

IPK Project 2

Varianta ZETA: Sniffer paketů

24.4.2022

Milan Hrabovský
xhrabo15

Obsah

1. Úvod

2. Implementácia

- 1.1. Argument parser
- 1.2. Zachytávanie paketov
 - 1.2.1. UDP
 - 1.2.2. TCP
 - 1.2.3. ARP
 - 1.2.4. ICMP

3. Zdroje

Úvod

Zachytávanie paketov je technika ktorou je možné sledovať sieťové údaje prichádzajúce do počítača a odchádzajúce z počítača. Údaje sa prenášajú po sieti vo forme paketov ktoré obsahujú všetky potrebné údaje pre prenos po sieti a prenášané dáta. Pakety sa zachytávajú na vybranom rozhraní zariadenia ako je napríklad eth0, any, atď. podľa zadaného filtra, ktorý môže filtrovať na základe zadaného protokolu, portu, typu ip adresy atď. Užívateľ si vie taktiež zvoliť dĺžku behu programu tj. Koľko paketov bude zachytených. Na zachytávanie paketov užívateľ musí mať potrebné oprávnenia.

Implementácia

Na implementovanie paketového snifferu som použil knižnicu pcap.h ktorá obsahuje všetky potrebné funkcie a štruktúry na zachytávanie paketov. Ďalej používam knižnice z kolekcie netinet a arpa, kde sú pomocné štruktúry k spracováaniu informácií z paketov. Na spracovanie a výpis času využívam knižnicu time.h.

Argument parser

Na spracovanie argumentov príkazovej riadky som si definoval vlastnú triedu ArgumentParser ktorá načíta všetky argumenty a získa tak potrebné údaje na vytvorenie príslušnej konfigurácie pre zachytávanie paketov. Pri inicializácii triedy constructor prejde všetky argumenty príkazovej riadky, načíta ich a skontroluje ich parametre. Pokiaľ je zadaný neznámy parameter, program sa ukončí s chybou 1. Trieda ArgumentParser zároveň obsahuje filter a metódu makeFilter ktorá je zavolaná po načítaní argumentov a vygeneruje filter potrebný na zachytávanie paketov podľa argumentov zadaných v príkazovom riadku.

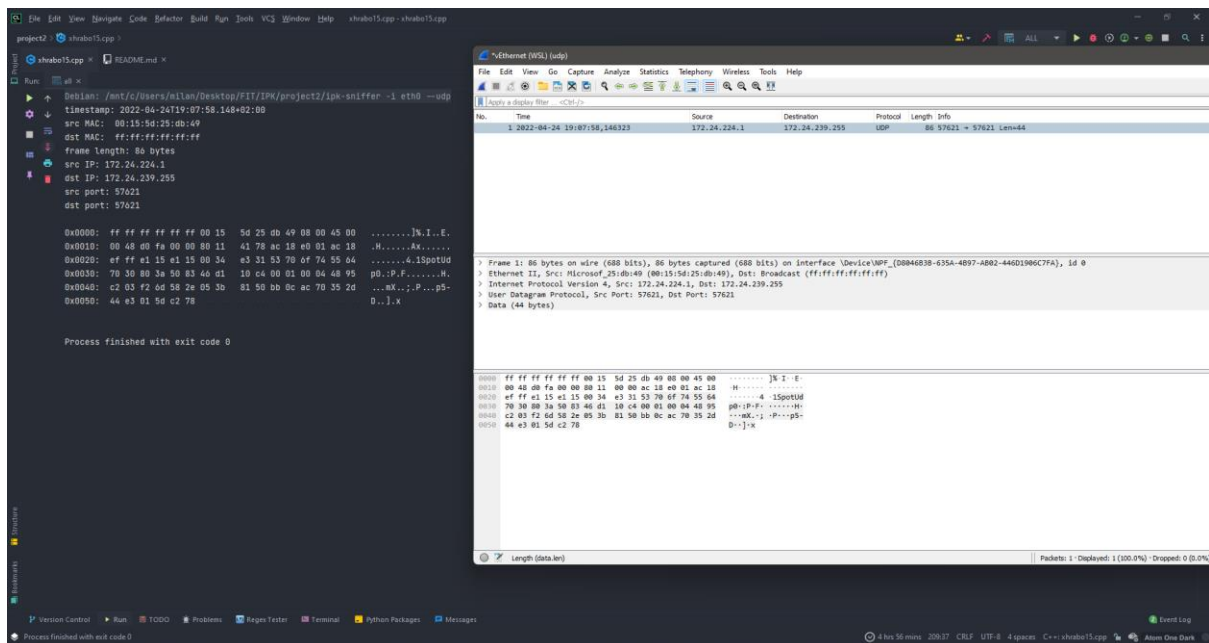
Zachytávanie paketov

Zachytávanie paketov som implementoval podľa [návodu](#), kde využívam funkcie z knižnice pcap.h.

