IPK – Projekt 2: Zeta

Manuál

Ondřej Sloup

# 

Obsah

[Volba projektu a prostředí 3](#_Toc69834822)

[Použité knihovny 3](#_Toc69834823)

[Použité zdroje 3](#_Toc69834824)

[Popis programu 4](#_Toc69834825)

[ArgumentParser 4](#_Toc69834826)

[NetworkTools 4](#_Toc69834827)

[Funkce 4](#_Toc69834828)

[Návratové kódy 5](#_Toc69834829)

[Testování 6](#_Toc69834830)

[Příklady spuštění 6](#_Toc69834831)

[Windows 6](#_Toc69834832)

[Manjaro 8](#_Toc69834833)

[Ubuntu 11](#_Toc69834834)

[EndeavourOS 13](#_Toc69834835)

[Reference 16](#_Toc69834836)

# 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](https://github.com/dotnet/command-line-api) – .NET knihovna pro parsování argumentů a dynamického generování outputu
* [SharpPcap](https://github.com/chmorgan/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 zajištují funkčnost snifferu a listování zařízení.

## ArgumentParser

Třída pracuje s knihovnou [System.CommandLine](#_Použité_knihovny), 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 zařízení.

Klíčovým prvkem je knihovna [SharpPcap](#_Použité_knihovny), 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 vypsaných 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 vypsat 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

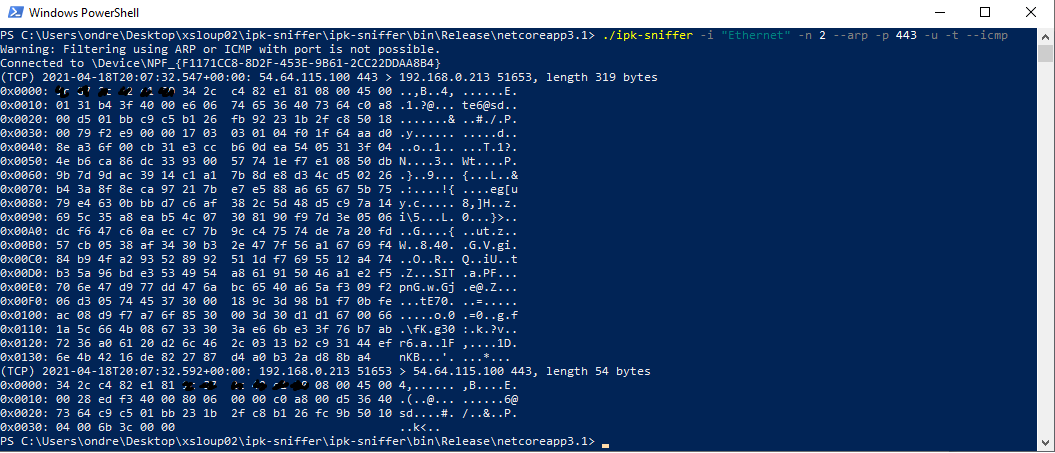
Bylo testováno samotné sestavení projektu, a i jeho spuštění.

