

# IPK – Projekt 2: Zeta

Manuál  
Ondřej Sloup

## Obsah

Volba projektu a prostředí	3
Použité knihovny	3
Použité zdroje	3
Popis programu	4
ArgumentParser	4
Funkce	4
Návratové kódy	5
Testování	6
Příklady spuštění	6
Windows	6
Manjaro	8
Ubuntu	11
Reference	13

## Volba projektu a prostředí

Zvolil jsem si tento projekt z daných možností, protože mi jeho zadání přišlo zajímavé, a i lehce proveditelné. Přesnou představu, co má program dělat jsem získal díky častému používání softwaru WireShark.

Program je jednoduchá konzolová aplikace napsaná v C# .NET 3.1 (zachování kompatibility na referenčním počítači). Zvolil jsem si toto prostředí kvůli mým předchozím zkušenostem v C# a samotné jednoduchosti jazyka C#.

Aplikace je multiplatformní a byla testována i sestavena na Windows i na Linux.

## Použité knihovny

Knihovny jsou nainstalované pomocí balíčkového manageru NuGet, který je automaticky importoval a zakomponoval do projektu.

Použil jsem knihovny:

- [System.CommandLine](#) – .NET knihovna pro parsování argumentů a dynamického generování outputu
- [SharpPcap](#) – .NET knihovna pro zachytávání packetů ze zařízení

## Použité zdroje

Při psaní aplikace jsem se inspiroval řešením z příkladů z oficiálních GitHubů jednotlivých použitých knihoven (Pluskal, a další, 2021) (Sequeira, a další, 2021) a teorii jsem čerpal z CodeProject (Gal, a další, 2014) projektu.

## Popis programu

Konzolová aplikace obsahuje 2 třídy – **ArgumentParser.cs** a **NetworkTools.cs**. Třída **ArgumentParser** se stará a překlad uživatelských argumentů do vnitřních proměnných, které dále kontrolují tok programu. Třída **NetworkTools** je knihovna jednotlivých funkcí, které zajišťují funkčnost snifferu a listování zařízení.

### ArgumentParser

Třída pracuje s knihovnou **System.CommandLine**, která parsuje jednotlivé argumenty, jak je specifikováno v zadání. Knihovna byla doporučena na fóru a generuje veškeré potřebné informace včetně `--help` a `--version`.

Třída také ověřuje, zda port je ve validním rozmezí od 0 do 65535 a zda není specifikován argument `-p` s argumentem `--icmp` nebo `--arp` což není možné. Pokud je zde i jiný specifikátor (například `-t`) je vypsáno varování.

### NetworkTools

Třída Network tools obsahuje soubor funkcí, které jsou používány pro získávání informací o packetech a jednotlivých zařízeních.

Klíčovým prvkem je knihovna **SharpPcap**, která je používána pro veškeré operace – získávání zařízení, filtrace a výpis packetů.

### Funkce

- **ListDevices()** – Funkce vyhledá veškerá dostupná zařízení a vytvoří dictionary, které obsahuje seznam všech zajímavých informací o zařízení – Název, MAC adresa, uživatelsky přívětivé jméno a popis
- **SniffPacket()** – Hlavní funkce, která je zavolána pro zjišťování provozu na síti. Jsou ji předány argumenty, podle kterých najde správné zařízení, vytvoří filtr a započne prohledávání sítě
- **OnArrivalHandler()** – Funkce (handler) je napojena na vnitřní funkci knihovny **SharpPcap** – **device.StartCapture()** – a je zavolána v případě příchozího packetu. Tato funkce se pokusí extrahovat packet na jeden z podporovaných a vypíše informace o jeho IP, portu (pokud je dostupný) a jeho data. Zároveň počítá počet již vypsáných packetů, aby byl splněn argument `-n`.

- **GetDeviceInfo()** – Funkce pro překlad jména zařízení z uživatelského vstupu na validní hodnotu. Funkce akceptuje uživatelsky přívětivé jméno jako argument.
- **WriteTcpOrUdp()** – Funkce vypisuje informace o TCP nebo UDP packetu jak pro IPv4, tak i pro IPv6.
- **WriteIcmp()** – Funkce vypisuje informace o ICMP packetu jak pro IPv4, tak i pro IPv6.
- **WritePacketData()** – Funkce pro vypsání dat z packetu v HEX i ASCII včetně offsetu. Funkce ověřuje pomocí jednoduché podmínky, jestli se jedná o tisknutelný či netisknutelný znak.
- **CreateFilter()** – Funkce vytvoří textovou podmínku na základě uživatelských parametrů, která je předána filtru. (Tato podmínka jde vypsát odkomentováním 80. řádku v kóde

## Návratové kódy

Soubor ReturnCode.cs obsahuje veškeré návratové kódy aplikace a jejich význam.

## Testování

Aplikaci jsem otestoval na systémech:

1. Windows 10 Pro, verze 20H2 (OS Build: 19042.867)
2. Manjaro (Linux 5.10.26-1-MANJARO)
3. Ubuntu 20.04.2 LTS - Linux 5.8.0-48-generic – referenční počítač

Bylo testováno samotné sestavení projektu, a i jeho spuštění.

