VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

EXTRAKCE TUNELOVANÝCH DAT DO SAMOSTATNÝCH TOKŮ

TUNNELED DATA EXTRACTION INTO SEPARATE FLOWS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **ROMAN NAHÁLKA**
AUTHOR

VEDOUCÍ PRÁCE **Ing. MARTIN HOLKOVIČ**
SUPERVISOR

BRNO 2018

Zadání bakalářské práce/20798/2017/xnahal01

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2017/2018

Zadání bakalářské práce

Řešitel: **Nahálka Roman**
Obor: Informační technologie
Téma: **Extrakce tunelovaných dat do samostatných toků**
Tunneled Data Extraction into Separate Flows
Kategorie: Počítačové sítě

Pokyny:

1. Nastudujte protokoly používané v rámci síťové architektury TCP/IP a možnosti spracovávaní síťových toků.
2. Po konzultaci s vedoucím práce se seznamte s vybranými tunelovacími protokoly a zaměřte se na možnost extrakci zapouzdřených dat. Vytvořte testovací dataset.
3. Navrhněte extraktor tunelovaných dát.
4. Návrh implementujte a otestujte na vytvořených testovacích datech.
5. Proveďte zhodnocení výsledků.

Literatura:

- Hanks, S., Li, T., Farinacci, D., and P. Traina, "Generic Routing Encapsulation over IPv4 networks", RFC 1702, DOI 10.17487/RFC1702, October 1994.
- Townsley, W., Valencia, A., Rubens, A., Pall, G., Zorn, G., and B. Palter, "Layer Two Tunneling Protocol "L2TP"", RFC 2661, DOI 10.17487/RFC2661, August 1999.
- Sanders, Chris. Practical packet analysis: Using Wireshark to solve real-world network problems. No Starch Press, 2017.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 až 3 ze zadaní.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Holkovič Martin, Ing.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 16. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Cílem této práce je navrhnout a implementovat aplikaci pro extrakci tunelovaných dat do samostatných toků. Aplikace bude odstraňovat všechny vrstvy zapouzdření, které se ve vstupním souboru nachází. Využití nástroje spočívá v lepší analýze a diagnostice síťové komunikace. Díky odstranění tunelů ze síťového toku už bude tok obsahovat pouze data, se kterými se má dále pracovat. Práce se v teoretické části také zabývá síťovou architekturou TCP/IP, tunelovacími protokoly a způsoby, jakými lze na síti zachytávat komunikace. V praktické části je popsán způsob, jakým byly získávány testovací data, návrh cílové aplikace, implementace tohoto návrhu a testování výsledné aplikace.

Abstract

The goal of this work is to design and implement an application for extraction of tunneled data into seperate flows. The app will be removing all layers of encapsulation, that the file contains.The use of the app lays in better analysis and diagnostics of network communication. Thanks to removing the tunnels from the network flow, it will only contain data we can use. In the theoretical part, the work focuses on network architecture TCP/IP, the tunneling protocols and ways of capturing communication on the network. The practical part describes the way of retrieving test data, it also contains a design of the target application, as well as implementation of this design and testing of the final application.

Klíčová slova

TCP/IP, Wireshark, tunelování, extrakce, PCAP, IPIP, GRE, L2TP, PPTP

Keywords

TCP/IP, Wireshark, tunneling, extraction, PCAP, IPIP, GRE, L2TP, PPTP

Citace

NAHÁLKA, Roman. *Extrakce tunelovaných dat do samostatných toků*. Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Martin Holkovič

Extrakce tunelovaných dat do samostatných toků

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Martina Holkoviče. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Roman Nahálka
15. května 2018

Poděkování

Rád bych poděkoval vedoucímu mé práce, Ing. Holkovičovi, za odborné vedení bakalářské práce a cenné rady při vypracování práce. Také bych rád poděkoval rodině a přítelkyni za poskytnutou podporu při psaní práce.

Obsah

Kapitola 1

Úvod

V dnešní době je tunelování síťového provozu čím dál tím častější záležitost. Tunelování se totiž využívá u virtuálních privátních sítí (VPN). Používání VPN umožňuje poskytovat na internetu soukromí, které je v poslední době častěji vyžadováno. Při tunelování dochází k zapouzdření dat od některé vrstvy do jiného protokolu. Toto zapouzdření může překonávat některé omezení sítě a poskytovat soukromí.

Cílem této práce je navrhnout a implementovat aplikaci pro extrakci tunelovaných dat do samostatných toků. Bylo také nutné vytvořit testovací data pro testování aplikace. K tomu bylo zapotřebí vytvořit testovací síť pro vytváření tunelů mezi dvěma stanicemi v sítí a tento síťový provoz zachytávat. Aplikace by měla sloužit k lepší analýze a diagnostice zachycené komunikace na síti. Spousta nástrojů, které jsou určeny k analýze a diagnostice, totiž nepracuje s tunelovaným provozem dobře.