## Příklady spuštění

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

### Windows

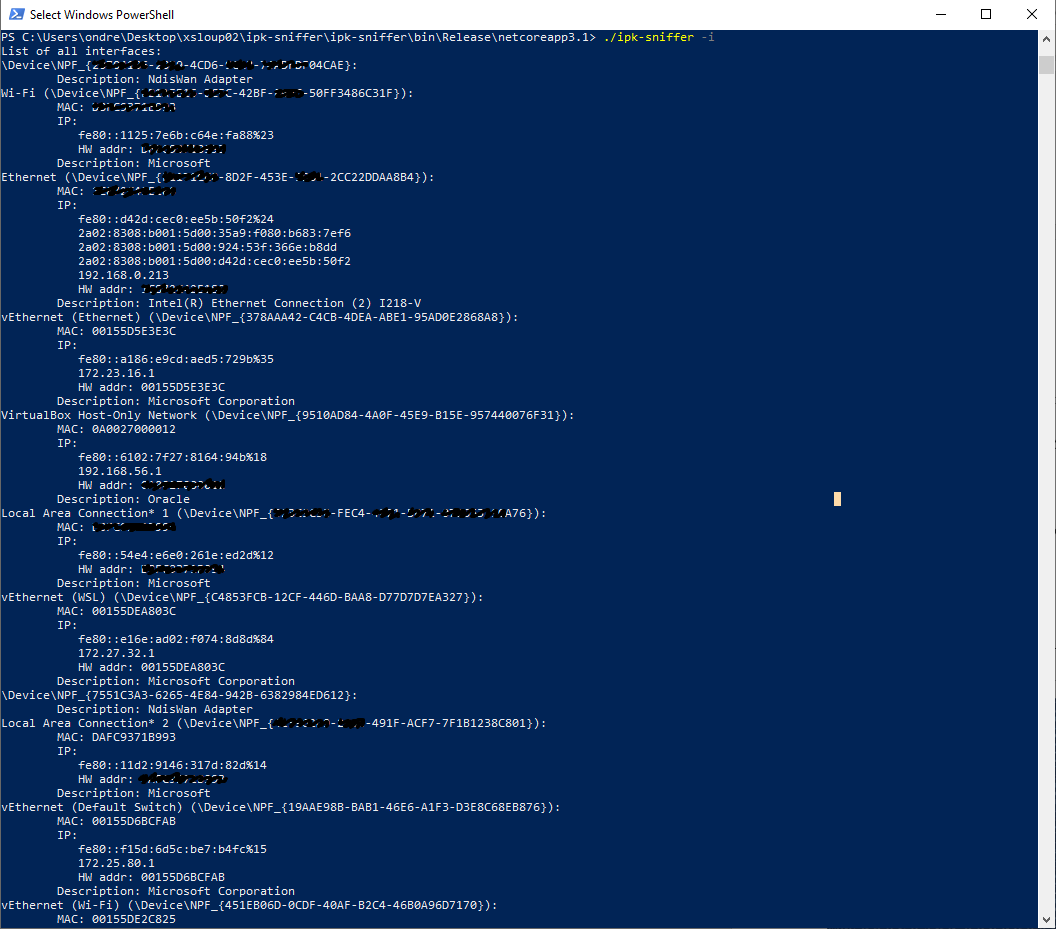
./ipk-sniffer -i "Ethernet" -n 2 --arp -p 443 -u -t –icmp



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 vypsány díky specifikaci argumentu “-n 2”

