FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

2. projekt do IPK

Varianta ZETA: Sniffer paketů

25. dubna 2021 Filip Jahn (xjahnf00)

Obsah

1 Úvod

Cílem projektu bylo naprogramování snifferu paketů na zadaném rozhraní. Analyzátor paketů je program, který umožňuje ochytávat a analyzovat komunikaci v počítačové síti. Tento síťový analyzátor dokáže obě části zároveň. Využití lze nacházet např. při detekci zneužívání sítě, nebo při analýze problémů v síti.

Spouštění programu umožňuje zadání přepínačů pro specifikaci, které pakety se mají odchytávat, jejich množství a na jakém rozhranní se budou operace vykonávat. Vypsání dat z paketu probíhá ve dvou podobách – v hexa a ascii reprezentaci.

V první kapitole si popíšeme, jaké knihovny byly využity pro realizaci projektu. Uvedeme si i seznam globálních proměnných a jejich obhajobu, proč byly použity a za jakým účelem. V následující kapitole se podíváme, jak probíhá inicializace pcap rozhranní. Zde si také uvedeme, jakým způsobem je kupř. plněn řetězec filterStr a k čemu slouží. Dále nás čeká rozebrání paketu. Z čeho se skládá a jak z něj získáme veškeré potřebné informace. A tak se samovolně dostaneme do poslední fáze dokumentace, kde nás čeká popis procesu vypsání paketu.

2 Knihovní funkce

Knihovna	Využití
`arpa`	Převody adres na řetězce.
`netinet`	Přetypování paketu, díky tomu získání dat.
`pcap`	Aplikační rozhranní pro odchyt síťové komunikace.

Tabulka 1: Přehledová tabulka použitých knihoven

2.1 Definovaná makra a globální proměnné

Makra

Název	Délka	Popis
`ETH_HDR`	14	Ethernetová hlavička má vždy 14 bajtů. O tolik se posouváme, chceme-li číst další hlavičku.
`IPV6_HDR`	40	Hlavička protokolu IPV6 má konstantní délku hlavičky a to 40 bajtů.

Tabulka 2: Makra a jejich využití

Proměnné Argumenty reprezentují globální proměnné, abychom k nim umožnili přístup na více místech v programu. K jejich naplnění hodnotami využíváme funkci parse_arguments a ve funkci fill_filter_str na jejich základě vytvoříme filtr (podrobněji pak v sekci Filtrovací pravidla 3.2.1). Absencí předávání argumentů jako parametr napříč funkcemi se tak řešení stává čitelnějším.

- **boolean:**
  - tcp, udp, arp, icmp – V sekci zpracování argumentů se těmto proměnným přiřadí `true`, pakliže tak uživatel zadá při spuštění. Pokud je proměnná nastavena na `true`, budou se odchytávat právě takové pakety. Pokud uživatel nezadá ani jeden protokol, budou se uvažovat všechny.
- **char*:**
  - device – Rozhranní, na kterém budeme odchytávat.
  - port – Slouží pro filtr paketů. Budeme tak uvažovat pouze pakety se zadaným portem.
- **int:**
  - numOfPackets – Počet paketů, kolik se má vypsat. Výchozí hodnota je nastavena na jedna.

3 Hlavní funkce `main`

3.1 Analýza argumentů

Po spuštění programu se vrháme na syntaktickou analýzu zadaných argumentů při spuštění programu. Celá kontrola je založena na cyklu, který prochází zadané argumenty za pomoci knihovní funkce getopt_long [10]. Využitím této funkce si značně ulehčíme práci, když může být argument zadán jak krátkým zápisem, tak dlouhým. Zároveň zde probíhá sémantická kontrola. Jestliže přijdeme na to, že uživatel zadal nesmyslné požadavky, vypíšeme error na `stderr` a ukončujeme aplikaci s návratovým kódem 3.

Při správně zadaných argumentech přiřazujeme hodnoty náležícím proměnným a pokračujeme inicializací `pcap` rozhraní.