V práci je nejprve popsána síťová architektura TCP/IP. Tomuto problému se věnuje kapitola 2. V této kapitole jsou popsány jednotlivé vrstvy této síťové architektury a základní protokoly. Nakonec je v kapitole popsána možnost přidávání a odebírání vrstev. V kapitole 3 jsou popsány možnosti zachytávání a zpracování dat na síti. Kapitola se věnuje základním technikám, kterými lze zachytávat data na síti. Dále je zde popsán formát souboru, ve kterém jsou soubory se zachycenými daty ukládány a aplikace pro práci s těmito soubory. Kapitola 4 se věnuje jednotlivým tunelovacím protokolům, se kterými se bude v této práci pracovat. Také je zde popsán základní princip tunelování.

Kapitola 5 se věnuje návrhu samotné aplikace. Je zde popsán základní princip aplikace a jednotlivé části, do kterých byla aplikace rozdělena. Kapitola 6 se věnuje implementaci aplikace pro extrahování tunelovaných dat. V této kapitole jsou popsány vytvořené třídy a použité knihovny, které byly využity pro implementaci požadované aplikace. Poslední kapitola 7 se věnuje vytvářením testovacích dat pro cílovou aplikaci. Jsou zde popsány nástroje, které byly k tomuto účelu použity, použitá testovací topologie pro vytváření tunelů a prostředí, ve kterém byly tyto tunely vytvářeny.

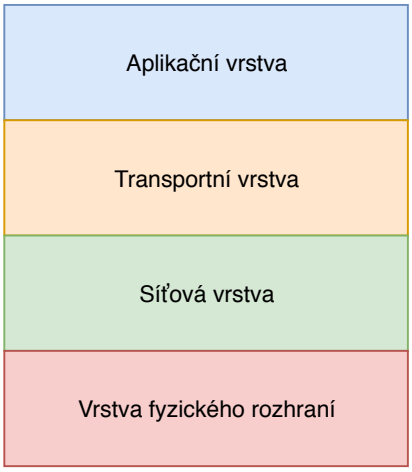
Kapitola 2

Síťová architektura TCP/IP

Tato kapitola popisuje síťovou architekturu modelu TCP/IP a základní protokoly, které tento model představuje. Informace v této kapitole jsou volně převzaty z knížek *Velký průvodce protokoly TCP/IP a systémem DNS* [5] a *Síťové aplikace a jejich architektura* [7]. Obsahem kapitoly jsou také hlavičky protokolů, pomocích kterých se budou detekovat tunelovací protokoly.

TCP/IP je rodina protokolů, která slouží ke komunikaci v počítačové síti a tvoří základ internetu. Rodina protokolů TCP/IP obsahuje více než stovku různých protokolů. V této kapitole budou popsány jen základní protokoly, které se v této architektuře objevují. Protokoly jsou soubory pravidel, podle kterých probíhá komunikace mezi klienty. Internetové protokoly bývají standardizované v dokumentech RFC.

Architektura je rozdělena do 4 vrstev. Každá vrstva využívá služby vyšší vrstvy a zároveň poskytuje služby pro vrstvy nižší. Architektura je znázorněna na obrázku 2.1.



Obrázek 2.1: Architektura TCP/IP.

2.1 Vrstva fyzického rozhraní

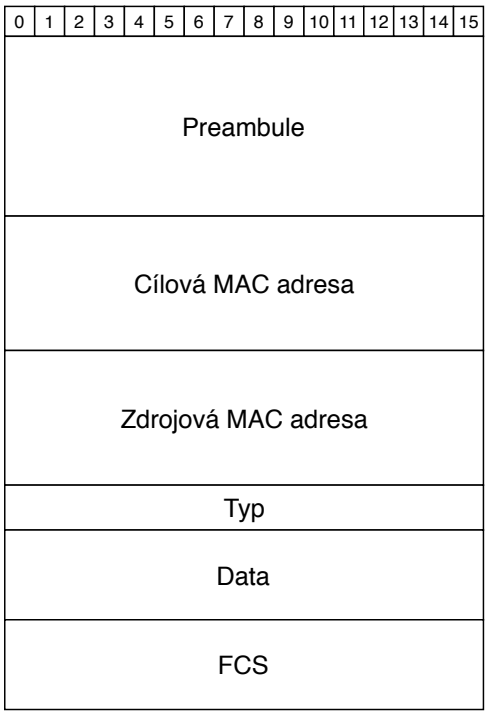
Jedná se o nejnižší vrstvu v TCP/IP modelu a má na starost přenos rámců mezi dvěma přímo připojenými zařízeními. Tato vrstva popisuje, jakým způsobem dochází k přístupu na fyzické médium, jako jsou například koaxiální kabely a optické vlákna, adresování a kódování.