## Příklady spuštění

Citlivé údaje byly cenzurovány.

### Windows

```
./ipk-sniffer -i "Ethernet" -n 2 --arp -p 443 -u -t -icmp
```

```

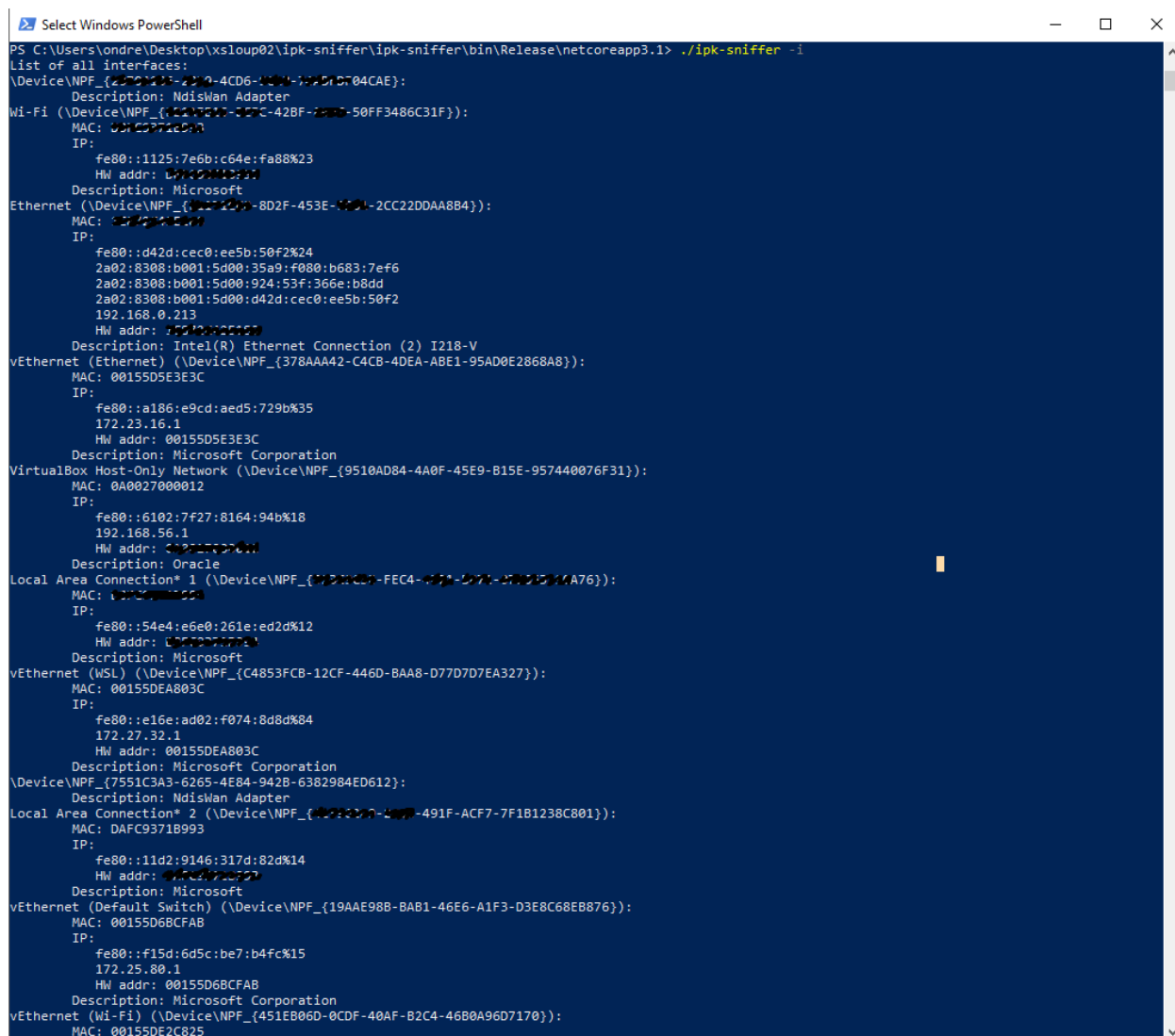
PS C:\Users\ondre\Desktop\xsloup02\ipk-sniffer\ipk-sniffer\bin\Release\netcoreapp3.1> ./ipk-sniffer -i "Ethernet" -n 2 --arp -p 443 -u -t -icmp
Warning: Filtering using ARP or ICMP with port is not possible.
Connected to \Device\NPF_{F1171CC8-8D2F-453E-9B61-2CC2DDAA8B4}
(TCP) 2021-04-18T20:07:32.547+00:00: 54.64.115.100 443 > 192.168.0.213 51653, length 319 bytes
0x0000: 45 00 00 34 2c c4 82 e1 81 08 00 45 00 ..B..4, .....E.
0x0010: 01 31 b4 3f 40 00 e6 06 74 65 36 40 73 64 c0 a8 .1.?@... te6@sd..
0x0020: 00 d5 01 bb c9 c5 b1 26 fb 92 23 1b 2f c8 50 18 .....& ..#./.P.
0x0030: 00 79 f2 e9 00 00 17 03 03 01 04 f0 1f 64 aa d0 .y..... ..d..
0x0040: 8e a3 6f 00 cb 31 e3 cc b6 0d ea 54 05 31 3f 04 ..o.1.. ...T.1?.
0x0050: 4e b6 ca 86 dc 33 93 00 57 74 1e f7 e1 08 50 db N....3.. Wt....P.
0x0060: 9b 7d 9d ac 39 14 c1 a1 7b 8d e8 d3 4c d5 02 26 .}.9... {...L.&
0x0070: b4 3a 8f 8e ca 97 21 7b e7 e5 88 a6 65 67 5b 75 .:....!{ ....eg[u
0x0080: 79 e4 63 0b bb d7 c6 af 38 2c 5d 48 d5 c9 7a 14 y.c..... 8.]H.z.
0x0090: 69 5c 35 a8 ea b5 c4 07 30 81 90 f9 7d 3e 05 06 i\5...L. 0...}>..
0x00A0: dc f6 47 c6 0a ec c7 7b 9c c4 75 74 de 7a 20 fd .G....{ ..ut.z..
0x00B0: 57 cb 05 38 af 34 30 b3 2e 47 7f 56 a1 67 69 f4 W..8.40. .G.V.gi.
0x00C0: 84 b9 4f a2 93 52 89 92 51 1d f7 69 55 12 a4 74 ..O..R.. Q..iU..t
0x00D0: b3 5a 96 bd e3 53 49 54 a8 61 91 50 46 a1 e2 f5 .Z...SIT .a.PF...
0x00E0: 79 6e 47 d9 77 dd 47 6a bc 65 40 a6 5a f3 09 f2 png.w.Gj .e@Z...
0x00F0: 06 d3 05 74 45 37 30 00 18 9c 3d 98 b1 f7 0b fe ....tE70. ...=0..g.f
0x0100: ac 08 d9 f7 a7 6f 85 30 00 3d 30 d1 d1 67 00 66 .....o.0 ..k.?v..
0x0110: 1a 5c 66 4b 08 67 33 30 3a e6 6b e3 3f 76 b7 ab .\FK.g30 :k.?v..
0x0120: 72 36 a0 61 20 d2 6c 46 2c 03 13 b2 c9 31 44 ef r6.a..lF ,...lD.
0x0130: 6e 4b 42 16 de 82 27 87 d4 a0 b3 2a d8 8b a4 nKB....'..*....
0x0030: 04 00 6b 3c 00 00 ..k<..
PS C:\Users\ondre\Desktop\xsloup02\ipk-sniffer\ipk-sniffer\bin\Release\netcoreapp3.1>

