

# IPK – Projekt 2: Zeta

Manuál  
Ondřej Sloup

## Obsah

Volba projektu a prostředí	3
Použité knihovny	3
Použité zdroje	3
Popis programu	4
ArgumentParser	4
NetworkTools	4
Funkce	4
Návratové kódy	5
Testování	6
Příklady spuštění	6
Windows	6
Manjaro	8
Ubuntu	11
EndeavourOS	13
Reference	16

## 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, kde bylo velmi dobře vysvětlena knihovna SharpPcap a její jednotlivé funkce.

## 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 paketů.

### 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 paketů, 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
4. EndeavourOS – Linux x86\_64 5.11.15-arch1-2

Bylo testováno samotné sestavení projektu, a i jeho spuštění.

## Příklady spuštění

Citlivé údaje byly cenzurovány. Jednotlivé packety jsem simuloval pomocí příkazu **netcat** (nc), **nping** nebo jsem testoval při normálním procházení internetu.

## 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 28 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 a5 88 a6 65 67 5b 75 .....l{ ...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 4c 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: 70 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. ...=....
0x0100: ac 08 d9 f7 a7 6f 85 30 00 3d 30 d1 d1 67 00 66 .....o.0 .=#.g.f
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....' . ...*...
(TCP) 2021-04-18T20:07:32.592+00:00: 192.168.0.213 51653 > 54.64.115.100 443, length 54 bytes