Tato vrstva není konkrétně specifikovaná, protože záleží na použité přenosové technologie. Tou může být Ethernet, Token Ring, telefonní linka X.25 a další. Nezávislosti síťové vrstvy na nějaké konkrétní technologii umožňuje architektuře TCP/IP rychlou adaptaci na nové technologie. Dnes nejznámější a nejpoužívanější přenosová technologie je Ethernet.

Fyzická zařízení na této vrstvě jsou identifikovaná pomocí MAC adresy. Jedná se o 48 bitovou adresu, zapisovanou šesticí dvojciferných hexadecimálních čísel oddělená dvojtečkou. Každému zařízení je MAC adresa přidělena při výrobě.

2.1.1 Ethernet

Tento protokol zapouzdřuje data do rámců. Rámce jsou tvořeny tzv. preambulí, ethernetovou hlavičkou, samotnými daty a kontrolním součtem. Formát ethernetového rámce je znázorněn na obrázku 2.2.



Obrázek 2.2: Ethernetový rámec

Popis položek ethernetového rámce:

- **preambule** – slouží k synchronizaci hodin příjemce;
- **cílová MAC adresa** – MAC adresa cílového síťového rozhraní;
- **zdrojová MAC adresa** – MAC adresa zdrojového síťového rozhraní;
- **typ** – určuje následující protokol;
- **FCS** – cyklický redundantní součet, umožňující odhalení poškození rámce.

2.2 Síťová vrstva

Síťová vrstva má na starost vše, co se týká vytváření, adresování a směrování dat, které se mají přenášet po síti na kterékoliv zařízení. Přenášená data zabaluje do IP datagramů, které obsahují všechny informace potřebné k doručení dat na koncovou stanici. Data se snaží doručovat tou nejvhodnější cestou. Protokoly pracující na síťové vrstvě jsou například IP, ARP, ICMP, IGMP.

2.2.1 IPv4

Internetový protokol verze 4 (IPv4) [9] je nespojovaný protokol nezajišťující spolehlivost, který má na starost adresaci a směrování datagramů mezi dvěma zařízeními. Protokol definuje základní jednotku dat, která jsou přenášena na úrovni síťové vrstvy, nazvanou IP datagram. Protokol definuje vnitřní formát těchto datagramů a další aspekty nespolehlivé a nespojované služby, jako jsou například podmínky toho, kdy mají být zahazovány pakety, kdy se mají generovat chybová hlášení a jak mají tato hlášení vypadat.

IPv4 datagramy mohou být při přenášení přes síť fragmentovány do menších částí. K této fragmentaci dochází, aby bylo možné datagramy přenést přes každou část přenosové trasy. Každá tato část má totiž definovanou svoji maximální přenosovou jednotku (MTU). Tato jednotka určuje maximální velikost IP datagramu, který je možný poslat přes dané síťové rozhraní. Hlavička protokolu IPv4 má obvykle 20 bytů a je znázorněna na obrázku 2.3.

Obrázek 2.3: Hlavička IPv4.

Popis položek z hlavičky protokolu IPv4:

- **verze** – verze IP, v tomto případě 4;
- **IHL** – délka hlavičky vyjádřená jako počet bajtů vydělených číslem 4;
- **typ služby** – obsahuje pole značek pro mechanismy zajišťující služby s definovanou kvalitou (QoS);
- **celková délka** – délka IP datagramu (IP hlavička a data);
- **identifikace** – jednoznačný identifikátor, sloužící k identifikaci datagramu, do které patří tento fragment. Každý fragment datagramu má stejný identifikátor;
- **příznaky** – celkem tři příznaky, které slouží k řízení fragmentace;
- **offset fragmentu** – pozice v původním datagramu, na které začíná tento fragment;