```

Zde mohu demonstrovat:

1. Filtrování podle portu jak příchozí, tak i odchozí
2. Varování uživatele na kombinaci ARP a ICMP hledání packetu v kombinaci s portem, ale díky nastavení TCP je použito filtrování pouze na TCP a port
3. Pouze dva packety byly vypsaný díky specifikaci argumentu “-n 2”

```
./ipk-sniffer -i
```



```

PS C:\Users\ondrej\Desktop\xsloup02\ipk-sniffer\ipk-sniffer\bin\Release\netcoreapp3.1> ./ipk-sniffer -i
List of all interfaces:
\Device\NPF_{40000000-0000-4CD6-8000-000000000000}:
  Description: Ndiswan Adapter
  MAC: 000000000000
  IP:
    fe80::1125:7e6b:c64e:fa88%23
    HW addr: 000000000000
  Description: Microsoft
Ethernet (\Device\NPF_{40000000-0000-4CD6-8000-000000000000}):
  MAC: 000000000000
  IP:
    fe80::d42d:cec0:ee5b:50f2%24
    2a02:8308:b001:5d00:35a9:f080:b683:7ef6
    2a02:8308:b001:5d00:924:53f:366e:b8dd
    2a02:8308:b001:5d00:d42d:cec0:ee5b:50f2
    192.168.0.213
    HW addr: 000000000000
  Description: Intel(R) Ethernet Connection (2) I218-V
vEthernet (Ethernet) (\Device\NPF_{378AAA42-C4CB-4DEA-ABE1-95AD0E2868A8}):
  MAC: 00155D5E3E3C
  IP:
    fe80::a186:e9cd:aed5:729b%35
    172.23.16.1
    HW addr: 00155D5E3E3C
  Description: Microsoft Corporation
VirtualBox Host-Only Network (\Device\NPF_{9510AD84-4A0F-45E9-B15E-957440076F31}):
  MAC: 0A0027000012
  IP:
    fe80::6102:7f27:8164:94b%18
    192.168.56.1
    HW addr: 0A0027000012
  Description: Oracle
Local Area Connection* 1 (\Device\NPF_{00000000-0000-0000-0000-00000000A76}):
  MAC: 000000000000
  IP:
    fe80::54e4:e6e0:261e:ed2d%12
    HW addr: 000000000000
  Description: Microsoft
vEthernet (WSL) (\Device\NPF_{C4853FCB-12CF-446D-BAA8-D77D7D7EA327}):
  MAC: 00155DEA803C
  IP:
    fe80::e16e:ad02:f074:8d8d%84
    172.27.32.1
    HW addr: 00155DEA803C
  Description: Microsoft Corporation
\Device\NPF_{7551C3A3-6265-4E84-942B-6382984ED612}:
  Description: Ndiswan Adapter
Local Area Connection* 2 (\Device\NPF_{00000000-0000-0000-0000-00000000A76}):
  MAC: DAF09371B993
  IP:
    fe80::11d2:9146:317d:82d%14
    HW addr: 000000000000
  Description: Microsoft
vEthernet (Default Switch) (\Device\NPF_{19AAE98B-BAB1-46E6-A1F3-D3E8C68EB876}):
  MAC: 00155D68CFAB
  IP:
    fe80::f15d:6d5c:be7:b4fc%15
    172.25.80.1
    HW addr: 00155D68CFAB
  Description: Microsoft Corporation
vEthernet (Wi-Fi) (\Device\NPF_{451EB06D-0CDF-40AF-B2C4-46B0A96D7170}):
  MAC: 00155DE2C825

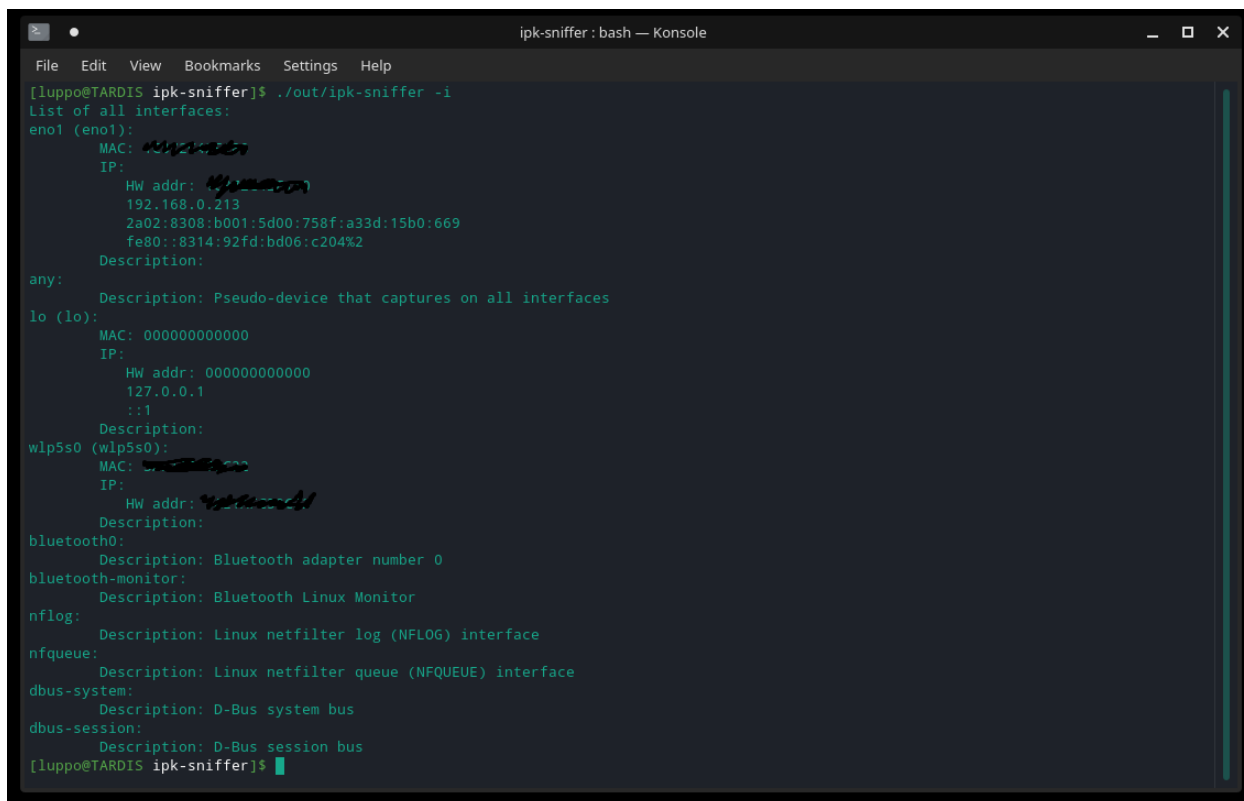
```

Takto vypisují všechna zařízení. Je zde kompletní systémové jméno (generováno z GUID) a i uživatelsky přívětivé.