3.2 Inicializace `pcap`

Proniknutí do úvodní problematiky „jak začít“ značně usnadňuje blog, který uvádím v citaci [4]. Inicializace je zde krásně vysvětlena a ukázána na praktických příkladech. Funkce main využívá velkou většinu popsaných funkcí.

3.2.1 Filtrovací pravidla

Pro otevření zařízení na sledování paketů využíváme knihovní funkci `pcap_open_live` [11] z knihovny `pcap`[1]. Pokud otevření proběhne bez problémů, bude dalším úkolem kompilace řetězce filterStr.

Řetězec `filterStr` udržuje informace o tom, které hlavičky paketů se mají odchytávat, příp. na jakých portech sledovat síťový provoz. Abychom mohli filtr aplikovat v naší problematice, musíme nejprve převést daný řetězec na filtr programu, kterému pcap rozumí. Kompilaci provádíme za použití knihovní funkce `pcap_compile` [7] ze stejné knihovny jako v předchozím případě.

V neposlední řadě nám zbývá filtr vložit do pcap rozhraní. K tomuto účelu je využita třetí funkce, a to `pcap_setfilter` [9]. Pakliže se do tohoto bodu nevyskytly žádné problémy, jsme plně připraveni. Pokud by se však nějaké objevily, bude vyhlášena chybová hláška na `stderr` a program bude ukončen s návratovou hodnotou 2.

3.2.2 Čtení jednotlivých paketů

Nyní přichází čas na odchytávání paketů, jenž je vykonáváno ve funkci `pcap_loop`[8]. Tato funkce většinu času pouze vyčkává. S příchodem každého paketu se pak volá funkce, jejíž název je součástí argumentů pcap_loop.

Po vypsání určitého počtu paketů[2] dochází k uzavření sledování a ukončení aplikace s návratovým kódem 0.

[1] Viz: Tabulka 2

[2] Počet paketů je odvozen od toho, kolik si uživatel zadal při spuštění argumentem -n počet.

4 Funkce `process_packet`

Po přijmutí paketu si ukládáme čas jeho přijetí. Jak čas získáváme a jak ho převádíme na formát rfc3339 je popsáno v sekci 5.2. Nyní musíme rozhodnout, o který internetový protokol se jedná.

Tato informace je nesena v první hlavičce. Jedná se o `ethernetovou hlavičku`. Tato hlavička je vždy 14 bajtů dlouhá[3]. Vytvoříme si proto strukturu: `struct ether_header *ipvNum`, do které uložíme paket přetypovaný na danou strukturu. Tím získáme možnost se odkázat na parametr ve struktuře, kde je k nalezení typ následující hlavičky. Na výběr máme z těchto tří hlaviček: IPv4, IPv6 nebo ARP. Veškeré internetové protokoly jsou popsány v následujících sekcích.

4.1 IPv4

Prvním protokolem, na který se podíváme blíže, bude `IPv4`. Využíváme zde strukturu `struct iphdr *ipHeader`, která je blíže popsaná zde v dokumentaci [14]. Abychom ale mohli přijatý paket přetypovat na tuto strukturu, musíme se prvně posunout o ethernetovou hlavičku. Tak učiníme bitovým posuvem o 14 bajtů. Tím se dostaneme na `IP hlavičku`, která poskytuje parametr `protocol`, díky němuž můžeme určit jaký protokol transportní vrstvy je využit. Potřebná data ukládáme do proměnné `ipHeader`. Přepínač tak rozhoduje na základě ipHeader - > protocol. Zde máme opět výběr ze tří možností: ICMP, TCP nebo UDP.