0x0000: 34 2c c4 82 e1 81 08 00 08 00 45 00 4,.....,B....E.
0x0010: 00 28 ed f3 40 00 80 06 00 00 c0 a8 00 d5 36 40 .(.@... .....6@
0x0020: 73 64 c9 c5 01 bb 23 1b 2f c8 b1 26 fc 9b 50 10 sd....#. /..&..P.
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`

```

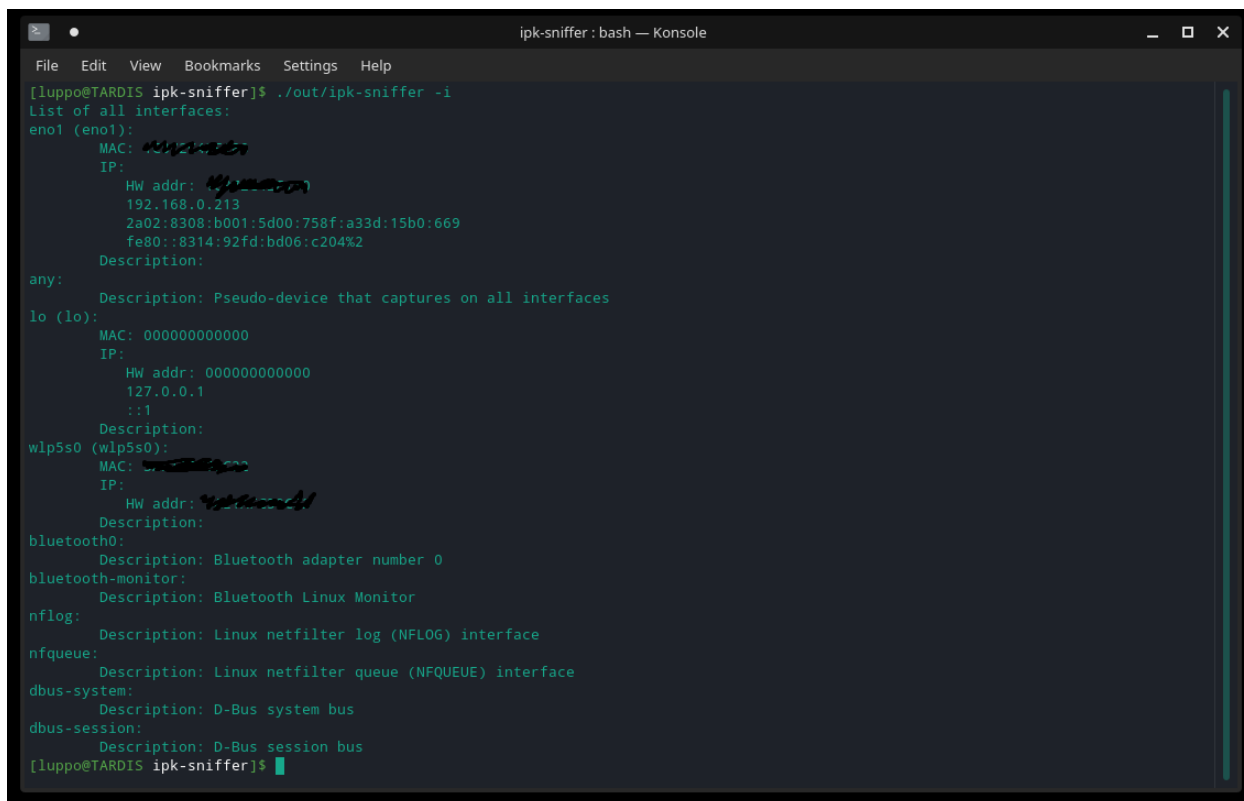
PS C:\Users\ondrej\Desktop\xsloup02\ipk-sniffer\ipk-sniffer\bin\Release\netcoreapp3.1> ./ipk-sniffer -i
List of all interfaces:
\Device\NPF_{...-4CD6-...-...-...}:
  Description: NdisWan Adapter
Wi-Fi (\Device\NPF_{...-42BF-...-50FF3486C31F}):
  MAC: ...
  IP:
    fe80::1125:7e6b:c64e:fa88%23
    HW addr: ...
  Description: Microsoft
Ethernet (\Device\NPF_{...-8D2F-453E-...-2CC2DDAA8B4}):
  MAC: ...
  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: ...
  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: ...
  Description: Oracle
Local Area Connection* 1 (\Device\NPF_{...-FEC4-...-...-...A76}):
  MAC: ...
  IP:
    fe80::54e4:e6e0:261e:ed2d%12
    HW addr: ...
  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_{751C3A3-6265-4E84-942B-6382984ED612}:
  Description: NdisWan Adapter
Local Area Connection* 2 (\Device\NPF_{...-491F-ACF7-7F1B1238C801}):
  MAC: DAF09371B993
  IP:
    fe80::11d2:9146:317d:82d%14
    HW addr: ...
  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

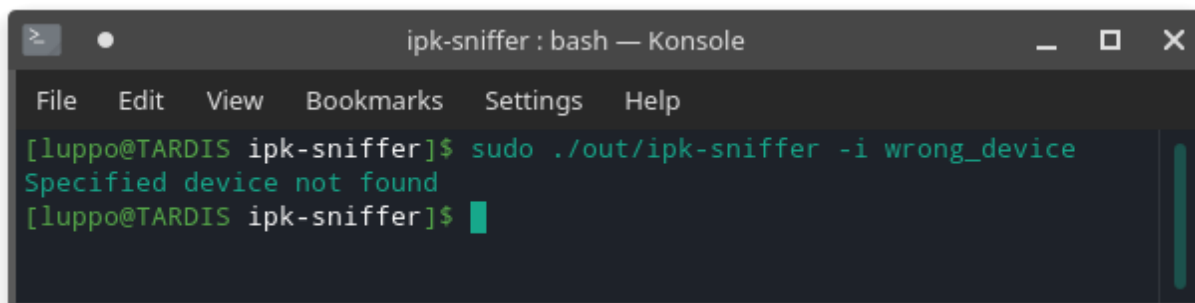
### Výpis všech zařízení

```
./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 vypsání 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]