## Manjaro

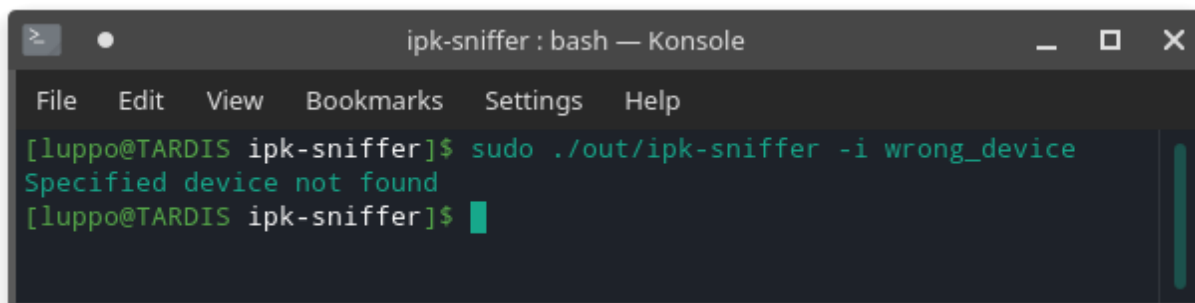
### Výpis všech zařízení

```
./ipk-sniffer -i
```



```
ipk-sniffer : bash — Konsole
File Edit View Bookmarks Settings Help
[luppo@TARDIS ipk-sniffer]$ ./out/ipk-sniffer -i
List of all interfaces:
eno1 (eno1):
  MAC: XXXXXXXXXX
  IP:
  HW addr: XXXXXXXXXX
  192.168.0.213
  2a02:8308:b001:5d00:758f:a33d:15b0:669
  fe80::8314:92fd:bd06:c204%2
  Description:
any:
  Description: Pseudo-device that captures on all interfaces
lo (lo):
  MAC: 000000000000
  IP:
  HW addr: 000000000000
  127.0.0.1
  ::1
  Description:
wlp5s0 (wlp5s0):
  MAC: XXXXXXXXXX
  IP:
  HW addr: XXXXXXXXXX
  Description:
bluetooth0:
  Description: Bluetooth adapter number 0
bluetooth-monitor:
  Description: Bluetooth Linux Monitor
nflog:
  Description: Linux netfilter log (NFLOG) interface
nfqueue:
  Description: Linux netfilter queue (NFQUEUE) interface
dbus-system:
  Description: D-Bus system bus
dbus-session:
  Description: D-Bus session bus
[luppo@TARDIS ipk-sniffer]$
```

### Příklad vypsaní špatného zařízení



```
ipk-sniffer : bash — Konsole
File Edit View Bookmarks Settings Help
[luppo@TARDIS ipk-sniffer]$ sudo ./out/ipk-sniffer -i wrong_device
Specified device not found
[luppo@TARDIS ipk-sniffer]$
```



## Výpis arp packetu

```
sudo ./out/ipk-sniffer -i eno1 -arp
```

```

ipk-sniffer : bash — Konsole
File Edit View Bookmarks Settings Help
[luppo@TARDIS ipk-sniffer]$ sudo ./out/ipk-sniffer -i eno1 --arp
Connected to eno1
ARP 2021-04-18T21:35:14.439+00:00: 192.168.0.234 > 192.168.0.234, length 60 bytes
0x0000: 34 2c c4 82 99 97 66 e5 6b 56 6b 66 08 06 00 01 4,....f. kVkf....
0x0010: 08 00 06 04 00 02 66 e5 6b 56 6b 66 c0 a8 00 ea .....f. kVkf....
0x0020: ff ff ff ff ff ff c0 a8 00 ea 00 00 00 00 00 00 .....
0x0030: 00 00 00 00 00 00 00 00 20 20 20 20 .....
[luppo@TARDIS ipk-sniffer]$

