VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

IPK – Počítačové komunikace a sítě

Implementační dokumentace k 2. projektu
Varianta ZETA - Sniffer paketů

28. dubna 2020

Michal Dostál (xdosta51)

Obsah

1 Úvod

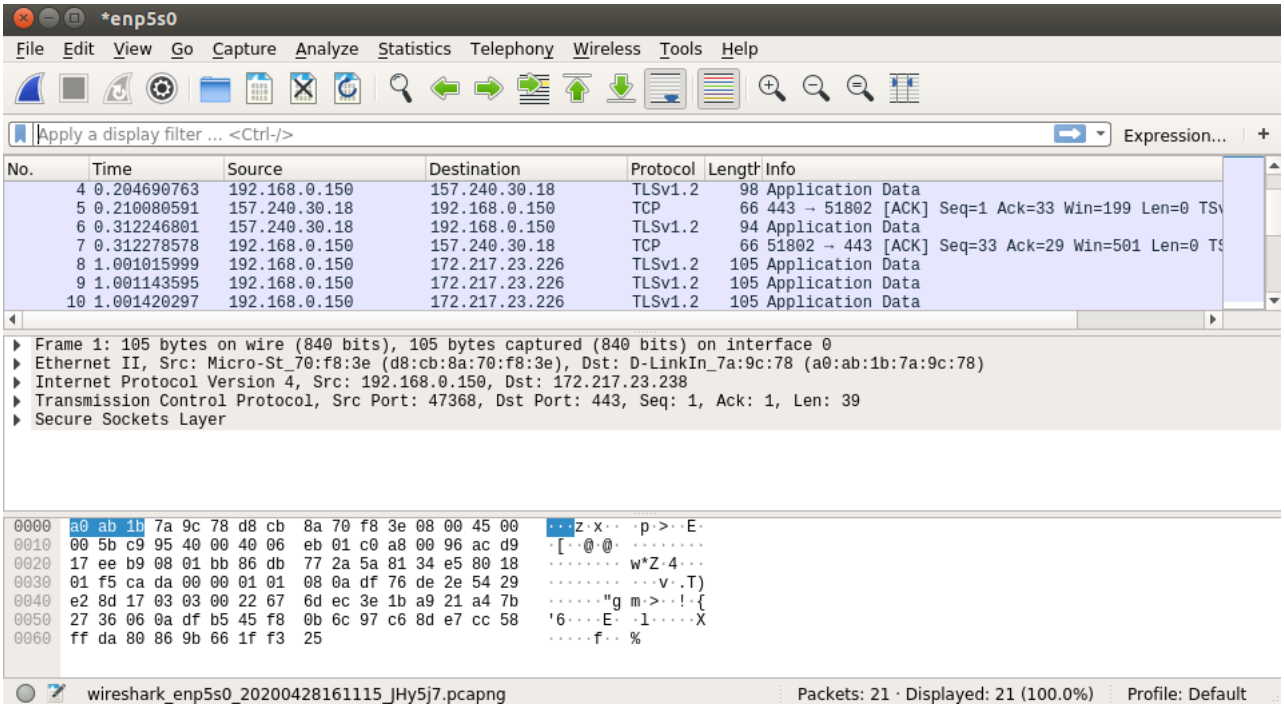
Cílem tohoto projektu bylo navrhnout a implementovat síťový analyzátor, který bude schopný na síťovém rozhraní zachytávat pakety. Tato aplikace je implemetována pomocí jazyku c#. Jelikož program pracuje se síťovými rozrahními, je potřeba jej spouštět s administrátorskými právy.

2 Nastudované informace

K implementaci tohoto analyzátoru bylo potřeba předem nastudovat několik důležitých informací, které vedly k úspěšnému výsledku.

2.1 Wireshark

Před návrhem samotného projektu jsem se důkladně seznámil s nástrojem Wireshark, který mi byl po celou dobu programování oporou. Po instalaci tohoto nástroje jsem studoval dokumentaci, která je volně dostupná a můžete ji nalézt třeba zde: [6].



Obrázek 1: Odchycení paketu v aplikaci Wireshark

2.2 TCP paket a UDP paket

Další důležitou součástí projektu bylo nastudovat strukturu jednotlivých paketů. UDP paket je jednoduchý, vypovídá to o něm už samotný popis jeho standardu viz [4]. TCP paket už je o dost složitější a k jeho pochopení je potřeba se nad ním i trochu zamyslet, díky čemuž je jeho standard lehce složitější viz [5].