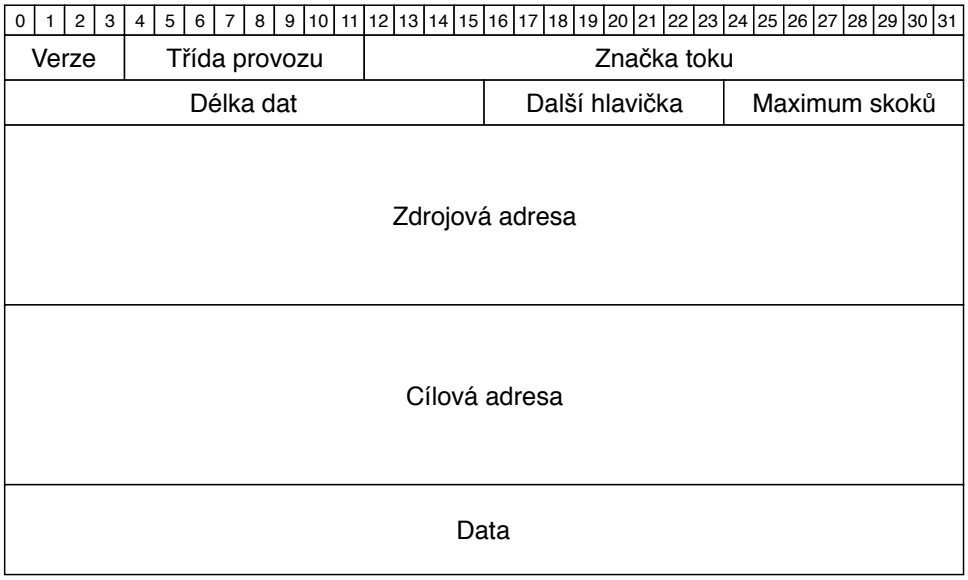
- **TTL** – délka života datagramu. Jedná se o ochranu proti zacyklení, pokud dosáhne hodnoty 0, datagram bude zahozen;
- **protokol** – určuje protokol, který následuje po IP hlavičce;
- **kontrolní součet hlavičky** - kontrolní součet vypočítaný pouze z IP hlavičky;
- **zdrojová adresa** – IPv4 adresa odesílatele;
- **cílová adresa** – IPv4 adresa příjemce.

2.2.2 IPv6

Internetový protokol verze 6 (IPv6) [2] dnes nahrazuje již nedostačující protokol IPv4. Hlavním důvodem přechodu z IPv4 na IPv6 je nedostatek adresovatelných IPv4 adres, které již byly oficiálně vyčerpány roku 2011. Mezi další výhody tohoto protokolu patří zjednodušení při přidělování adres a zjednodušení přečíslování při změně poskytovatele, kdy stačí na směrovači pouze změnit prefix sítě.

Hlavička protokolu IPv6 byla oproti protokolu IPv4 změněna. Počet položek v hlavičce byl zredukován na 8, velikost hlavičky byla nicméně zvýšena o dvojnásobek na 40 bytů. Hlavním důvodem větší velikosti je velikost zdrojové a cílové adresy, které dohromady mají 32 bytů. Hlavička je znázorněna na obrázku 2.4.

IPv6 může být také rozšířen o další hlavičky. Tyto hlavičky se nazývají rozšiřující hlavičky. Pokud paket obsahuje rozšiřující hlavičku, je typ rozšiřující hlavičky uveden v hlavičce IPv6 v poli, které obsahuje informaci o následující hlavičce. Každá tato rozšiřující hlavička má svou vlastní informaci o následující hlavičce. Rozšiřující hlavičky mohou například sloužit k směrování, šifrování a autentizaci.



Obrázek 2.4: Hlavička IPv6.

Popis položek z hlavičky protokolu IPv6:

- **verze** – verze IP, v tomto případě 6;
- **třída provozu** – umožňuje specifikovat požadavky na vlastnosti sítě, který daný datagram má;
- **značka toku** – pomocí značky toku se identifikuje tok, který je chápán jako proud datagramů od stejného odesílatele ke stejnému příjemci se stejnými vlastnosti;
- **délka dat** – délka přenášených dat;
- **další hlavička** – určuje typ dat, které následují po IP hlavičce. Může jít buď o další protokol nebo o rozšiřující hlavičku;
- **maximum skoků** – obdoba TTL u IPv4;
- **zdrojová adresa** – IPv6 adresa odesílatele;
- **cílová adresa** – IPv6 adresa příjemce.

2.3 Transportní vrstva

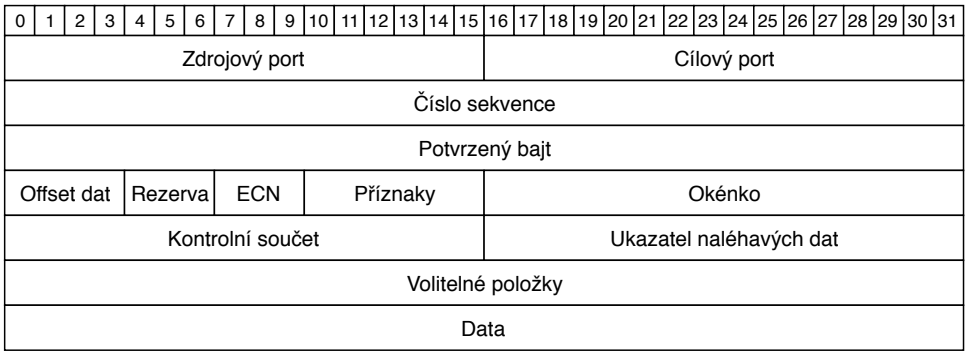
Transportní vrstva má na starost přenos dat mezi aplikacemi na zdrojovém a cílovém počítači. Protokoly transportní vrstvy rozdělují přenášená data na menší části, které se posílají po síti. Tyto části nazýváme pakety. Nejčastějšími příklady protokolů, které pracují na transportní vrstvě jsou protokoly UDP a TCP.