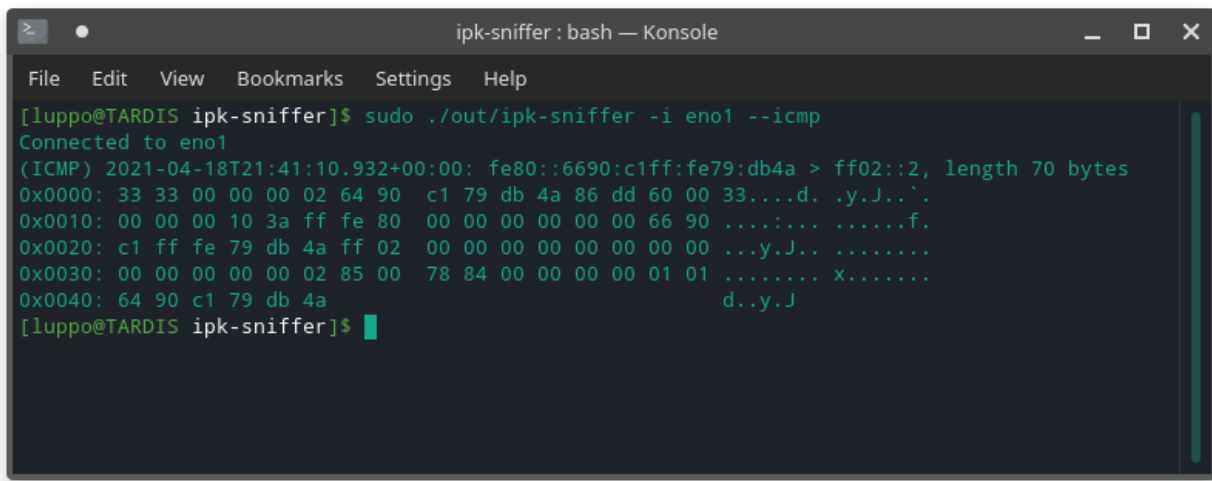
Options:
  -i, --interface <interface>  Interface on which packet sniffer will listen. Without optional argument prints list of interfaces
  -p, --port <port>             Specified listening port. If not specified, listen on all
  -t, --tcp                     Display TCP packets
  -u, --udp                     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

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

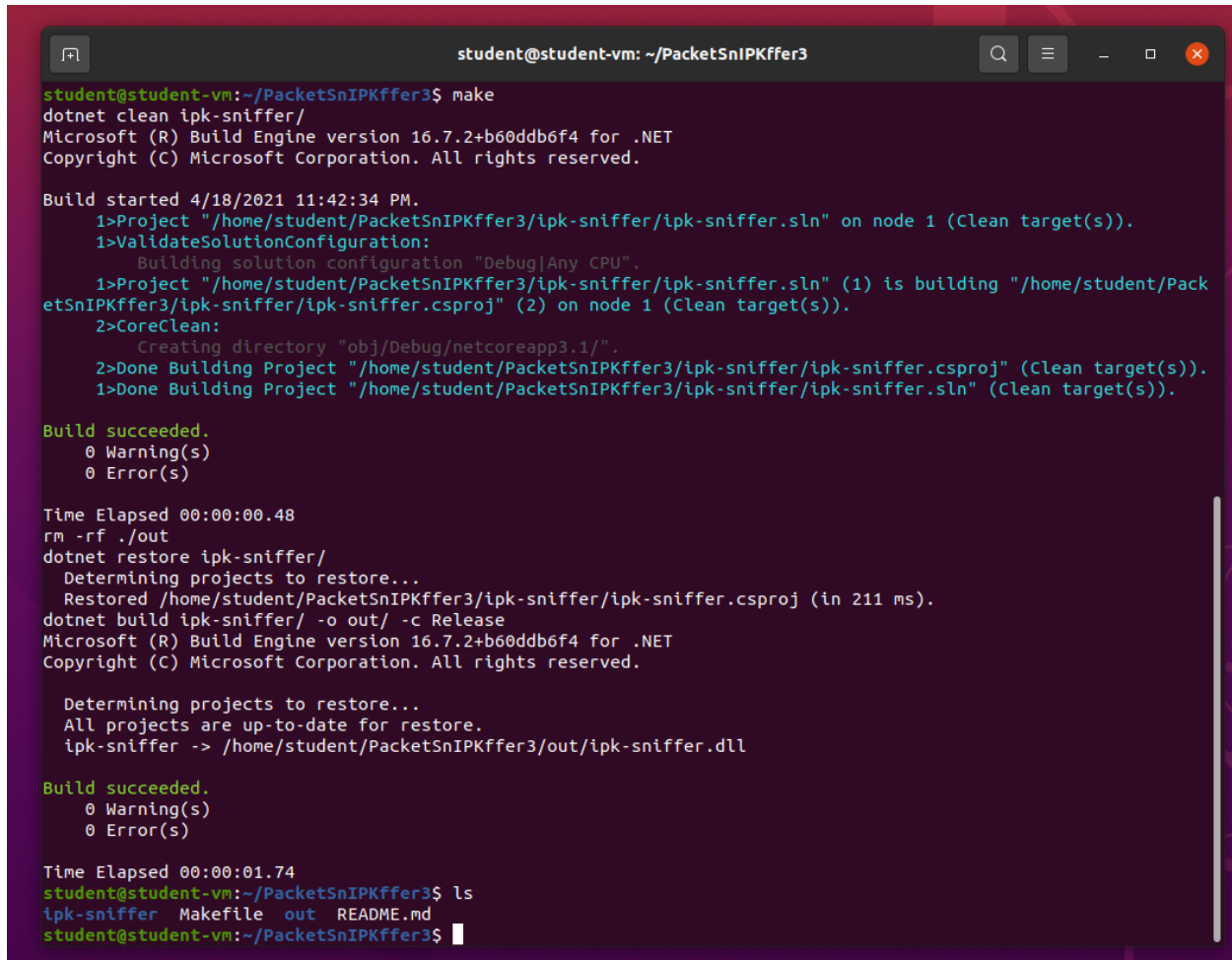


```
ipk-sniffer : bash — Konsole
File Edit View Bookmarks Settings Help
[luppo@TARDIS ipk-sniffer]$ sudo ./out/ipk-sniffer -i eno1 --icmp
Connected to eno1
(ICMP) 2021-04-18T21:41:10.932+00:00: fe80::6690:c1ff:fe79:db4a > ff02::2, length 70 bytes
0x0000: 33 33 00 00 00 02 64 90 c1 79 db 4a 86 dd 60 00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

make

A terminal window titled 'student@student-vm: ~/PacketSnIPKffer3' showing the execution of 'make'. The terminal output includes the following text:

```
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.

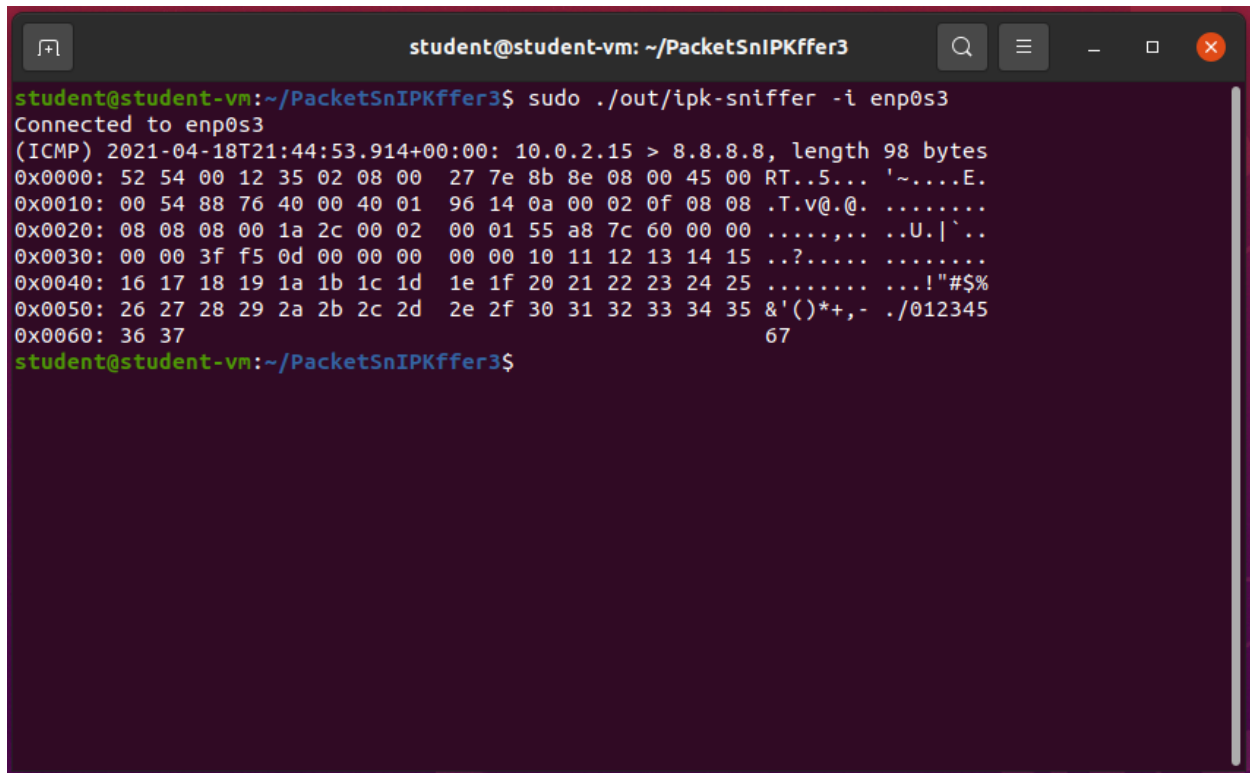
  Determining projects to restore...
  All projects are up-to-date for restore.
  ipk-sniffer -> /home/student/PackageSnIPKffer3/out/ipk-sniffer.dll