ICMP

Pokud se jedná o ICMP, přesuneme se do funkce `icmp_v4`. Před tisknutím každého paketu[4] však ještě vytiskneme název protokolu. Hlavní důvod byl kvůli testování, ale tento doplněk mi přišel užitečný a tak byl zachován. V ICMP nás zajímají pouze IP adresy, které jsou k nalezení ve struktuře `struct ip* iphdrVar` [15]. Abychom však mohli tyto adresy vypsat, musíme je překonvertovat z IPv4 notace do binární podoby. S tím nám pomáhá funkce inet_ntoa [5]. Ze snímků obrazovky[5] můžeme vidět, že výpis probíhá standardně jako u ostatních protokolů, chybí zde pouze porty.

TCP a UDP

Pokud však přepínač rozhodl, že se jedná o TCP nebo UDP protokol, musíme si vypočítat proměnnou délku hlavičky v obou případech. To je lehká nevýhoda oproti IPv6 protokolu. Výpočet probíhá následujícím způsobem: vezmeme si již zmíněnou proměnnou ipHeader, která skýtá IP hlavičku. Můžeme zde nalézt parametr s názvem `ihl`, který nese informaci o délce hlavičky jako počet 32bitových slov. Pro získání délky v bajtech je potřeba vynásobit tuto délku čtyřmi [20]. Hodnotu pak ukládáme do proměnné `ipLen`.

Nyní se posuneme do funkce `tcp_v4` nebo `udp_v4`, ve kterých získáme IP adresy stejně, jako je tomu v ICMP. Novinkou je zde získání portů. Na ty se musíme z IP hlavičky dále posunout o hodnotu z proměnné ipLen. Tady se nám situace rozděluje.

- TPC – Hlavičku si ukládáme do proměnné `tcphdrVar`, jenž je strukturou struct tcphdr* [17]. K portům pak přistupujeme tcphdr4->th_sport nebo tcphdr4->th_dport.
- UDP – Hlavičku musíme naopak přetypovat na strukturu struct udphdr *`udphdrVar` [18]. K portům se pak dostaneme udphdrVar->uh_sport a udphdrVar->uh_dport.

V obou případech (jak TCP tak UDP) musíme překonvertovat porty, jež jsou v síti posílány po bajtech, na celá čísla. K tomu nám dopomáhá funkce `ntohs` [6].

[3] Připomenutí: Tabulka 2

[4] Před každým paketem tiskneme prvně název protokolu ohraničený závorkami. Až na další řádek veškerá získaná data z paketu.

[5] Veškeré snímky jsou k nalezení v sekci 6.

4.2 IPv6

Druhým internetovým protokolem, který si popíšeme, je protokol IPv6. Podobně jako u předchozího protokolu, i zde máme IP hlavičku, kterou si udržujeme v proměnné s názvem `ip6Header`. Tato proměnná je ale struktury `struct ip6_hdr*` [16]. Na základě přepínání `ip6Header->ip6_nxt` máme na výběr z následujících možností: ICMPv6, TCP a UDP.

ICMPv6

Přesouváme se do funkce `icmp_v6`. Pro získání IP adres z paketu používáme knihovní funkci `inet_ntop`[13], která převádí adresu z binární podoby do textové. Do funkce vkládáme argument přetypované ip6 hlavičky a ukazatele na IP adresu. Výpis pak provádíme stejným způsobem jako u ICMP protokolu IPv4.

TCP

Jak již bylo řečeno dříve, u IPv6 máme statickou délku hlavičky a to 40 bajtů. Ve funkci `tcp_v6` si zjistíme IP adresy stejně jako je tomu u ICMPv6. Pro vytáhnutí čísla protokolu z paketu se musíme posunout na hlavičku TCP. Posun je tak nejen o ethernetovou hlavičku, ale i o 40 bajtů[6]. Přetypováním na `struct tcphdr*` získáme možnost přistoupit k parametrům `th_sport` a `th_dport`, kde náleží hledaná data. Pro konvertování na řetězec využíváme knihovní funkci ntohs [6].

UDP

Transportní protokol UDP se liší od TCP pouze v získávání portů. Ve funkci `udp_v6` musíme přetypovat hlavičku na UDP. Ukázka struktury: `struct udphdr *udphdrVar`. Přistupování k parametrům udržující číslo portu je pak `udphdrVar->uh_sport` a `udphdrVar->uh_dport`.