./ipk-sniffer -i

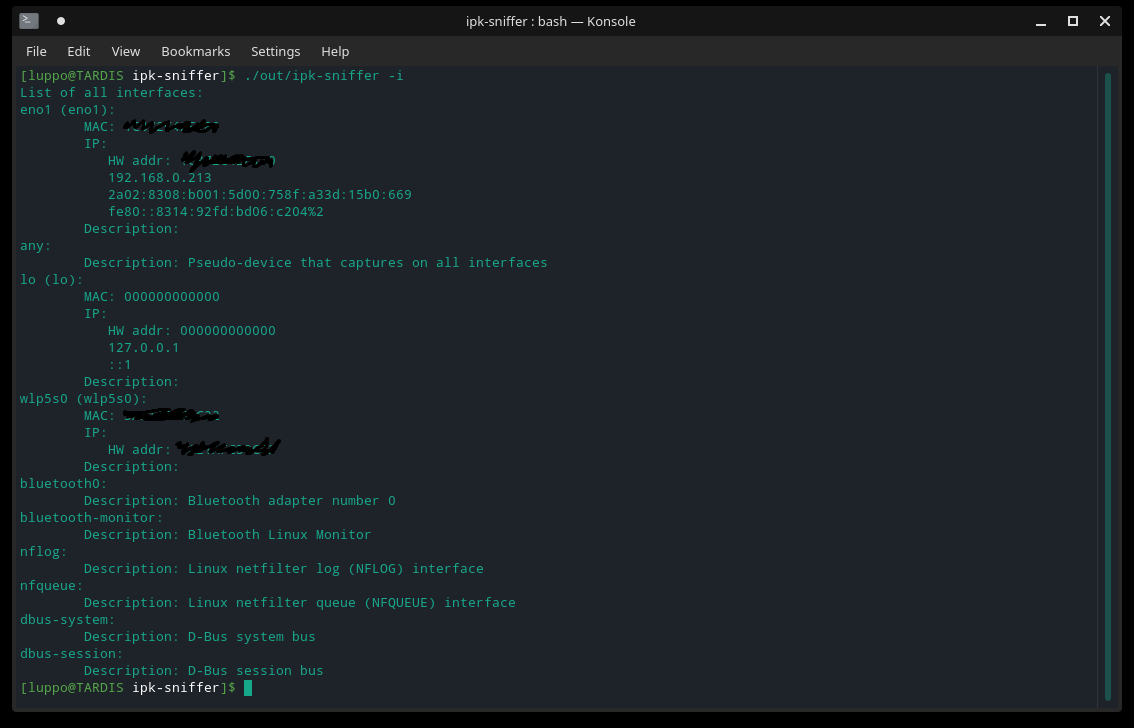


Takto vypisuji všechna zařízení. Je zde kompletní systémové jméno (generováno z GUID) a i uživatelsky přívětivé.