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

```
sudo ./out/ipk-sniffer -i enp0s3
```

A terminal window titled 'student@student-vm: ~/PacketSnIPKffer3' with standard window controls. The prompt is 'student@student-vm:~/PacketSnIPKffer3\$'. The user has entered '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, 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$
```

## EndeavourOS

### Vypsání několika UDP packetů včetně IPv6 (stress testing)

```
sudo ./out/ipk-sniffer -i eno1 --udp -n 9999999
```

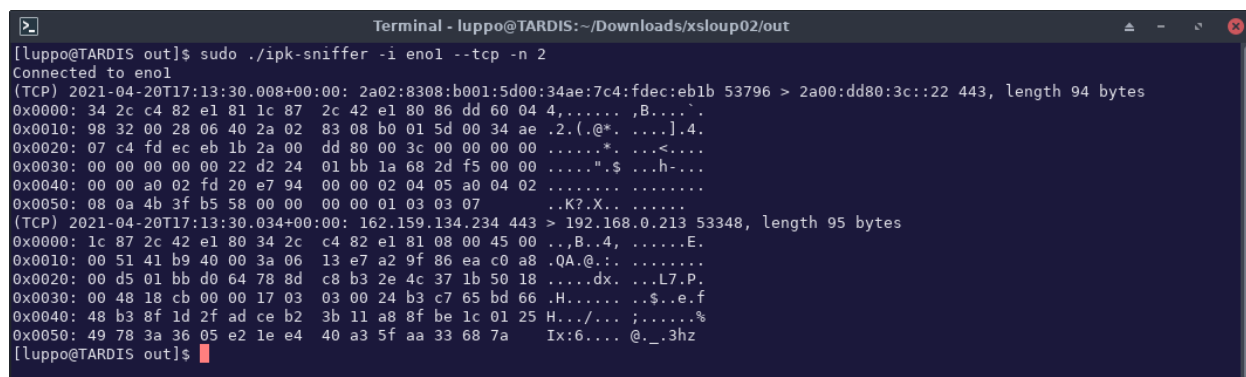
```

