

# IPK 2. Projekt – Analyzátor síťových paketů

Tomáš Dvořáček (xdvora3d)

# Obsah

1 Úvod	2
2 Nastudované informace .....	2
2.1 protokol TCP .....	2
2.2 protokol UDP .....	2
3 Implementace .....	3
3.1 Zachytávání paketů .....	3
3.2 Překlad IP na FQDN .....	3
3.3 Zpracování zachycených paketů .....	3
4 Testování .....	4
5 Zdroje .....	5

# 1 Úvod

Úkolem tohoto projektu je zachytávat TCP | UDP pakety na zadaném rozhraní, a následně vypsat jejich obsah na standardní výstup v hexadecimálním a textovém formátu. Je též možné zvolit port na kterém se bude zachytávat a celkový počet paketů, které budou zachyceny.

## 2 Nastudované informace

### 2.1 TCP

Transmission control protocol je spolehlivý protokol transportní vrstvy. Jeho hlavička obsahuje kontrolní bity, které zajišťují spolehlivost. Důležitým parametrem, zejména pro tento projekt je délka TCP hlavičky, která se může pohybovat od minimálních 20 bytů do volitelných 60 bytů.

		TCP segment header																															
Offsets	Octet	0								1								2								3							
Octet	Bit	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0	7	6	5	4	3	2	1	0
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 0 0 0		N S	C W R	E C E	U R G	A C K	P S H	R S T	S Y N	F I N	Window Size																
16	128	Checksum																Urgent pointer (if URG set)															
20	160	Options (if <i>data offset</i> > 5. Padded at the end with "0" bytes if necessary.)																															
...	...	...																															

TCP hlavička, Zdroj: [1]

### 2.2 UDP

User datagram protocol je na rozdíl od TCP nespolehlivý, ale jeho výhodou je menší režie při přenosu dat. Jeho délka je 8 bytů.

		UDP datagram header																															
Offsets	Octet	0								1								2								3							
	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Length																Checksum															

UDP hlavička, Zdroj: [2]

## 3 Implementace

Program je členěn do 5 funkcí, funkce *main* nejprve zpracuje argumenty, ze kterých se sestaví filtr pro *p\_cap* což je funkce, jež zachytává samotné pakety. Tato funkce si zároveň po každém zachyceném paketu spustí obslužnou funkci *packet\_handler* v níž se roztrídí pakety podle protokolů transportní vrstvy. Poslední dvě funkce *print\_packet* a *print\_data* už slouží jen ke správnému formátování a výpisu na standardní výstup.

### 3.1 Zachytávání paketů

Před zachytáváním paketů je nejprve potřeba nastavit příslušné filtry. O toto se starají funkce *pcap\_compile* (kompiluje textový řetězec na filtrovací program, viz. [3]) a funkce *pcap\_setfilter* (nastaví filtrovací program, viz. [4]). Následně je spuštěna funkce *pcap\_loop*, která zajišťuje samotné zachytávání paketů. U této funkce je možné zvolit kolik paketů se má zachytit. Při každém zachycení se spustí obslužná funkce *packet\_handler*.