2.3.1 TCP

Transmission Control Protocol (TCP) [10] je nejpoužívanější protokol transportní vrstvy v rodině protokolů TCP/IP. Jedná se o spolehlivou a spojovanou doručovací službu a tedy musí byt před výměnou dat mezi klienty ustanoveno spojení. Po ustavení spojení si mohou klienti mezi sebou posílat data v obou směrech. Spolehlivost protokolu se zajišťuje pomocí číslování jednotlivých segmentů a kontrolou jejich přijetí. Pokud není některý ze segmentů přijat nebo je poškozen, požádá se o jeho opětovné odeslání. Data se znovu odešlou také v případě, pokud již vypršel čas, kdy měla být přijata zpráva o doručení dat. Integrita přenášených dat je zajištěna kontrolním součtem.

Pro rozlišování jednotlivých aplikací používá protokol TCP čísla portů. Číslo portu může nabývat hodnot 1 až 65535, přičemž porty 1 až 1023 jsou vyhrazené pro nejběžnější služby. Pokud má nějaká komunikace na síti shodnou pětici údajů a to cílovou a zdrojovou IP adresu, cílový a zdrojový port a stejný protokol, patří všechny tyto pakety do jednoho síťového toku.

Hlavička protokolu TCP má obvykle 20 bytů a je znázorněna na obrázku 2.5. Pro výpočet kontrolního součtu se v hlavičce používá tzv. pseudohlavička. Tato hlavička se pomyslně připojí k vlastnímu datagramu, avšak není ve skutečnosti odesílatelem k příjemci odesílána. Pseudohlavička obsahuje IP adresy odesílatele a příjemce, identifikátor protokolu a velikost datagramu bez této pseudohlavičky.



Obrázek 2.5: Hlavička TCP.

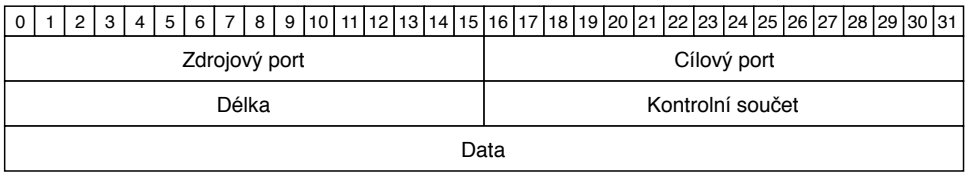
Popis položek z hlavičky protokolu TCP.

- **zdrojový port** – číslo zdrojového portu;
- **cílový port** – číslo cílového portu;
- **číslo sekvence** – pořadové číslo prvního datového bytu v tomto segmentu;
- **potvrzený bajt** – hodnota dalšího pořadového čísla, který odesílatel očekává;
- **offset dat** – označení, kde začínají data;
- **okénko** – množství dat v bajtech, které je potvrzováno najednou;
- **kontrolní součet** – vypočítaný kontrolní součet z TCP datagramu a pseudohlavičky;

2.3.2 UDP

User Datagram Protocol (UDP) [8] je doručovací služba, běžící na transportní vrstvě. Na rozdíl od protokolu TCP je UDP nespojovaná a nespolehlivá služba. UDP tedy před výměnou nenavazuje spojení a nijak nekontroluje, zda-li vůbec a v jakém pořadí byly data doručeny.

Stejně jako protokol TCP i UDP používá pro rozlišování jednotlivých aplikací čísla portů. Protokol UDP má svoji vlastní sadu portů, které jsou nezávislé. Hlavička je oproti protokolu TCP menší. V hlavičce se přenášejí pouze informace o zdrojovém a cílovém portu, délka UDP paketu včetně dat a kontrolní součet. Stejně jako u protokolu TCP, i zde se pro výpočet kontrolního součtu používá pseudohlavička. Velikost hlavičky je 8 bytů a je znázorněna na obrázku 2.6.



Obrázek 2.6: Hlavička UDP.

Popis položek UDP hlavičky:

- **zdrojový port** – číslo zdrojového portu;

- **cílový port** – číslo cílového portu;
- **délka** – délka UDP datagramu (UDP hlavička + data);
- **kontrolní součet** – vypočítaný kontrolní součet z UDP datagramu a pseudohlavičky.

2.4 Aplikační vrstva

Aplikační vrstvu tvoří procesy a aplikace, které komunikují po síti. Tyto procesy a aplikace definují protokoly, které používají k výměně dat. Většina protokolů aplikační vrstvy vychází z modelu klient/server. Klient žádá o konkrétní služby a je iniciátorem veškeré komunikace, zatímco server poskytuje své služby pouze na základě žádosti klienta. Způsob komunikace mezi klientem a serverem si určuje každý protokol sám.