Terminal - luppo@TARDIS:~/Downloads/xsloup02/out
^C
[luppo@TARDIS out]$ sudo ./ipk-sniffer -i eno1 --udp -n 9999999
Connected to eno1
(UDP) 2021-04-20T17:09:26.633+00:00: 192.168.0.67 137 > 192.168.0.255 137, length 92 bytes
0x0000: ff ff ff ff ff ff 50 e5 49 52 66 ba 08 00 45 00 .....P. IRf...E.
0x0010: 00 4e 80 59 00 00 80 11 37 b3 c0 a8 00 43 c0 a8 .N.Y.... 7....C..
0x0020: 00 ff 00 89 00 89 00 3a 64 d2 d5 94 01 10 00 01 .....: d.....
0x0030: 00 00 00 00 00 00 20 46 48 46 41 45 42 45 45 43 .....F HFAEBEEC
0x0040: 41 43 41 43 41 43 41 43 41 43 41 43 41 43 41 43 ACACACAC ACACACAC
0x0050: 41 43 41 43 41 41 41 00 00 20 00 01 ACACAAA. ....
(UDP) 2021-04-20T17:09:26.633+00:00: 192.168.0.67 5353 > 224.0.0.251 5353, length 70 bytes
0x0000: 01 00 5e 00 00 fb 50 e5 49 52 66 ba 08 00 45 00 ..^...P. IRf...E.
0x0010: 00 38 0d d3 00 00 01 11 09 fc c0 a8 00 43 e0 00 .8..... ..C..
0x0020: 00 fb 14 e9 14 e9 00 24 1f 3b 00 00 00 00 00 01 .....$ ;.....
0x0030: 00 00 00 00 00 00 04 77 70 61 64 05 6c 6f 63 61 .....w pad.loca
0x0040: 6c 00 00 01 00 01 l.....
(UDP) 2021-04-20T17:09:26.633+00:00: fe80::3915:60c2:bf24:245b 5353 > ff02::fb 5353, length 90 bytes
0x0000: 33 33 00 00 00 fb 50 e5 49 52 66 ba 86 dd 60 06 33....P. IRf...`
0x0010: ee b6 00 24 11 01 fe 80 00 00 00 00 00 39 15 ...$. ....9.
0x0020: 60 c2 bf 24 24 5b ff 02 00 00 00 00 00 00 00 00 `..$$[. ....
0x0030: 00 00 00 00 00 fb 14 e9 14 e9 00 24 45 4c 00 00 ..... ..$EL..
0x0040: 00 00 00 01 00 00 00 00 00 00 04 77 70 61 64 05 ..... ..wpad.
0x0050: 6c 6f 63 61 6c 00 00 01 00 01 local... ..
(UDP) 2021-04-20T17:09:26.634+00:00: 192.168.0.67 5353 > 224.0.0.251 5353, length 70 bytes
0x0000: 01 00 5e 00 00 fb 50 e5 49 52 66 ba 08 00 45 00 ..^...P. IRf...E.
0x0010: 00 38 0d d4 00 00 01 11 09 fb c0 a8 00 43 e0 00 .8..... ..C..
0x0020: 00 fb 14 e9 14 e9 00 24 1f 20 00 00 00 00 00 01 .....$ .....
0x0030: 00 00 00 00 00 00 04 77 70 61 64 05 6c 6f 63 61 .....w pad.loca
0x0040: 6c 00 00 1c 00 01 l.....
(UDP) 2021-04-20T17:09:26.634+00:00: fe80::3915:60c2:bf24:245b 5353 > ff02::fb 5353, length 90 bytes
0x0000: 33 33 00 00 00 fb 50 e5 49 52 66 ba 86 dd 60 06 33....P. IRf...`
0x0010: ee b6 00 24 11 01 fe 80 00 00 00 00 00 39 15 ...$. ....9.
0x0020: 60 c2 bf 24 24 5b ff 02 00 00 00 00 00 00 00 00 `..$$[. ....
0x0030: 00 00 00 00 00 fb 14 e9 14 e9 00 24 45 31 00 00 ..... ..$EL..
0x0040: 00 00 00 01 00 00 00 00 00 00 04 77 70 61 64 05 ..... ..wpad.
0x0050: 6c 6f 63 61 6c 00 00 1c 00 01 local... ..
(UDP) 2021-04-20T17:09:26.635+00:00: fe80::3915:60c2:bf24:245b 53415 > ff02::1:3 5355, length 84 bytes
0x0000: 33 33 00 01 00 03 50 e5 49 52 66 ba 86 dd 60 01 33....P. IRf...`
0x0010: c4 8d 00 1e 11 01 fe 80 00 00 00 00 00 39 15 ..... ..9.
0x0020: 60 c2 bf 24 24 5b ff 02 00 00 00 00 00 00 00 00 `..$$[. ....
0x0030: 00 00 00 01 00 03 d0 a7 14 eb 00 1e cb 31 fb 32 ..... ..1.2
0x0040: 00 00 00 01 00 00 00 00 00 00 04 77 70 61 64 00 ..... ..wpad.
0x0050: 00 01 00 01 ....
(UDP) 2021-04-20T17:09:26.635+00:00: 192.168.0.67 53415 > 224.0.0.252 5355, length 64 bytes
0x0000: 01 00 5e 00 00 fc 50 e5 49 52 66 ba 08 00 45 00 ..^...P. IRf...E.
0x0010: 00 32 fb 7c 00 00 01 11 1c 57 c0 a8 00 43 e0 00 .2.|.... .W...C..
0x0020: 00 fc d0 a7 14 eb 00 1e a4 28 fb 32 00 00 00 01 ..... ..(.2....
0x0030: 00 00 00 00 00 00 04 77 70 61 64 00 00 01 00 01 .....w pad....
(UDP) 2021-04-20T17:09:26.635+00:00: fe80::3915:60c2:bf24:245b 62734 > ff02::1:3 5355, length 84 bytes
0x0000: 33 33 00 01 00 03 50 e5 49 52 66 ba 86 dd 60 01 33....P. IRf...`
0x0010: 1b 00 00 1e 11 01 fe 80 00 00 00 00 00 39 15 ..... ..9.
0x0020: 60 c2 bf 24 24 5b ff 02 00 00 00 00 00 00 00 00 `..$$[. ....
0x0030: 00 00 00 01 00 03 f5 0e 14 eb 00 1e d2 4e cf 93 ..... ..N..
0x0040: 00 00 00 01 00 00 00 00 00 00 04 77 70 61 64 00 ..... ..wpad.
0x0050: 00 1c 00 01 ....

```

Vypsání několika TCP packetů včetně IPv6 (stress testing)

`sudo ./out/ipk-sniffer -i eno1 --tcp -n 2`



