → 80
     2 0.000006253 ::1     ::1          TCP         74    80 → 35634

  Interface id: 0 (lo)
  Encapsulation type: Ethernet (1)
  Arrival Time: May  2, 2020 15:38:21.949365022 CEST
  [Time shift for this packet: 0.000000000 seconds]

0000  00 00 00 00 00 00 00 00  00 00 00 00 86 dd 60 0b  ........ ......`.
0010  9c f4 00 28 06 40 00 00  00 00 00 00 00 00 00 00  ...(.@.. ........
0020  00 00 00 00 00 01 00 00  00 00 00 00 00 00 00 00  ........ ........
0030  00 00 00 00 00 01 8b 32  00 50 57 d0 36 d6 00 00  .......2 .PW.6...
0040  00 00 a0 02 ff c4 00 30  00 00 02 04 ff c4 04 02  .......0 ........
0050  08 0a 3e c3 5e d6 00 00  00 00 01 03 03 07        ..>.^... ......
```

Obrázok 7 – Výstup nástroja Wireshark

```
student@student-vm:~/Desktop/IPK2 s IPv6 working$ sudo ./ipk-sniffer -i lo
1.[15:38:21.949365] ::1 : 35634 > ::1 : 80

0x0000   00 00 00 00 00 00 00 00  00 00 00 00 86 DD 60 0B          ........  ......`.
0x0010   9C F4 00 28 06 40 00 00  00 00 00 00 00 00 00 00          ...(.@..  ........
0x0020   00 00 00 00 00 01 00 00  00 00 00 00 00 00 00 00          ........  ........
0x0030   00 00 00 00 00 01 8B 32  00 50 57 D0 36 D6 00 00          .......2  .PW.6...
0x0040   00 00 A0 02 FF C4 00 30  00 00 02 04 FF C4 04 02          .......0  ........
0x0050   08 0A 3E C3 5E D6 00 00  00 00 01 03 03 07                ..>.^...  ......


############################################################
```

Obrázok 8 – Výstup aplikácie ipk-sniffer

Rovnako ako v predošlých prípadoch sa výstupy zhodujú a program funguje správne.

6. Zdroje

[1] Moon, S.: Packet Sniffer Code in C using sockets | Linux [online], 2013, [vid. 1.05.2020]. Dostupné z: https://www.binarytides.com/packet-sniffer-code-c-linux/

[2] NanoDano: Using libpcap in C [online], 2015, [vid. 1.05.2020]. Dostupné z: https://www.devdungeon.com/content/using-libpcap-c

[3] Flexo♦: Can I use pcap library for receiving ipv6 packets? [online], 2011, [vid. 1.05.2020]. Dostupné z: https://stackoverflow.com/questions/6256821/can-i-use-pcap-library-for-receiving-ipv6-packets

[4] Algorism: getnameinfo() example problem [online], 2012, [vid. 1.05.2020]. Dostupné z: https://cboard.cprogramming.com/c-programming/169902-getnameinfo-example-problem.html