Po spracovaní argumentov je zavolaná funkcia pcap_findalldevs, ktorá získa všetky aktívne rozhrania a uloží ich do zoznamu. Pokiaľ užívateľ nešpecifikuje žiadne rozhranie je tento zoznam aktívnych vypísaný a program je následne úspešne ukončený, v opačnom prípade ak je špecifikované rozhranie, pomocou funkcie pcap_lookupnet program načíta sieťovú adresu a masku a pripraví zariadenie na zachytávanie paketov. Následne funkcia pcap_compile spracuje filter ktorý funkcia pcap_setfilter potom nastaví. Následne je zavolaná funkcia pcap_loop ktorá zachytáva pakety podľa zadaného filtra a vracia paket vo funkcii callback, kde sú pakety ďalej spracované. Maximálny počet zachytených paketov špecifikuje užívateľ, ak nič nezadá, základne je počet nastavený na 1 paket. Po úspešnom získaní paketu sa ďalej spracuje vo funkcii callback, kde sa prevedie časová známka na aktuálny čas vo formáte RFC3339. Ďalej je získaná src MAC adresa, dst Mac adresa a dĺžka paketu, ktoré sú následne vypísané na štandardný výstup. Ďalej sa získava EtherType podľa ktorého je ďalej spracovaná hlavička v pakete. Spracováva sa ipv4 hlavička, ipv6 hlavička alebo arp hlavička. Následne je celý paket vypísaný HEX aj ASCII formáte

UDP

Zachytáva sa ipv4 alebo ipv6 adresa, podľa ktorej je hlavička paketu ďalej spracovaná. Na získanie portov adresy používam štruktúru udphdr.



The screenshot displays two windows. The left window is a C++ IDE showing the output of a program named 'xhobot15.cpp'. The output shows the capture of a UDP packet with the following details:

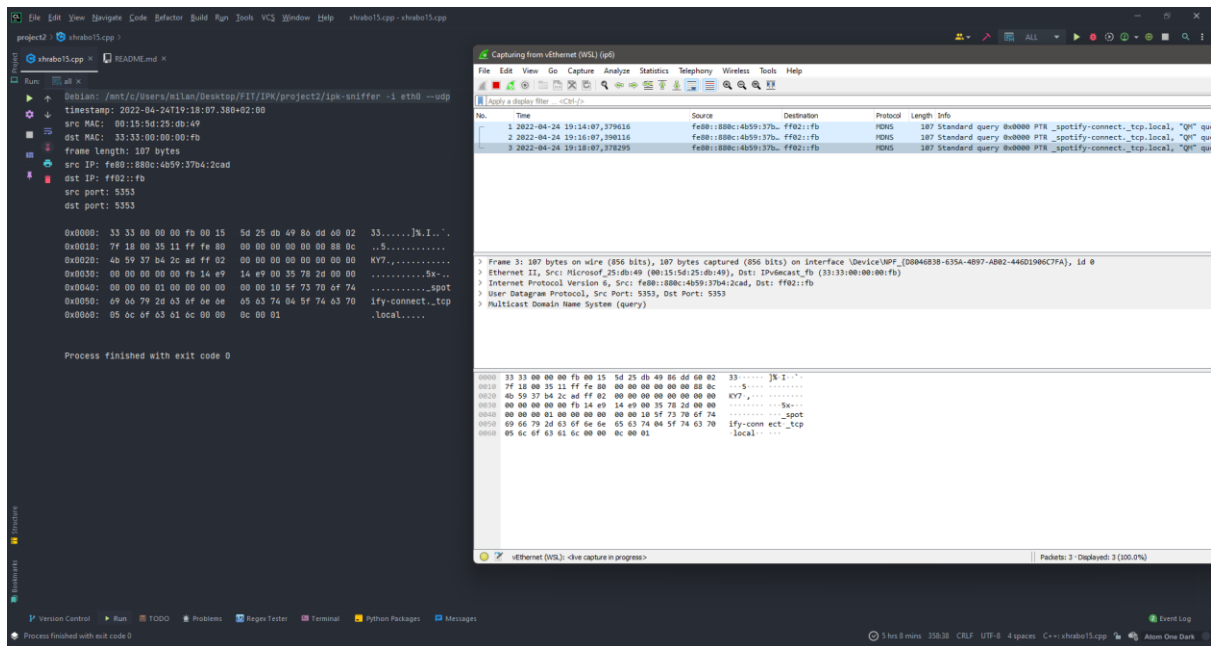
```
timestamp: 2022-04-24T19:07:58.148+02:00
src MAC: 00:15:5d:25:db:49
dst MAC: ff:ff:ff:ff:ff:ff
frame length: 86 bytes
src IP: 172.24.224.1
dst IP: 172.24.239.255
src port: 57621
dst port: 57621

0x0000: ff ff ff ff ff 00 15 5d 25 db 49 00 45 00 .....N.I..E.
0x0010: 00 48 d0 fa 00 00 80 11 41 78 ac 18 e0 01 ac 18 .H.....AK.....
0x0020: ef ff e1 15 e1 15 00 34 e3 31 53 70 0f 74 55 64 .....4:spotUD
0x0030: 70 30 80 3a 50 83 46 d1 10 c4 00 01 00 04 48 95 p0:P.P.....M.
0x0040: c2 03 f2 66 58 2a 05 3b 81 50 bb 8c ac 70 35 2d ...M...;P...p5-
0x0050: 44 e3 01 5d c2 78 0...X
```