4.3 ARP

Nejjednodušším protokolem je ARP. U toho zjišťujeme nejen IP, ale i MAC adresy. Ty poté vypisujeme v závorkách za příslušnou IP adresou. Navíc nás zajímá, jestli se jedná o požadavek nebo odpověď. Tuto informaci vypisujeme v závorkách před sdělením o délce paketu.

Pro získání IP adresy používáme strukturu `struct ether_arp*` [19], v níž se odkazujeme na `arp_spa` a `arp_tpa` parametry. Pro získání MAC adresy však musíme využít knihovní funkci `ether_ntoa` [12].

5 Ostatní funkce

5.1 Funkce `print_interface`

Při zadání samotného parametru -i nebo --interface je proveden výpis rozhranní, na kterých je možnost odchytávání paketů. Tak se tomu stane i v případě, kdy je program spuštěn bez argumentů. Funkci jsem našel na internetu a lehce ji upravil pro své řešení[1].

5.2 Funkce `time_rfc3339`

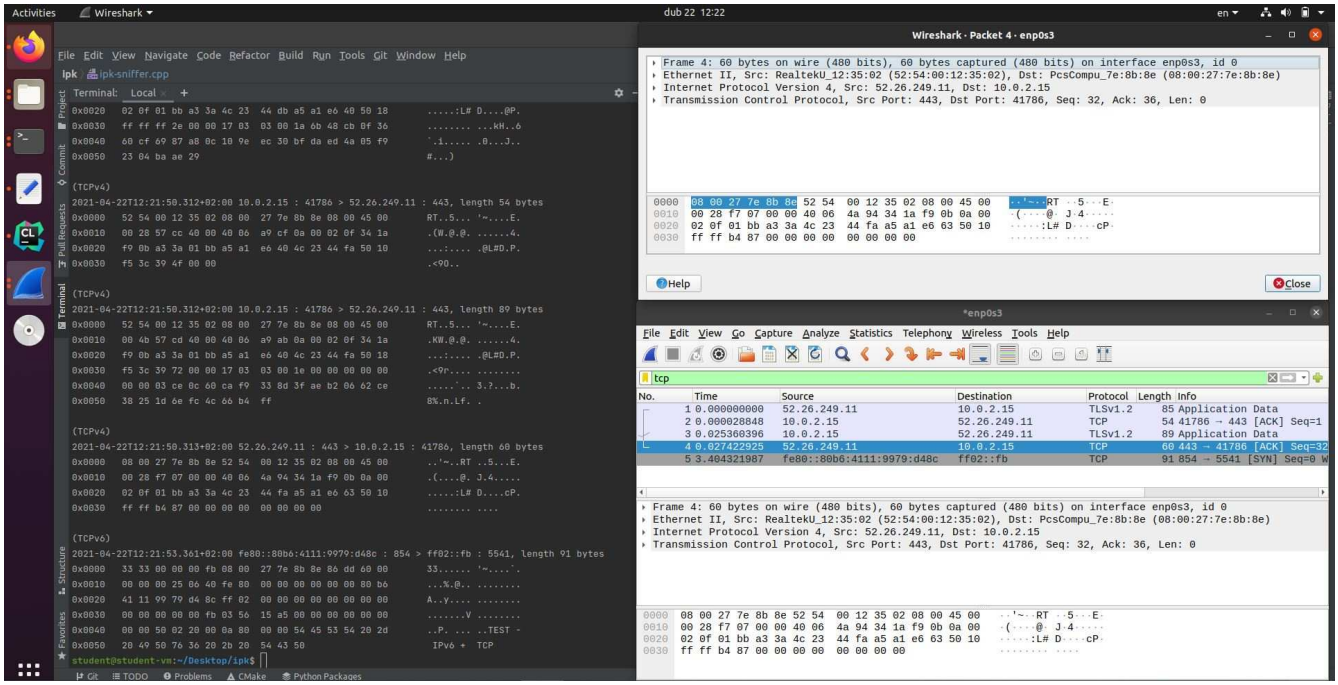
Čas získáváme užitím funkce strftime [3]. Tato funkce nám ale nedokáže poskytnout čas v milisekundách na tři desetiny, jak je požadováno v zadání. A tak milisekundy „přilepuji“ funkcí sprintf ručně [2].