```

Terminal - luppo@TARDIS:~/Downloads/xsloup02/out
[luppo@TARDIS out]$ sudo ./ipk-sniffer -i eno1 --tcp -n 2
Connected to eno1
(TCP) 2021-04-20T17:13:30.008+00:00: 2a02:8308:b001:5d00:34ae:7c4:fdec:eb1b 53796 > 2a00:dd80:3c::22 443, length 94 bytes
0x0000: 34 2c c4 82 e1 81 1c 87 2c 42 e1 80 86 dd 60 04 4,.....,B....`
0x0010: 98 32 00 28 06 40 2a 02 83 08 b0 01 5d 00 34 ae .2.(. @*. ....].4.
0x0020: 07 c4 fd ec eb 1b 2a 00 dd 80 00 3c 00 00 00 00 .....*. ...<...
0x0030: 00 00 00 00 00 22 d2 24 01 bb 1a 68 2d f5 00 00 .....". $ ...h-...
0x0040: 00 00 a0 02 fd 20 e7 94 00 00 02 04 05 a0 04 02 .....
0x0050: 08 0a 4b 3f b5 58 00 00 00 00 01 03 03 07 ..K?.X. ....
(TCP) 2021-04-20T17:13:30.034+00:00: 162.159.134.234 443 > 192.168.0.213 53348, length 95 bytes
0x0000: 1c 87 2c 42 e1 80 34 2c c4 82 e1 81 08 00 45 00 ..,B..4, .....E.
0x0010: 00 51 41 b9 40 00 3a 06 13 e7 a2 9f 86 ea c0 a8 .QA.@.:. ....
0x0020: 00 d5 01 bb d0 64 78 8d c8 b3 2e 4c 37 1b 50 18 ....dx. ...L7.P.
0x0030: 00 48 18 cb 00 00 17 03 03 00 24 b3 c7 65 bd 66 .H..... ..$.e.f
0x0040: 48 b3 8f 1d 2f ad ce b2 3b 11 a8 8f be 1c 01 25 H.../... ;.....%
0x0050: 49 78 3a 36 05 e2 1e e4 40 a3 5f aa 33 68 7a Ix:6.... @._.3hz
[luppo@TARDIS out]$

```

## Kontrola dat

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

The image shows a Windows PowerShell window and two Wireshark packet capture windows. The PowerShell window displays the output of the `./ipk-sniffer` command, showing captured network packets with their details and hex data. The first Wireshark window shows a packet capture on the 'Ethernet' interface, with a packet list table and a detailed view of a TCP RST packet. The second Wireshark window shows a similar packet capture, but with a different packet selected for the detailed view.

**Windows PowerShell Output:**