```

## Příklady špatného portu a kombinace

```

ipk-sniffer : bash — Konsole
File Edit View Bookmarks Settings Help
[luppo@TARDIS ipk-sniffer]$ sudo ./out/ipk-sniffer -i eno1 --arp -p 999999999
Specified port is not valid. It needs to be greater than 0 and lower than 65535
[luppo@TARDIS ipk-sniffer]$ sudo ./out/ipk-sniffer -i eno1 --arp -p 45
Port specification cannot be combined with ARP or ICMP argument
[luppo@TARDIS ipk-sniffer]$ sudo ./out/ipk-sniffer -i eno1 --arp -p 999999999798778
Cannot parse argument '999999999798778' for option '--port' as expected type System.Nullable`1[System.Int32].

ipk-sniffer:
  IPK Project 2: Zeta -- xsloup02

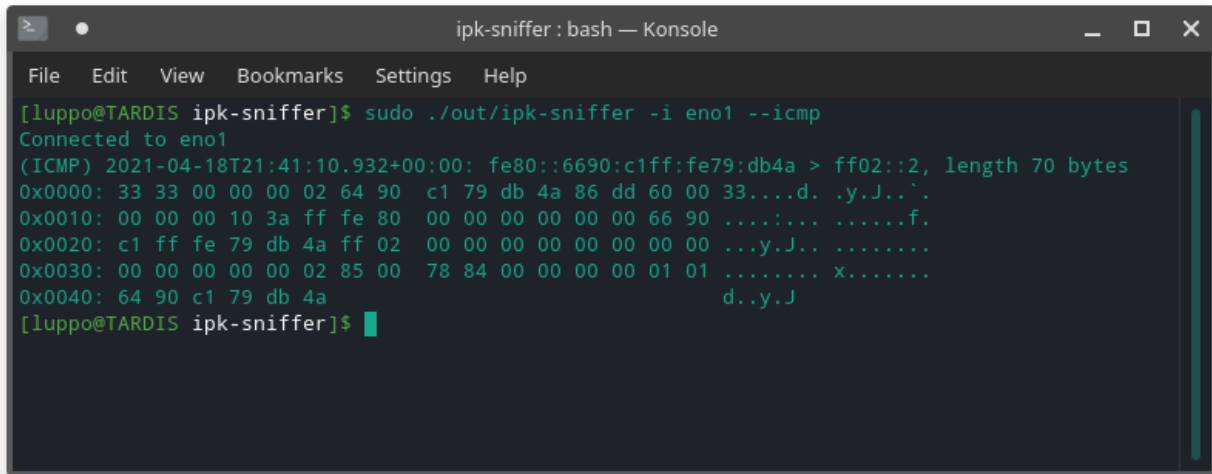
Usage:
  ipk-sniffer [options]

Options:
  -i, --interface <interface>  Interface on which packet sniffer will listen. Without optional argument prints list of interfaces
  -p, --port <port>             Specified listening port. If not specified, listen on all
  -t, --tcp                     Display TCP packets