[6] Pro připomenutí viz sekce 2.1

6 Testování

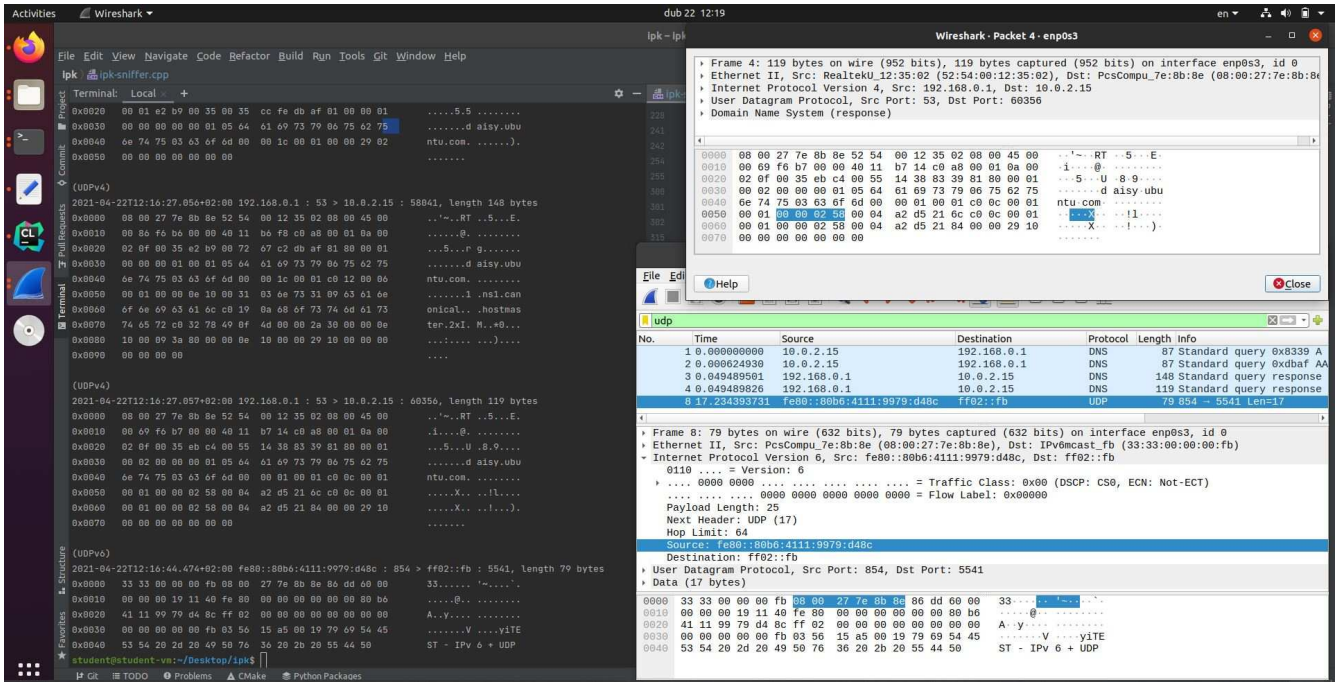
Testování proběhlo porovnáním výstupu programu s referenčním programem Wireshark. Veškerá funkcionalita může být viděna na následujících snímcích obrazovky, kde na levé straně je můj program a na pravé je možnost porovnání minimálně dvou paketů.