Na aplikační vrstvě pracuje velké množství protokolů. Mezi nejznámější patří například protokoly FTP, POP3, DHCP, BitTorrent, IMAP a další. Pro tuto práci je tato vrstva dále nezajímavá, protože na tunelování nemá žádný vliv.

Kapitola 3

Zachytávání a zpracování dat

Tato kapitola je zaměřená na způsoby zachytávání paketů na síti a práci s nimi. V této kapitole jsou představeny základní metody, jakými se dají zachytávat data na síti a na způsoby zpracování již zachycených dat, respektive na aplikaci, která k tomuto účelu slouží. Informace uvedené v této kapitole jsou volně přejaty z knihy *Practical Packet Analysis* [11].

3.1 Zachytávání dat

Existuje několik způsobů jak zachytávat pakety, které jsou přenášeny přes síť. V této kapitole jsou popsaný dva nejpoužívanější způsoby zachytávání paketů v síti a to zrcadlení portu a zařízení tap.

Pro zachytávání paketů v síti je nutné vlastnit síťovou kartu, která umožňuje přechod do promiskuitního režimu. V promiskuitním režimu dokáže síťová karta zachytávat všechny data, která procházejí sítí. V dnešní době umožňují přechod do promiskuitního režimu téměř všechny síťově karty. Jediná situace, kdy nepotřebujeme mít přepnutou kartu v promiskuitním režimu, je situace, kdy chceme sledovat komunikaci pouze na našem zařízení.

Síťová karta, která není v promiskuitním režimu, přijímá pouze data, která jsou určena pro její MAC adresu nebo pokud jde o zprávy typu broadcast či multicast. Zprávy typu broadcast jsou totiž posílány všem účastníkům v dané síti. Zprávy typu multicast jsou zase posílány všem účastníkům dané multicastové skupiny.

3.1.1 Zrcadlení portu

Zrcadlení portu je asi nejjednodušší způsob zachytávání provozu cílového zařízení na síti. Nastavuje se v rozhraní přepínače a přepínač jej musí podporovat. Pro použití zrcadlení portu je také nutné mít na přepínači ještě jeden volný port, kam lze připojit sledovací zařízení.

Při konfiguraci zrcadlení portu se nastaví, který port bude zrcadlen na určitý port. Na tento port se potom připojí sledovací zařízení. Sledovací zařízení poté může vidět všechny pakety, které zařízení, připojené k zrcadlenému portu, přijímá i odesílá. Jedná se o jednoduchou metodu zachytávání dat, avšak ne úplně spolehlivou, protože může při ní docházet ke ztrátě dat.

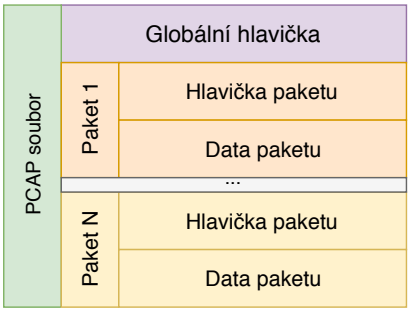
3.1.2 Zařízení „tap“

Zařízení „tap“ je zařízení, které se umístí mezi dva body v kabeláži, kde chceme zachytávat pakety. Jedná se o speciální zařízení, určené k zachytávání paketů.

Existují dva typy těchto zařízení a to agregované a neagregované. Hlavní rozdíl mezi těmito typy je počet portů. Agregované zařízení mají porty 3, neagregované 4. Neagregované zařízení mají na rozdíl od agregovaných 2 monitorovací porty. Jeden monitorovací port slouží pro odposlouchávání komunikace v jednom směru a druhý port pro odposlouchávání komunikace v opačném směru. Agregované zařízení mají pouze jeden monitorovací port, který slouží k zachytávání paketů v obou směrech.

3.1.3 PCAP soubory

Soubory typu PCAP jsou datové soubory se zachycenou síťovou komunikací. Jedná se o nejběžnější formát, do kterého se ukládá zachycená síťová komunikace. PCAP soubory vždy začínají globální hlavičkou, za kterou následuje 0 nebo více záznamů pro každý zachycený paket. Struktura této globální hlavičky je znázorněna ve výpise 3.1. Záznam paketu obsahuje hlavičku paketu, obsahující základní informace o paketu a samotná zachycená data paketu. Struktura hlavičky paketu je znázorněna ve výpise 3.2. Formát souboru PCAP je znázorněn na obrázku 3.1.