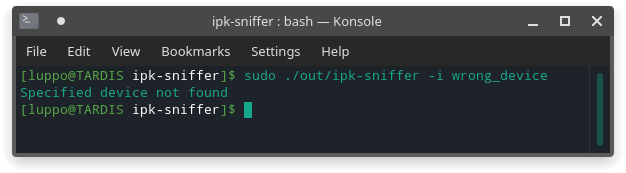
### Manjaro

#### Výpis všech zařízení

./ipk-sniffer -i

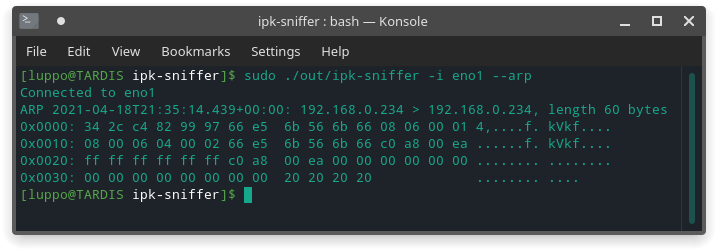


#### Příklad vypsání špatného zařízení

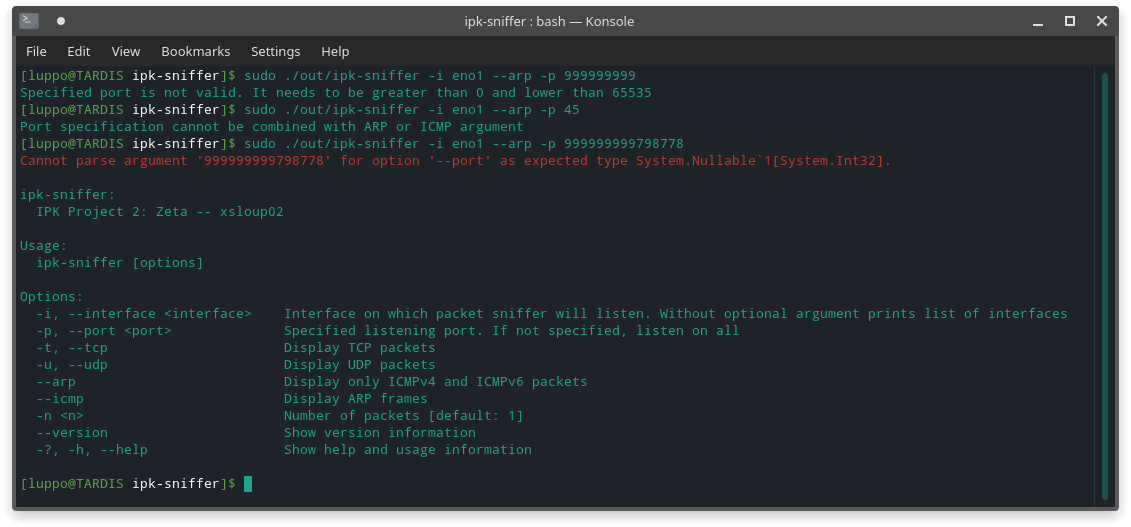


#### Výpis arp packetu

sudo ./out/ipk-sniffer -i eno1 –-arp

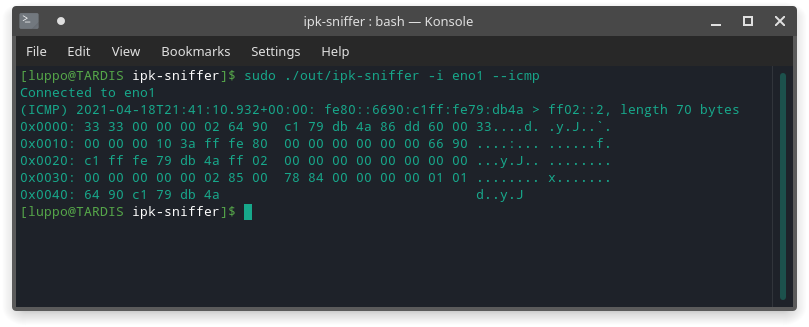


#### Příklady špatného portu a kombinace­



#### Příklad IPv6

sudo ./out/ipk-sniffer -i eno1 –-icmp



### Ubuntu

#### Make na referenčním zařízení

make