- TCP protokolu IPv4 + IPv6



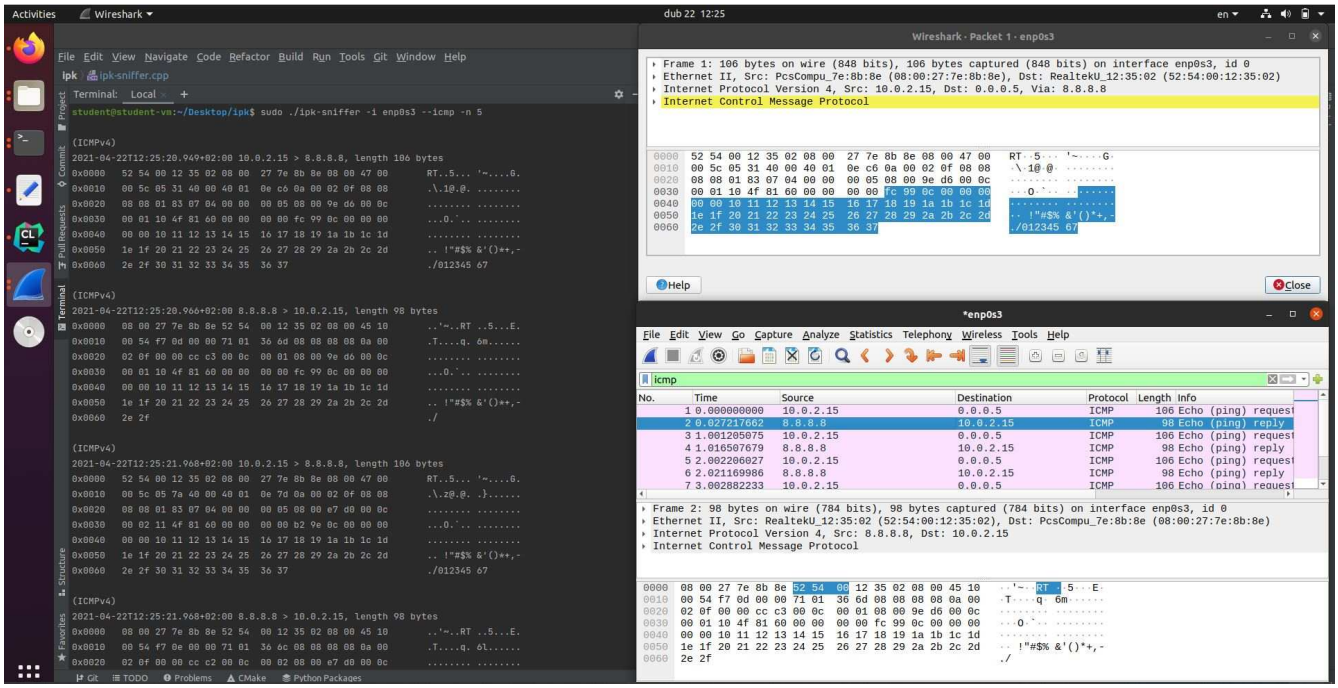
Obrázek 1: Protokoly TCPv4 spoleřčně s TCPv6