The right window is Wireshark, showing a capture of a UDP packet. The packet list shows a single packet of length 86 bytes. The packet details pane shows the following structure:

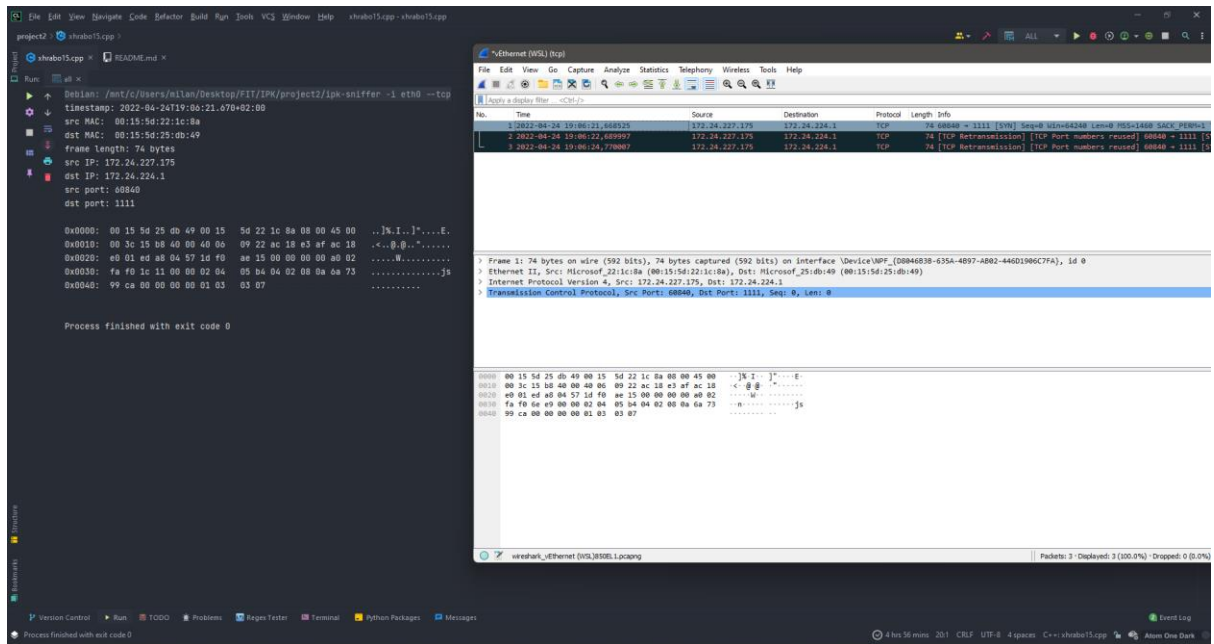
- Ethernet II, Src: Microsoft (08:00:00:00:00:00), Dst: Broadcast (ff:ff:ff:ff:ff:ff)
- Internet Protocol Version 4, Src: 172.24.224.1, Dst: 172.24.239.255
- User Datagram Protocol, Src Port: 57621, Dst Port: 57621
- Data (44 bytes)

The packet bytes pane shows the raw data in hexadecimal and ASCII format, matching the output of the C++ program.