#### 

#### ICMP packet

sudo ./out/ipk-sniffer -i enp0s3

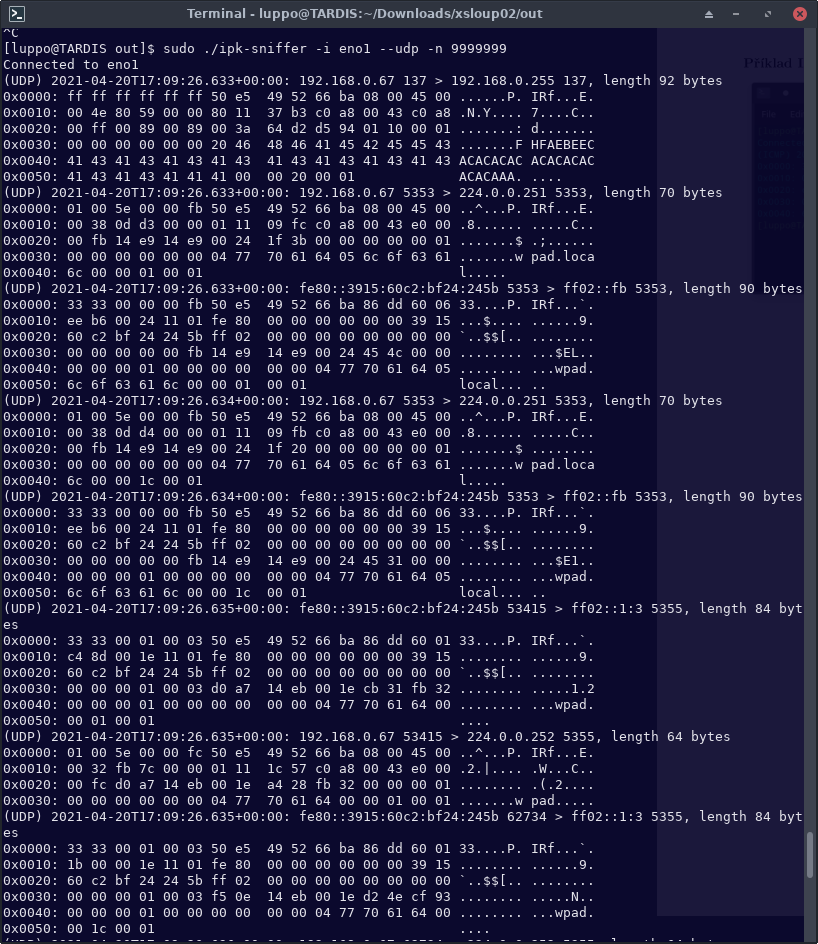
#### 

#### 

### EndeavourOS

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

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

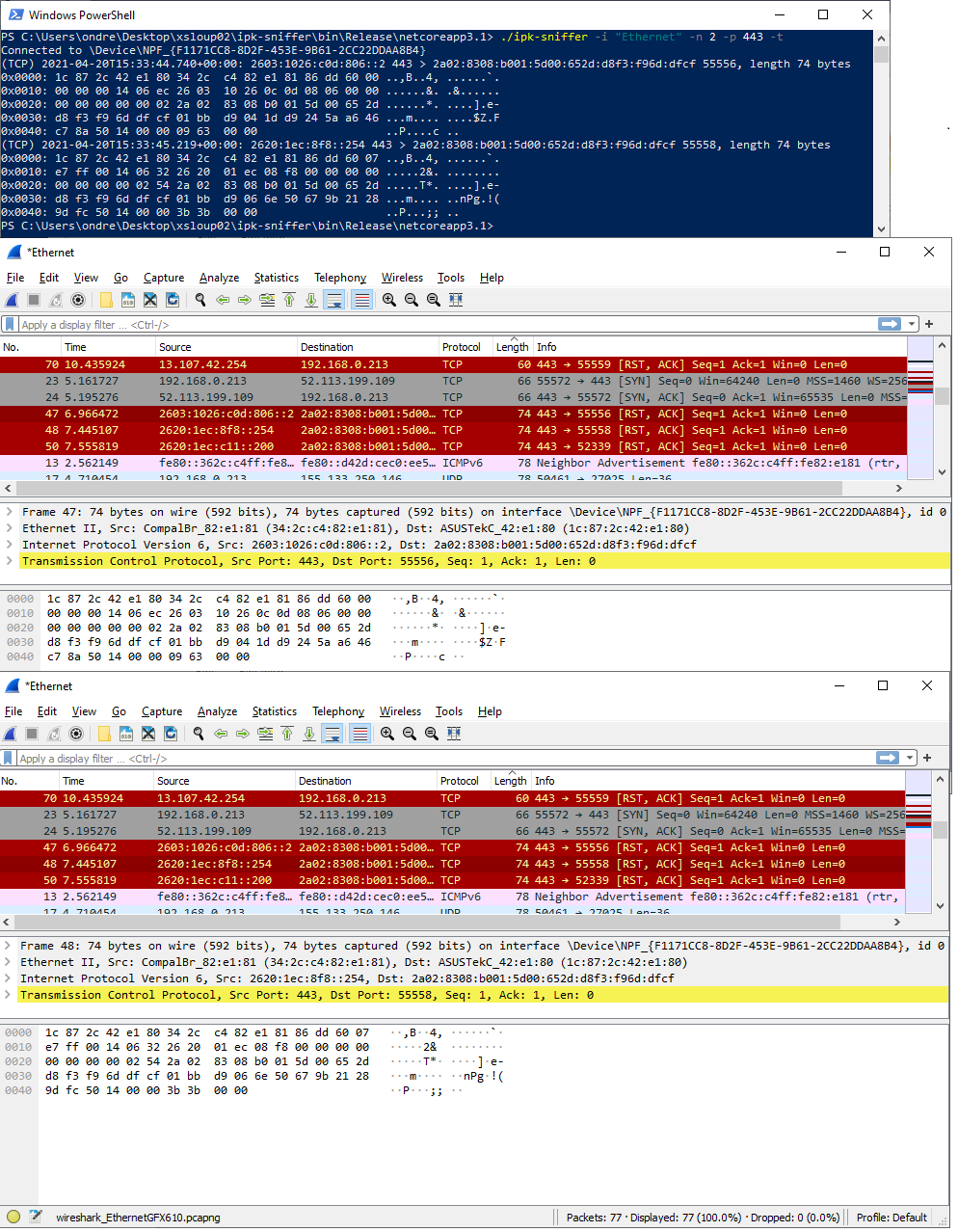


#### Vypsaní několika TCP packetů včetně IPv6 (stress testing)

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

#### 

#### Kontrola dat

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

# 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.