### 3.2 Překlad IP na FQDN

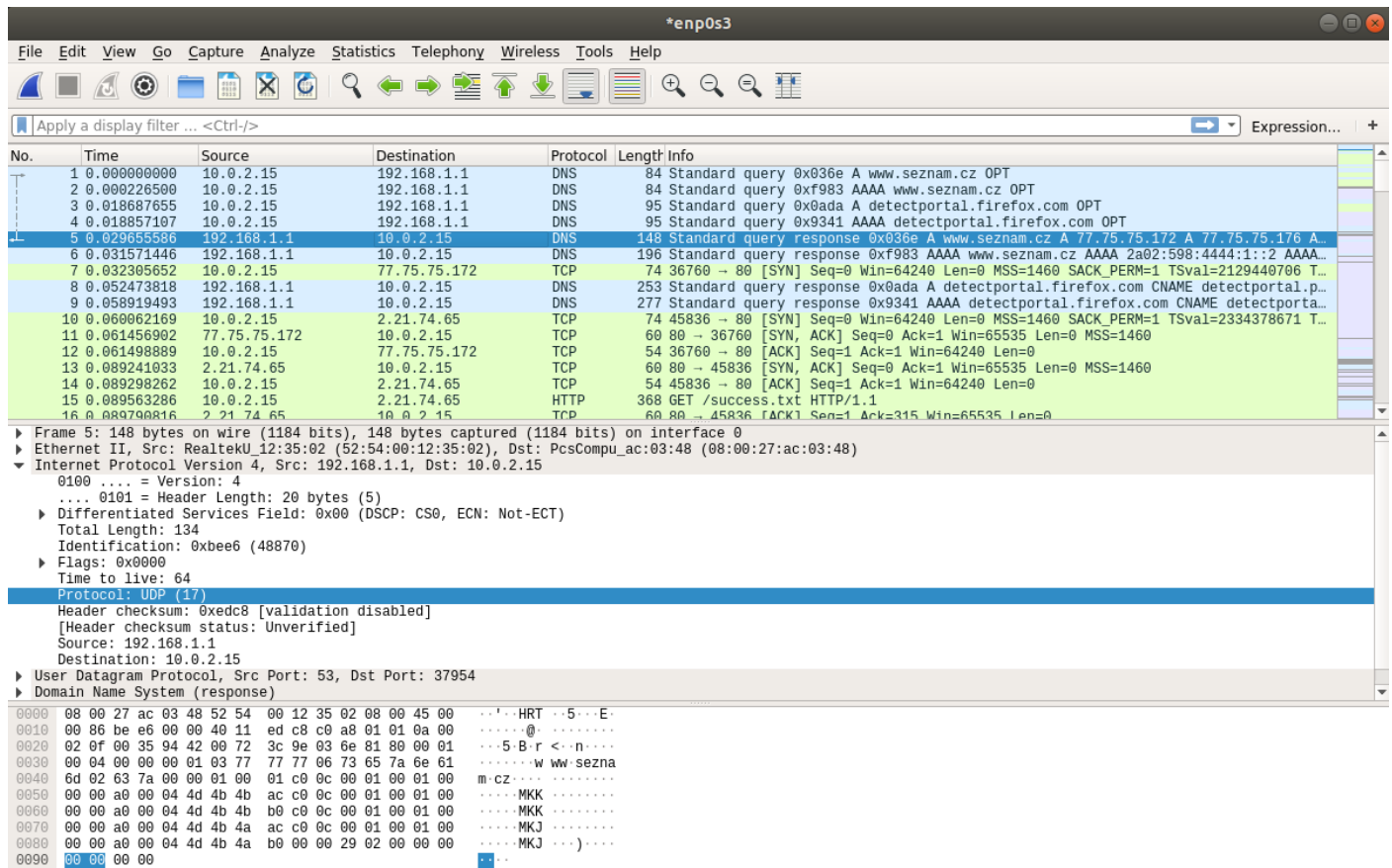
Pro převod ip na hostname u zdrojových a cílových ip adres je použita funkce *getnameinfo*. Zde je nutno poznamenat, že při převodu na doménové jméno může docházet ke generování dalších paketů, což může vést k zacyklení, proto jsem přidal další parametr **-h** pro vypnutí překladu ip na FQDN.

### 3.2 Zpracování zachycených paketů

Po zachycení paketu se paket namapuje na strukturu ip (s příslušným posunem odpovídajícím 14 bytů v případě ethernetové hlavičky nebo 16 bytů v případě linux cooked capture). Z této struktury se pak zjistí délka ip hlavičky a číslo použitého protokolu, přičemž číslo 6 odpovídá protokolu TCP a číslo 17 protokolu UDP. Následně se podle protokolu a verze ip vypočítá celková délka paketu a paket se vypíše.

## 4 Testování

Testování probíhalo prostým ručním porovnáváním standardního výstupu s výstupem programu wireshark. Pro otestování IPv6 jsem použil program curl, pro otestování IPv4 jsem monitoroval prostý síťový provoz.



The screenshot displays the Wireshark interface with a network traffic capture on interface \*enp0s3. The packet list shows several DNS queries and responses, followed by an HTTP GET request. The selected packet (No. 5) is a DNS Standard query response from 192.168.1.1 to 10.0.2.15. The packet details pane shows the following information:

- Frame 5: 148 bytes on wire (1184 bits), 148 bytes captured (1184 bits) on interface 0
- Ethernet II, Src: RealtekU\_12:35:02 (52:54:00:12:35:02), Dst: PcsCompu\_ac:03:48 (08:00:27:ac:03:48)
- Internet Protocol Version 4, Src: 192.168.1.1, Dst: 10.0.2.15
- 0100 .... = Version: 4
- .... 0101 = Header Length: 20 bytes (5)
- Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
- Total Length: 134
- Identification: 0xb2e6 (48870)
- Flags: 0x0000
- Time to live: 64
- Protocol: UDP (17)
- Header checksum: 0xedc8 [validation disabled]
- [Header checksum status: Unverified]
- Source: 192.168.1.1
- Destination: 10.0.2.15
- User Datagram Protocol, Src Port: 53, Dst Port: 37954
- Domain Name System (response)

The packet bytes pane shows the raw data in hexadecimal and ASCII format, including the DNS response structure and the domain name 'www.seznam.cz'.

Program wireshark

## 5 Zdroje

- [1] [https://en.wikipedia.org/wiki/Transmission\\_Control\\_Protocol](https://en.wikipedia.org/wiki/Transmission_Control_Protocol)
- [2] [https://en.wikipedia.org/wiki/User\\_Datagram\\_Protocol](https://en.wikipedia.org/wiki/User_Datagram_Protocol)
- [3] [https://www.tcpdump.org/manpages/pcap\\_compile.3pcap.html](https://www.tcpdump.org/manpages/pcap_compile.3pcap.html)
- [4] [https://www.tcpdump.org/manpages/pcap\\_setfilter.3pcap.html](https://www.tcpdump.org/manpages/pcap_setfilter.3pcap.html)