```
PS C:\Users\ondre\Desktop\xsloup02\ipk-sniffer\bin\Release\netcoreapp3.1> ./ipk-sniffer -i "Ethernet" -n 2 -p 443 -t
Connected to \Device\NPF_{F1171CC8-8D2F-453E-9B61-2CC2DDAA8B4}
(TCP) 2021-04-20T15:33:44.740+00:00: 2603:1026:c0d:806::2 443 > 2a02:8308:b001:5d00:652d:d8f3:f96d:dfcf 55556, length 74 bytes
0x0000: 1c 87 2c 42 e1 80 34 2c c4 82 e1 81 86 dd 60 00 ...B..4, .....
0x0010: 00 00 00 14 06 ec 26 03 10 26 0c 0d 08 06 00 00 .....& .&.....
0x0020: 00 00 00 00 02 2a 02 83 08 b0 01 5d 00 65 2d .....*. ....].e-
0x0030: d8 f3 f9 6d df cf 01 bb d9 04 1d d9 24 5a a6 46 ...m.....$Z.F
0x0040: c7 8a 50 14 00 00 09 63 00 00 ..P....c ..
(TCP) 2021-04-20T15:33:45.219+00:00: 2620:1ec:8f8::254 443 > 2a02:8308:b001:5d00:652d:d8f3:f96d:dfcf 55558, length 74 bytes
0x0000: 1c 87 2c 42 e1 80 34 2c c4 82 e1 81 86 dd 60 07 ...B..4, .....
0x0010: e7 ff 00 14 06 32 26 20 01 ec 08 f8 00 00 00 00 .....2& .&.....
0x0020: 00 00 00 00 02 54 2a 02 83 08 b0 01 5d 00 65 2d .....T*.....].e-
0x0030: d8 f3 f9 6d df cf 01 bb d9 06 6e 50 67 9b 21 28 ...m.....nPg.!(
0x0040: 9d fc 50 14 00 00 3b 3b 00 00 ..P....;; ..
PS C:\Users\ondre\Desktop\xsloup02\ipk-sniffer\bin\Release\netcoreapp3.1>
```

**Wireshark Packet Capture (Ethernet):**

No.	Time	Source	Destination	Protocol	Length	Info
70	10.435924	13.107.42.254	192.168.0.213	TCP	60	443 → 55559 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
23	5.161727	192.168.0.213	52.113.199.109	TCP	66	55572 → 443 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 WS=256
24	5.195276	52.113.199.109	192.168.0.213	TCP	66	443 → 55572 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=
47	6.966472	2603:1026:c0d:806::2	2a02:8308:b001:5d00...	TCP	74	443 → 55556 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
48	7.445107	2620:1ec:8f8::254	2a02:8308:b001:5d00...	TCP	74	443 → 55558 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
50	7.555819	2620:1ec:c11::200	2a02:8308:b001:5d00...	TCP	74	443 → 52339 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
13	2.562149	fe80::362c:c4ff:fe8...	fe80::d42d:cec0:ee5...	ICMPv6	78	Neighbor Advertisement fe80::362c:c4ff:fe82:e181 (rtr,
17	4.710454	192.168.0.213	155.133.250.146	UDP	78	50461 → 77035 Len=36

**Frame 47: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface \Device\NPF\_{F1171CC8-8D2F-453E-9B61-2CC2DDAA8B4}, id 0**

**Ethernet II, Src: CompalBr\_82:e1:81 (34:2c:c4:82:e1:81), Dst: ASUSTekC\_42:e1:80 (1c:87:2c:42:e1:80)**

**Internet Protocol Version 6, Src: 2603:1026:c0d:806::2, Dst: 2a02:8308:b001:5d00:652d:d8f3:f96d:dfcf**

**Transmission Control Protocol, Src Port: 443, Dst Port: 55556, Seq: 1, Ack: 1, Len: 0**

**Frame 48: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface \Device\NPF\_{F1171CC8-8D2F-453E-9B61-2CC2DDAA8B4}, id 0**

**Ethernet II, Src: CompalBr\_82:e1:81 (34:2c:c4:82:e1:81), Dst: ASUSTekC\_42:e1:80 (1c:87:2c:42:e1:80)**

**Internet Protocol Version 6, Src: 2620:1ec:8f8::254, Dst: 2a02:8308:b001:5d00:652d:d8f3:f96d:dfcf**

**Transmission Control Protocol, Src Port: 443, Dst Port: 55558, Seq: 1, Ack: 1, Len: 0**

**Wireshark EthernetGFX610.pcapng** | Packets: 77 · Displayed: 77 (100.0%) · Dropped: 0 (0.0%) | Profile: Default

## 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>.