2.3 SharpPcap

Na závěr literatury uvádím knihovnu, kterou jsem studoval od začátku implementace mého projektu až do jeho konce. Jedná se o knihovnu pro jazyk c#, pomocí které jsem implementoval zachycení paketů viz [2].

3 Popis implementace

Program je implementován v jedinném souboru, jelikož nebylo nutné rozdělovat tuto aplikaci do modulů. Po zpracování argumentů příkazové řádky začne provádění samotného programu, které na daném síťovém rozhraní filtruje a zachytává pakety. Zachytávání paketů probíhá v nekonečné smyčce, kterou je možno ukončit SIGINT signálem.

3.1 Volba jazyka

Jazyk jsem zvolil c# z důvodu toho, že je nejjednodušší na implementaci složitějších struktur a pracování s nimi. Program není psaný objektově orientovaně. Jeho rozsáhlost mi nepřišla pro tuto volbu dostačující.

3.2 Knihovny

K implementaci projektu jsem se rozhodl použít volně dostupnou knihovnu SharpPcap viz [2], která je kompatibilní s knihovnou libpcap. Inspiroval jsem se z příkladu v SharpPcap GIT repozitáři, kde je úkazka jednoduchého programu na odchycení paketu a následné vypsání jeho času a délky viz [3].

```
// otevre zarizeni pro naslouchani
int readTimeoutMilliseconds = 1000;
device.Open(DeviceMode.Promiscuous, readTimeoutMilliseconds);

/* promenna pro paket */
RawCapture packet;

/*promenna pro vypis pozadovaneho poctu paketu */
int paketky = 0;

/*cyklem prochazime paketu po paketu ze zvoleneho zarizeni */
while ((packet = device.GetNextPacket()) != null)
{
```

Obrázek 2: Ukázka kódu použitého z příkladu.

3.3 Zpracování paketu

Velice důležitou součástí projektu je analýza paketu na jeho části. Analýza paketu není prováděna ručně, ale jsou zde použité struktury z knihovny PacketDotNet, které většinu těžké práce udělají za nás. Tato součást je důležitá z důvodu rozhnodutí, zda se jedná o pakety typu UDP nebo TCP, protože jiné pakety tato aplikace nepodporuje. Dále je tento zpracovaný paket použit k zjištění portů. Viz například [1].

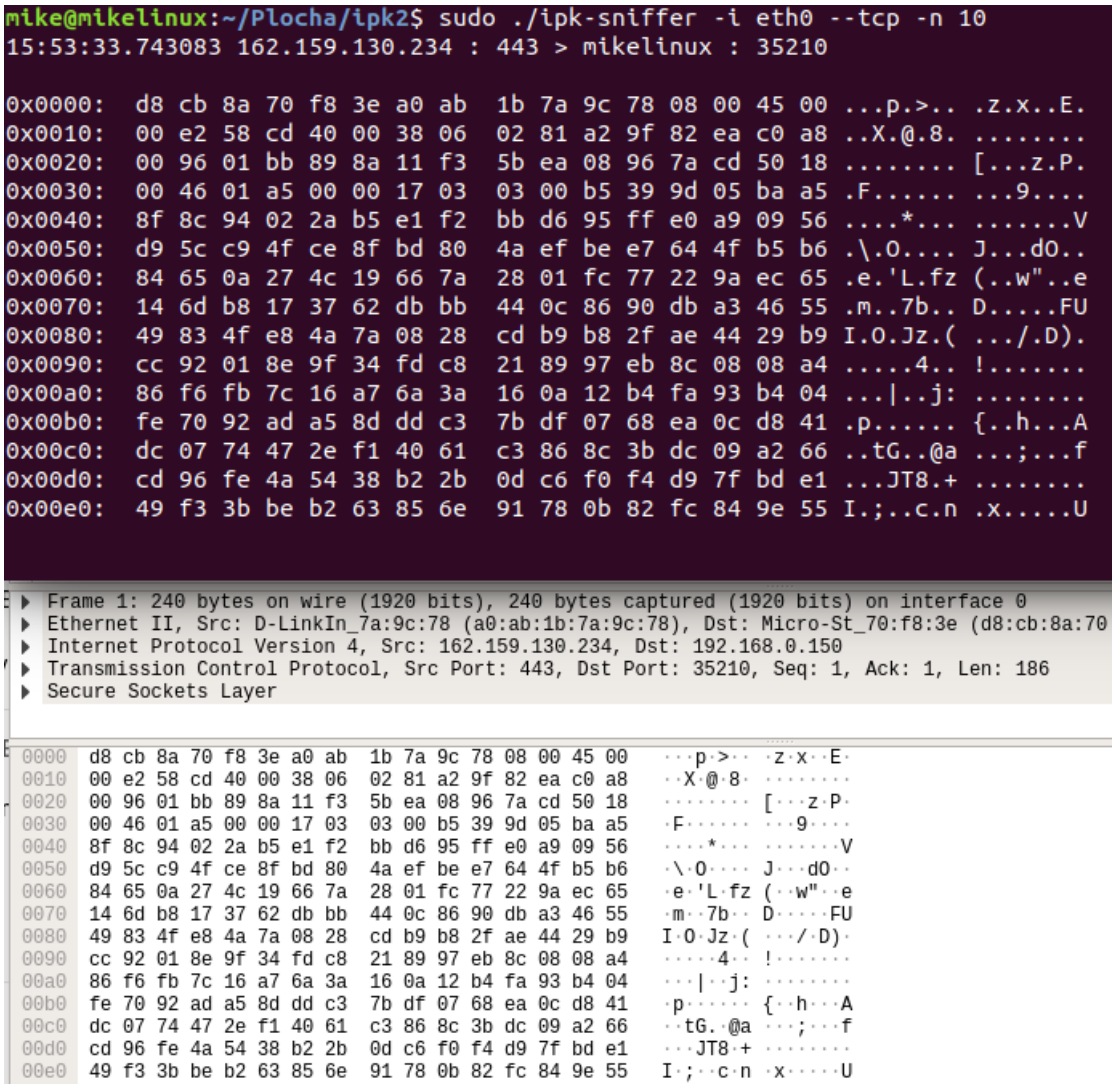
```
var tcp = packet_parse.Extract<PacketDotNet.TcpPacket>();
var udp = packet_parse.Extract<PacketDotNet.UdpPacket>();
/*ziskani ipv4 packetu */
var ipv4pak = packet_parse.Extract<PacketDotNet.IPv4Packet>();
```