TCP

Zachytáva sa ipv4 alebo ipv6 adresa, podľa ktorej je hlavička paketu ďalej spracovaná. Na získanie portov adresy používam štruktúru tcphdr.



ARP

Nezískavajú sa ip adresy ani porty, vypisujú sa len čas, MAC adresy, dĺžka a paket

The image shows a C++ program running in a terminal window and a Wireshark packet capture. The terminal window displays the output of the program, which is a list of ARP requests and replies. The Wireshark window shows the captured packets, including the ARP request and reply. The terminal output is as follows:

```
project2: ~$ ./xhabs15.cpp
Debian: /mnt/c/Users/milan/Desktop/FIT/IKP/project2/ipk-sniffer -i eth0 --arp
timestamp: 2022-04-24T19:10:24.092+02:00
src MAC: 08:15:5d:22:1c:8a
dst MAC: ff:ff:ff:ff:ff:ff
frame length: 58 bytes

0x0000: ff ff ff ff ff ff 00 15 5d 22 1c 8a 00 00 01 .....].....
0x0010: 08 00 0a 04 00 01 00 15 5d 22 1c 8a ac 18 e3 af .....}.....
0x0020: 00 00 00 00 00 00 0e fb 24 83 00 00 00 00 00 .....$......
0x0030: 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....

Process finished with exit code 0
```

The Wireshark window shows the captured packets, including the ARP request and reply. The packet list shows three packets: an ARP request from Microsoft_22:1c:8a to Broadcast, an ARP request from Microsoft_25:db:49 to Microsoft_22:1c:8a, and an ARP request from Microsoft_22:1c:8a to Microsoft_25:db:49. The packet details show the Ethernet II, Internet Protocol Version 4, and Internet Control Message Protocol (request) fields. The packet bytes show the raw data of the ARP request and reply.

ICMP

Zachytáva sa ipv4 alebo ipv6 adresa, podľa ktorej je hlavička paketu ďalej spracovaná. Porty nie sú získavané lebo icmp/icmp 6 nepotrebuje porty

The image shows a C++ program running in a terminal window and a Wireshark packet capture. The terminal window displays the output of the program, which is a list of ICMP requests and replies. The Wireshark window shows the captured packets, including the ICMP request and reply. The terminal output is as follows:

```
project2: ~$ ./xhabs15.cpp
Debian: /mnt/c/Users/milan/Desktop/FIT/IKP/project2/ipk-sniffer -i eth0 --icmp
timestamp: 2022-04-24T19:11:44.407+02:00
src MAC: 08:15:5d:22:1c:8a
dst MAC: 00:15:5d:25:db:49
frame length: 98 bytes
src IP: 172.24.227.175
dst IP: 142.251.36.131

0x0000: 00 15 5d 25 db 49 00 15 5d 22 1c 8a 00 00 45 00 ...[N.I...].....E.
0x0010: 00 54 8d 18 40 00 00 01 0a 4a ac 18 e3 af 8e fb ...T...B...J}.....
0x0020: 24 83 08 0a de 68 2a 00 01 00 04 65 62 00 00 ...$....h....eb...
0x0030: 00 00 ea 38 0a 00 00 00 00 00 18 11 12 13 14 15 ...B.....
0x0040: 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 .....!*"#$%&'()*+,-./0123456789:;=<>[]^_`{|}~.
0x0050: 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 6'()*+,-./0123456789:;=<>[]^_`{|}~.
0x0060: 36 37 67

Process finished with exit code 0
```

The Wireshark window shows the captured packets, including the ICMP request and reply. The packet list shows four packets: an ICMP echo (ping) request from 172.24.227.175 to 142.251.36.131, an ICMP echo (ping) reply from 142.251.36.131 to 172.24.227.175, an ICMP echo (ping) request from 172.24.227.175 to 142.251.36.131, and an ICMP echo (ping) reply from 142.251.36.131 to 172.24.227.175. The packet details show the Ethernet II, Internet Protocol Version 4, and Internet Control Message Protocol fields. The packet bytes show the raw data of the ICMP request and reply.

Zdroje

[Wikipedia](#)

[Pcap library](#)

[Time convert](#)

[Getting ip address](#)

[Getting MAC address](#)