- UDP protokolu IPv4 + IPv6



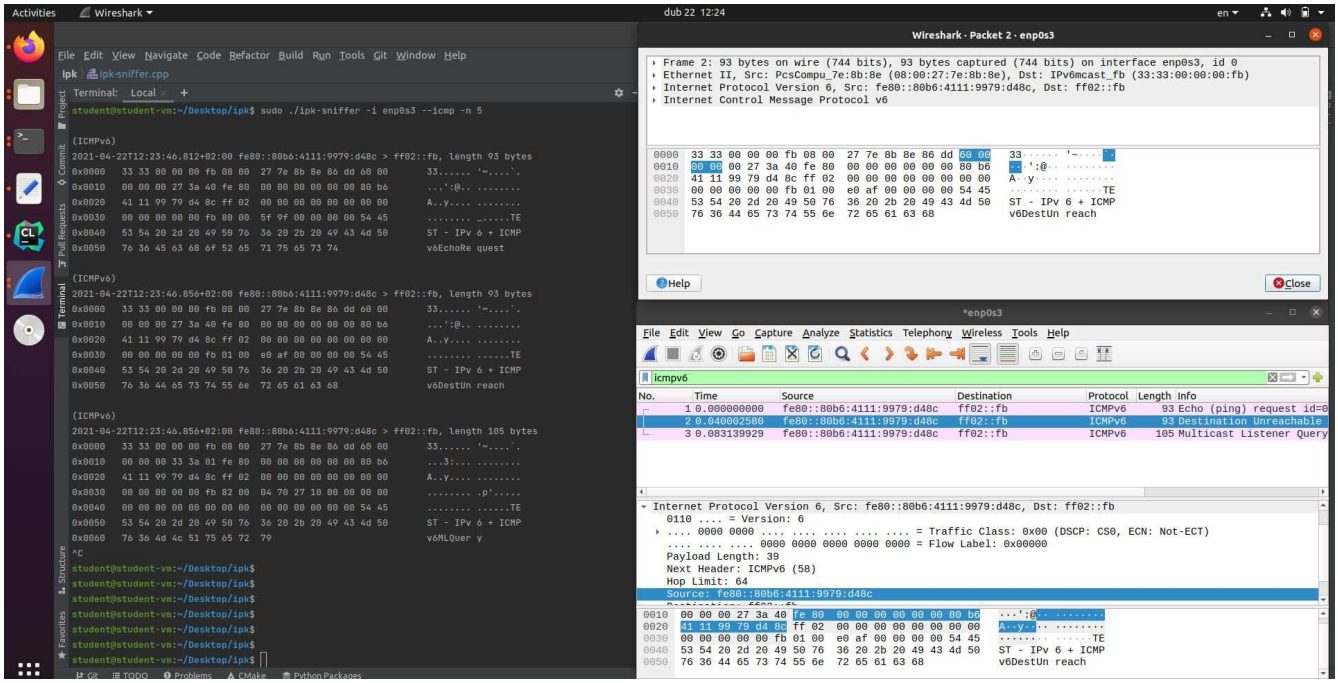
Obrázek 2: Protokoly UDPv4 spoleřčně s UDPv6

- ICMP protokolu IPv4



Obrázek 3: Protokoly ICMP

- ICMP protokolu IPv6



Obrázek 4: Protokoly ICMPv6

Reference

[1] BLOGSPOT.COM. *Zdroj pro výpis rozhranní* [online]. Poslední změna 1. listopadu 2020 [cit. 25. dubna 2021]. Dostupné na: <http://embeddedguruji.blogspot.com/2014/01/pcapfindalldevs-example.html>.

[2] CPLUSPLUS.COM. *Manuál k používání funkce sprintf* [online]. Poslední změna 1. listopadu 2020 [cit. 25. dubna 2021]. Dostupné na: <https://www.cplusplus.com/reference/cstdio/sprintf/>.

[3] CPLUSPLUS.COM. *Manuál k používání funkce strftime* [online]. Poslední změna 1. listopadu 2020 [cit. 25. dubna 2021]. Dostupné na: <http://www.cplusplus.com/reference/ctime/strftime/>.

[4] LINUX.DIE.NET. *Blog o úvodní inicializaci* [online]. Poslední změna 1. listopadu 2020 [cit. 25. dubna 2021]. Dostupné na: <https://www.tcpdump.org/pcap.html>.

[5] LINUX.DIE.NET. *Dokumentace inet_ntoa* [online]. Poslední změna 1. listopadu 2020 [cit. 25. dubna 2021]. Dostupné na: <https://linux.die.net/man/3/inet$_n$*toa*>.

[6] LINUX.DIE.NET. *Dokumentace ntohs* [online]. Poslední změna 1. listopadu 2020 [cit. 25. dubna 2021]. Dostupné na: <https://linux.die.net/man/3/ntohs>.