  -u, --udp                     Display UDP packets
  --arp                         Display only ICMPv4 and ICMPv6 packets
  --icmp                         Display ARP frames
  -n <n>                         Number of packets [default: 1]
  --version                     Show version information
  -?, -h, --help               Show help and usage information

[luppo@TARDIS ipk-sniffer]$

```

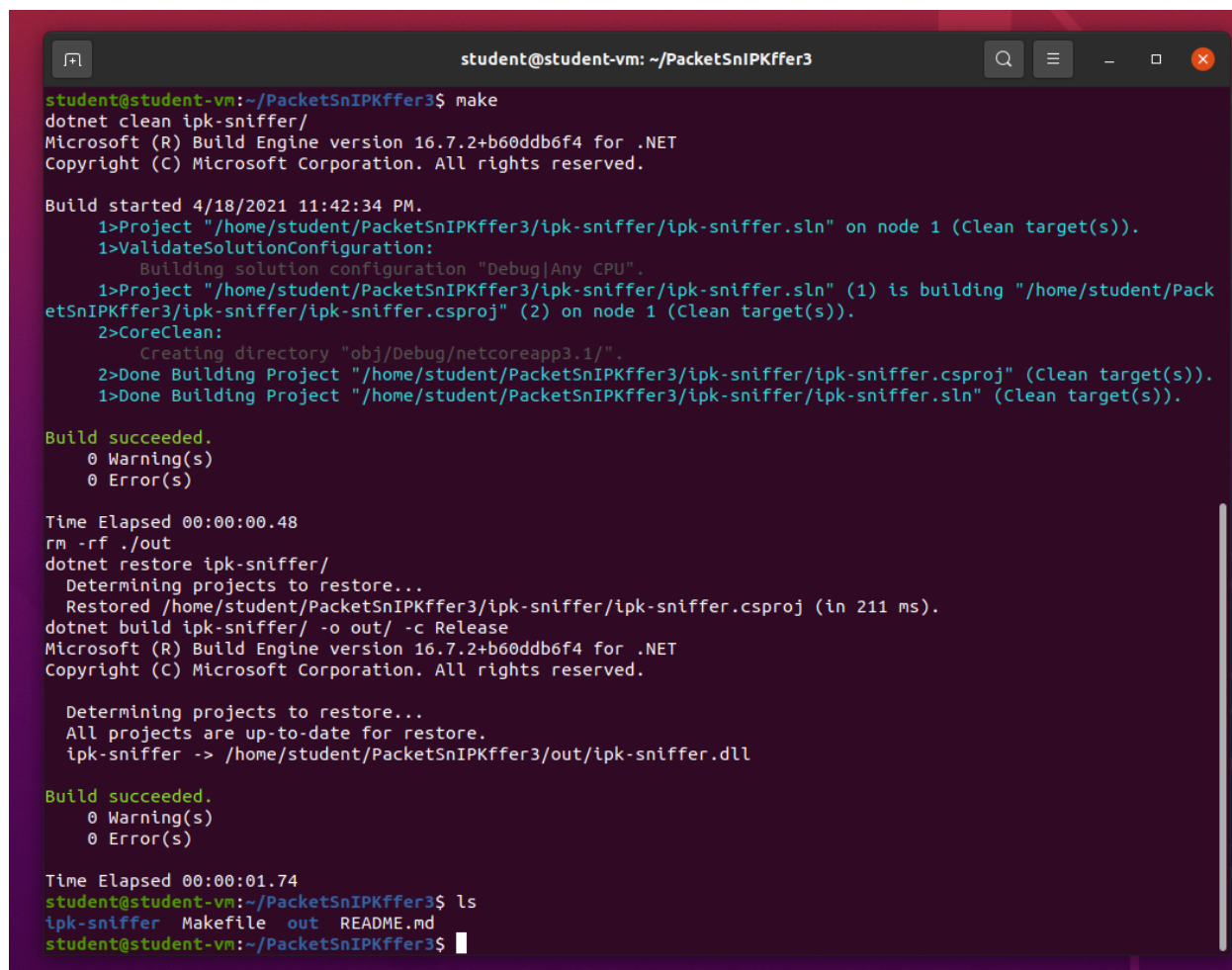
## Příklad IPv6



```
ipk-sniffer : bash — Konsole
File Edit View Bookmarks Settings Help
[luppo@TARDIS ipk-sniffer]$ sudo ./out/ipk-sniffer -i eno1 --icmp
Connected to eno1
(ICMP) 2021-04-18T21:41:10.932+00:00: fe80::6690:c1ff:fe79:db4a > ff02::2, length 70 bytes
0x0000: 33 33 00 00 00 02 64 90 c1 79 db 4a 86 dd 60 00 33...d. .y.J..`.
0x0010: 00 00 00 10 3a ff fe 80 00 00 00 00 00 00 66 90 .....f.
0x0020: c1 ff fe 79 db 4a ff 02 00 00 00 00 00 00 00 00 ...y.J.. .....
0x0030: 00 00 00 00 00 02 85 00 78 84 00 00 00 00 01 01 ..... x.....
0x0040: 64 90 c1 79 db 4a d..y.J
[luppo@TARDIS ipk-sniffer]$
```