Obrázek 3.1: Formát souboru PCAP

Struktura globální hlavičky:

```
1 typedef struct pcap_hdr_s {
2     guint32 magic_number; /* Magické číslo */
3     guint16 version_major; /* Hlavní číslo verze */
4     guint16 version_minor; /* Vedlejší číslo verze */
5     gint32 thiszone; /* Korekce lokální času ku GMT */
6     guint32 sigfigs; /* Přesnost časového razítka */
7     guint32 snaplen; /* Maximální velikost zachycených paketů v~bajtech */
8     guint32 network; /* Typ síťového zařízení */
9 } pcap_hdr_t;
```

Výpis 3.1: Struktura globální hlavičky

Struktura hlavičky paketu:

```
typedef struct pcaprec_hdr_s {
    guint32 ts_sec; /* Sekundy časového razítka*/
    guint32 ts_usec; /* Mikrosekundy časového razítka */
    guint32 incl_len; /* Velikost paketu uloženého v~souboru v~bajtech */
    guint32 orig_len; /* Skutečná délka paketu */
} pcaprec_hdr_t;
```

Výpis 3.2: Struktura hlavičky paketu

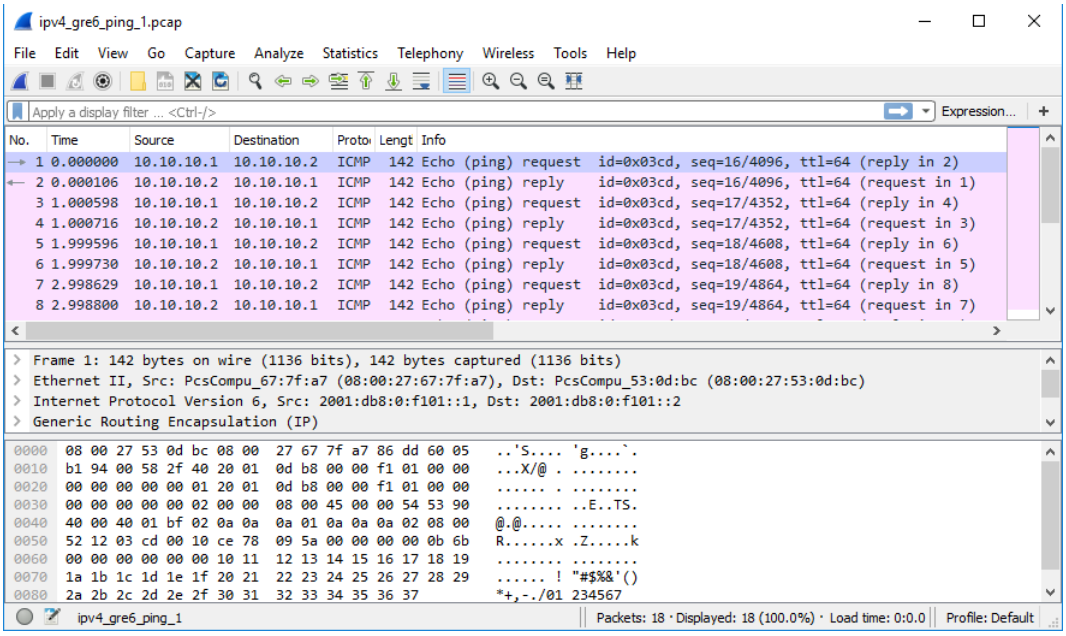
3.2 Zpracování dat

Tato část popisuje aplikace, které slouží k analýze zachycených paketů. K práci se zachycenými daty jsem používal aplikaci Wireshark a její odnož TShark.

3.2.1 Wireshark

Wireshark je aplikace určená pro zachytávání a analýzu paketů. Aplikace je multiplatformní a nabízí přehledné uživatelské rozhraní. Wireshark dokáže analyzovat zachycenou síťovou komunikaci do nejmenších detailů a vytvářet různé statistiky a data. Wireshark dále umožňuje přepnout kartu do promiskuitního režimu, aby uživatel mohl sledovat všechny pakety, které tečou v síti. Zachycené pakety lze také ukládat do několik různých formátu, z nichž nejčastější jsou PCAP soubory.

Wireshark dnes podporuje více jak 850 různých protokolů a každou aktualizací přibývají další nové protokoly. Nesmírnou výhodou je otevřený zdrojový kód. Díky tomu může přidávat podporu nějakého protokolu prakticky každý, kdo má patřičné programátorské schopnosti. Vzhled aplikace je zobrazen na obrázku 3.2.



Obrázek 3.2: Vzhled aplikace Wireshark

3.2.2 TShark

TShark je odlehčená verze Wiresharku, která neobsahuje grafické uživatelské rozhraní. Tato aplikace se ovládá pouze přes příkazový řádek a nabízí podobnou funkcionalitu jako Wireshark s tím, že některé funkce nejsou dostupné. Umožňuje tedy například zachytávání paketů na síťovém rozhraní, čtení PCAP souborů a ukládání zachycené komunikace do různých souborových formátů. Aplikace je ovládána pomocí přepínačů z příkazové řádky.