[7] LINUX.DIE.NET. *Dokumentace pcap_compile* [online]. Poslední změna 1. listopadu 2020 [cit. 25. dubna 2021]. Dostupné na: <https://linux.die.net/man/3/pcap$_c$*ompile*>.

[8] LINUX.DIE.NET. *Dokumentace pcap_loop* [online]. Poslední změna 1. listopadu 2020 [cit. 25. dubna 2021]. Dostupné na: <https://linux.die.net/man/3/pcap$_l$*oop*>.

[9] LINUX.DIE.NET. *Dokumentace pcap_setfilter* [online]. Poslední změna 1. listopadu 2020 [cit. 25. dubna 2021]. Dostupné na: <https://linux.die.net/man/3/pcap$_s$*etfilter*>.

[10] LINUX.DIE.NET. *Getopt_long* [online]. Poslední změna 1. listopadu 2020 [cit. 25. dubna 2021]. Dostupné na: <https://linux.die.net/man/3/getopt$_l$*ong*>.

[11] LINUX.DIE.NET. *Pcap_open_live* [online]. Poslední změna 1. listopadu 2020 [cit. 25. dubna 2021]. Dostupné na: <https://linux.die.net/man/3/pcap$_o$*pen$_l$ive*>.

[12] MAN7.ORG. *Dokumentace ether_ntoa* [online]. Poslední změna 1. listopadu 2020 [cit. 25. dubna 2021]. Dostupné na: <https://man7.org/linux/man-pages/man3/ether$_a$*ton*.3.*html*>.

[13] MAN7.ORG. *Dokumentace inet_ntop* [online]. Poslední změna 1. listopadu 2020 [cit. 25. dubna 2021]. Dostupné na: <https://man7.org/linux/man-pages/man3/inet$_n$*top*.3.*html*>.

[14] SITES.UCLOUVAIN.BE. *Dokumentace ip hlavičky* [online]. Poslední změna 1. listopadu 2020 [cit. 25. dubna 2021]. Dostupné na: <https://sites.uclouvain.be/SystInfo/usr/include/netinet/ip.h.html>.

[15] SITES.UCLOUVAIN.BE. *Dokumentace ip hlavičky* [online]. Poslední změna 1. listopadu 2020 [cit. 25. dubna 2021]. Dostupné na: <https://sites.uclouvain.be/SystInfo/usr/include/netinet/ip.h.html>.

[16] SITES.UCLOUVAIN.BE. *Dokumentace ip6 hlavičky* [online]. Poslední změna 1. listopadu 2020 [cit. 25. dubna 2021]. Dostupné na: <https://sites.uclouvain.be/SystInfo/usr/include/netinet/ip6.h.html>.

[17] SITES.UCLOUVAIN.BE. *Dokumentace tcpv4 hlavičky* [online]. Poslední změna 1. listopadu 2020 [cit. 25. dubna 2021]. Dostupné na: <https://sites.uclouvain.be/SystInfo/usr/include/netinet/ip.h.html>.

[18] SITES.UCLOUVAIN.BE. *Dokumentace udpv4 hlavičky* [online]. Poslední změna 1. listopadu 2020 [cit. 25. dubna 2021]. Dostupné na: <https://unix.superglobalmegacorp.com/Net2/newsrc/netinet/udp.h.html>.

[19] SUPERGLOBALMEGACORP.COM. *Dokumentace arp hlavičky* [online]. Poslední změna 1. listopadu 2020 [cit. 25. dubna 2021]. Dostupné na: <https://unix.superglobalmegacorp.com/Net2/newsrc/netinet/if$_e$$ther.h.html$>.

[20] WIKIPEDIA.ORG. *Výpočet délky hlavičky* [online]. Poslední změna 1. listopadu 2020 [cit. 25. dubna 2021]. Dostupné na: <https://cs.wikipedia.org/wiki/IPv4>.