## Ubuntu

### Make na referenčním zařízení



```
student@student-vm: ~/PacketSnIPKffer3
student@student-vm:~/PacketSnIPKffer3$ make
dotnet clean ipk-sniffer/
Microsoft (R) Build Engine version 16.7.2+b60ddb6f4 for .NET
Copyright (C) Microsoft Corporation. All rights reserved.

Build started 4/18/2021 11:42:34 PM.
  1>Project "/home/student/PackageSnIPKffer3/ipk-sniffer/ipk-sniffer.sln" on node 1 (Clean target(s)).
  1>ValidateSolutionConfiguration:
    Building solution configuration "Debug|Any CPU".
  1>Project "/home/student/PackageSnIPKffer3/ipk-sniffer/ipk-sniffer.sln" (1) is building "/home/student/PackageSnIPKffer3/ipk-sniffer/ipk-sniffer.csproj" (2) on node 1 (Clean target(s)).
  2>CoreClean:
    Creating directory "obj/Debug/netcoreapp3.1/".
  2>Done Building Project "/home/student/PackageSnIPKffer3/ipk-sniffer/ipk-sniffer.csproj" (Clean target(s)).
  1>Done Building Project "/home/student/PackageSnIPKffer3/ipk-sniffer/ipk-sniffer.sln" (Clean target(s)).

Build succeeded.
    0 Warning(s)
    0 Error(s)

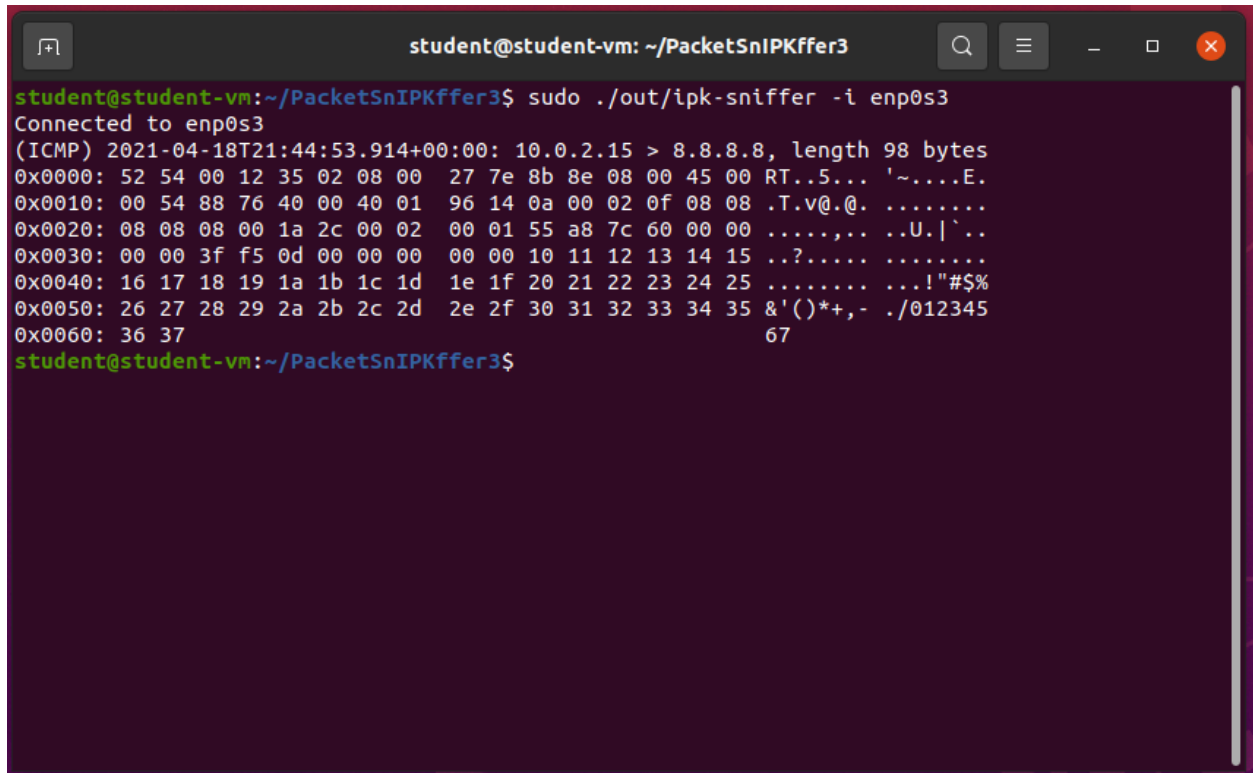
Time Elapsed 00:00:00.48
rm -rf ./out
dotnet restore ipk-sniffer/
  Determining projects to restore...
  Restored /home/student/PackageSnIPKffer3/ipk-sniffer/ipk-sniffer.csproj (in 211 ms).
dotnet build ipk-sniffer/ -o out/ -c Release
Microsoft (R) Build Engine version 16.7.2+b60ddb6f4 for .NET
Copyright (C) Microsoft Corporation. All rights reserved.

  Determining projects to restore...
  All projects are up-to-date for restore.
  ipk-sniffer -> /home/student/PackageSnIPKffer3/out/ipk-sniffer.dll

Build succeeded.
    0 Warning(s)
    0 Error(s)

Time Elapsed 00:00:01.74
student@student-vm:~/PacketSnIPKffer3$ ls
ipk-sniffer  Makefile  out  README.md
student@student-vm:~/PacketSnIPKffer3$
```