Obrázek 3: Ukázka struktury použíté z knihovny PacketDotNet

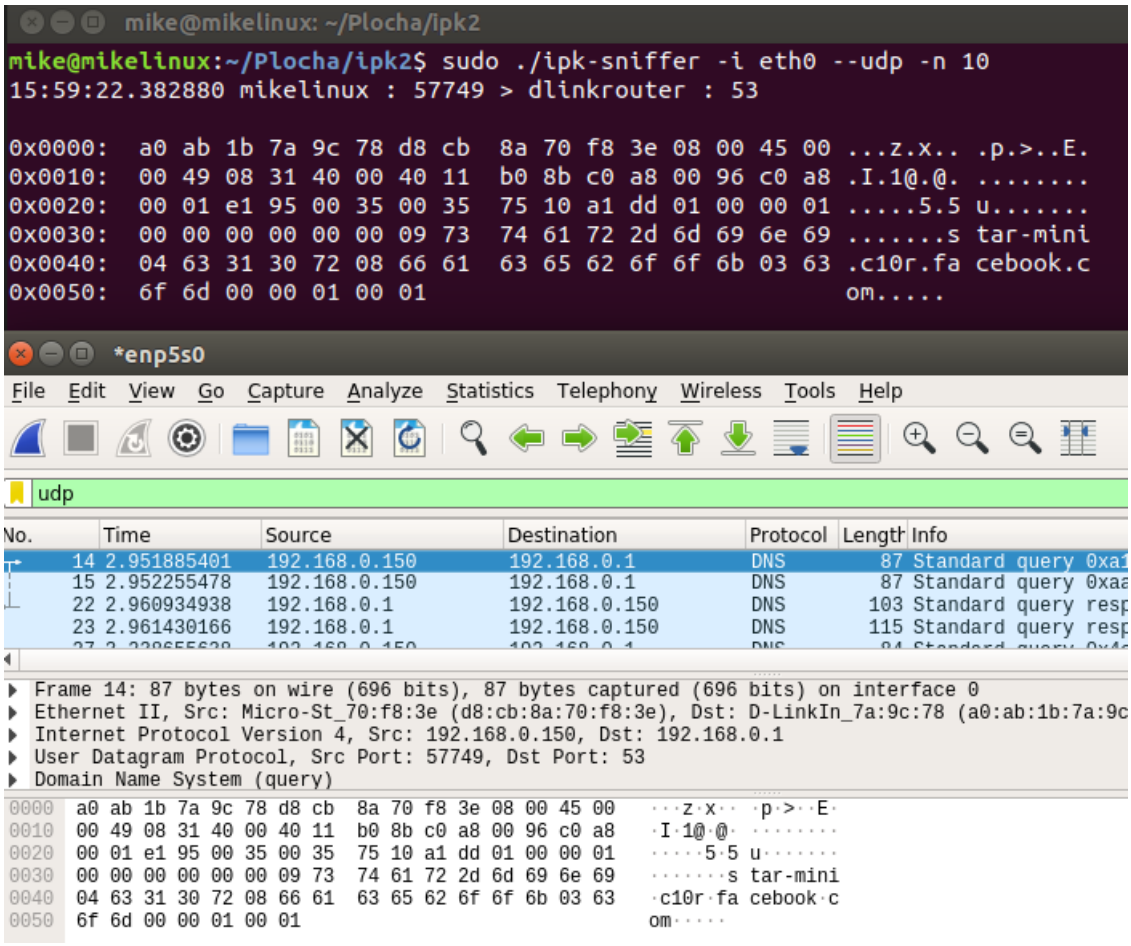
4 Testování

Testování aplikace probíhálo spuštěním mnou implementované aplikace současně s volně dostupným nástrojem (Wireshark)[6]. Nejprve jsem spustil filtrování paketů pomocí programu ipk-sniffer a hned na to filtrování v nástroji Wireshark. Poté jsem hledal první stejný paket pro obě aplikace. Když se mi podařilo takový najít, ručně jsem kontroloval zda jednotlivé výpisy sedí. Nalezení stejného paketu byl většinou oříšek, protože nelze oba programy spustit a vypnout v jeden okamžik. Příkládám snímky obrazovky z testování.

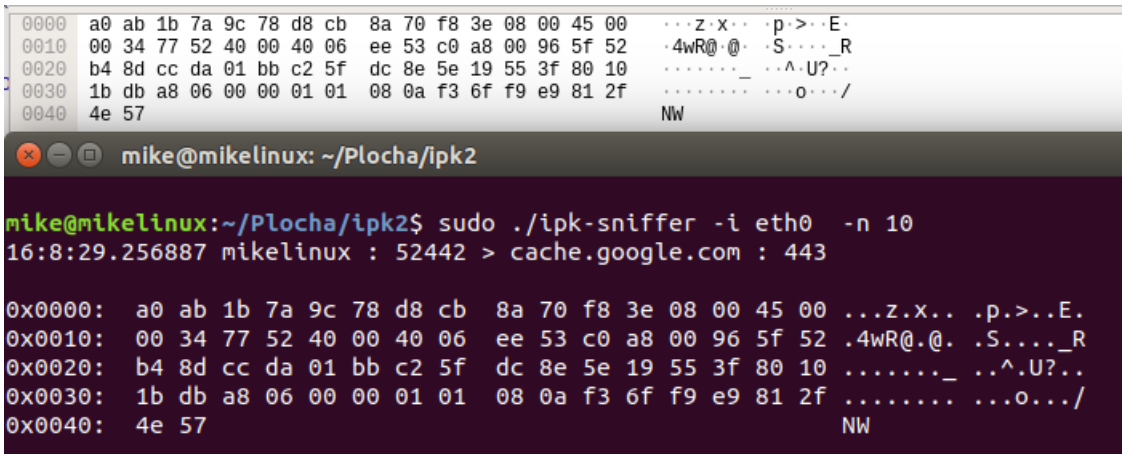
4.1 Snímky testování



Obrázek 4: Testování paketu TCP



Obrázek 5: Testování paketu UDP



Obrázek 6: Testování obou paketů

Reference

[1] chmorgan: packetnet. [online], rev. 1. duben 2020, [vid. 2020-04-28].
URL https://github.com/chmorgan/packetnet

[2] kayoub5: SharpPcap. [online], rev. 26. duben 2020, [vid. 2020-04-28].
URL https://github.com/chmorgan/sharppcap/

[3] kayoub5: SharpPcap. [online], rev. 29. prosinec 2019, [vid. 2020-04-28].
URL https://github.com/chmorgan/sharppcap/blob/master/Examples/Example4.BasicCapNoCallback/Example4.BasicCapNoCallback.cs

[4] Postel, J.: User Datagram Protocol. [online], rev. 28. srpen 1980, [vid. 2020-04-28].
URL https://tools.ietf.org/html/rfc768

[5] Wikipedia: Transmission Datagram Protocol. [online], rev. 20. duben 2020, [vid. 2020-04-28].
URL https://cs.wikipedia.org/wiki/Transmission_Control_Protocol

[6] Wireshark: Wireshark Documentation. [online], rev. 20. duben 2020, [vid. 2020-04-28].
URL https://www.wireshark.org/docs/