## ICMP packet

A terminal window titled 'student@student-vm: ~/PacketSnIPKffer3' with standard window controls. The prompt is 'student@student-vm:~/PacketSnIPKffer3\$'. The user enters 'sudo ./out/ipk-sniffer -i enp0s3'. The output shows 'Connected to enp0s3' followed by an ICMP packet capture line: '(ICMP) 2021-04-18T21:44:53.914+00:00: 10.0.2.15 > 8.8.8.8, length 98 bytes'. Below this is a hex dump of the packet data, with each line showing a hex address, two columns of hex bytes, and a corresponding ASCII representation. The hex dump ends at 0x0060: 36 37. The prompt returns to 'student@student-vm:~/PacketSnIPKffer3\$'.

```
student@student-vm:~/PacketSnIPKffer3$ sudo ./out/ipk-sniffer -i enp0s3
Connected to enp0s3
(ICMP) 2021-04-18T21:44:53.914+00:00: 10.0.2.15 > 8.8.8.8, length 98 bytes
0x0000: 52 54 00 12 35 02 08 00 27 7e 8b 8e 08 00 45 00 RT..5... '~....E.
0x0010: 00 54 88 76 40 00 40 01 96 14 0a 00 02 0f 08 08 .T.v@.@. ....
0x0020: 08 08 08 00 1a 2c 00 02 00 01 55 a8 7c 60 00 00 .....,.. ..U.|`..
0x0030: 00 00 3f f5 0d 00 00 00 00 00 10 11 12 13 14 15 ..?..... ....
0x0040: 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 ..... !"#$$%
0x0050: 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 &'()*+,- ./012345
0x0060: 36 37                                     67
student@student-vm:~/PacketSnIPKffer3$
```

## Reference

- [1] **Gal, Tamir a Morgan, Chris. 2014.** SharpPcap - A Packet Capture Framework for .NET. *Code Project*. [Online] 5. May 2014. [Citace: 18. Apr 2021.] <https://www.codeproject.com/Articles/12458/SharpPcap-A-Packet-Capture-Framework-for-NET>.
- [2] **Pluskal, Jan, a další. 2021.** sharppcap: GitHub repozitář. *GitHub repozitář*. [Online] 15. Apr 2021. <https://github.com/chmorgan/sharppcap>.
- [3] **Sequeira, Jon, a další. 2021.** command-line-api: Github repozitář. *Github repozitář*. [Online] 4. Apr 2021. [Citace: 18. Apr 2021.] <https://github.com/dotnet/command-line-api>.
- [4] **Wikipedia contributors. 2021.** ASCII: Printable characters. *Wikipedia*. [Online] 1017983559, 15. Apr 2021. [Citace: 18. Apr 2021.] <https://en.wikipedia.org/w/index.php?title=ASCII&oldid=1017983559>.