Příklady použití

Příklad příkazu pro zachytávání paketů:

```
$ tshark -i sitove_zarizeni -w nazev_souboru.pcap
```

Příkaz pro čtení PCAP souboru:

```
$ tshark -r nazev_souboru.pcap
```

Příklad příkazu pro převod PCAP souboru do formátu PDML (XML)

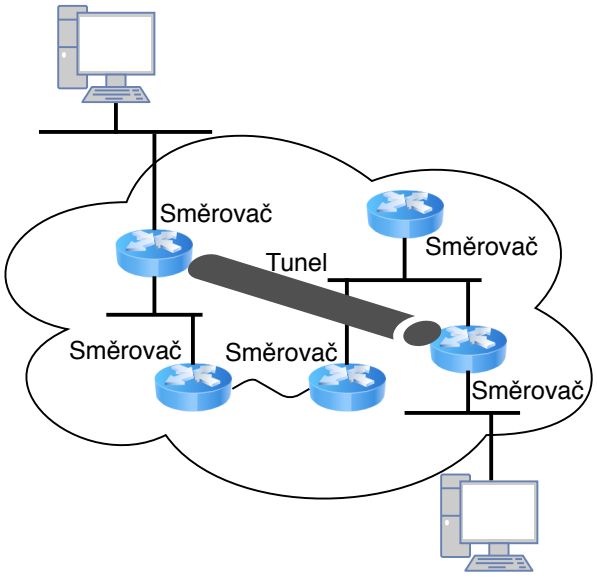
```
$ tshark -r nazev_souboru.pcap -T pdml > vystup.pdml
```

Kapitola 4

Tunelovací protokoly

Pojmem tunelování se v počítačových sítích rozumí situace, kdy jsou pakety jednoho síťového protokolu vkládány jako data do jiného síťového protokolu. Tunelování umožňuje například přenášet data přes nekompatibilní sítě, obcházet některá administrativní omezení sítě a poskytovat zabezpečené spojení přes nezabezpečenou síť. Základní princip tunelování je zobrazen na obrázku 4.1.

Tunelovací protokoly fungují na principu zapouzdření datové části paketu tunelovaného protokolu do protokolu jiného. Tunel zapouzdřuje užitečná data, vnitřní paket určený k doručení, do vnějšího paketu. Ten se poté pošle skrze tunel. Mezilehlé routery jej směrují jako vnější paket, takže jej předají do cílové sítě, kde je vnější paket odebrán a původní paket je směrován ke svému cíli. Při tunelování obvykle dojde k narušení pořadí vrstev, protože se data některé vrstvy modelu zapouzdřují do protokolu, běžící na stejné nebo jiné vrstvě.



Obrázek 4.1: Základní princip tunelování. Na obrázku je znázorněn princip vytvoření tunelu mezi dvěma směrovači.

4.1 GRE

Generic Routing Encapsulation (GRE) [4] je tunelovací protokol, který pracuje na síťové vrstvě. Protokol byl vyvinutý společností Cisco a umožňuje zapouzdření velké škály protokolů síťově vrstvy uvnitř virtuálního přímého spojení přes IP protokol. GRE protokol poskytuje bezstavové privátní spojení, které je ovšem nezabezpečené. Protokol totiž sám o sobě nepoužívá žádné šifrování dat.

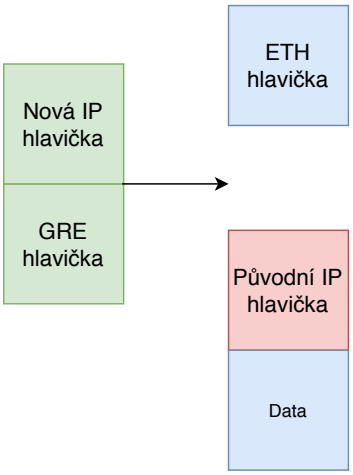
Při použití GRE tunelu přidáme do paketu dvě hlavičky. První přidanou hlavičkou je vnější IP hlavička, která je vložena před původní IP hlavičku. Mezi ně je poté vložena GRE hlavička. Standardní velikost hlavičky bez volitelných položek jsou 4 byty. Hlavička je znázorněna na obrázku 4.2. Protokol GRE definuje celkově dvě verze hlaviček. První verze (verze 0) je určena pro klasický samotný tunel. Druhá verze (verze 1) je určena pro protokol PPTP o kterém pojednává sekce 4.2.

Obrázek 4.2: GRE hlavička, verze 0

Popis vybraných položek GRE hlavičky:

- **C, R, K, S** – příznaky, které udávají informace o tom, zda se v hlavičce nachází dané volitelné položky;
- **rekurze** – obsahuje údaj o tom, kolik dalších zapouzdření je povoleno;
- **příznaky** – rezervované bity, musí byt nastaveny na 0;
- **verze** – verze GRE hlavičky: 1, pokud se jedná o protokol PPTP, jinak 0;
- **protokol** – označení protokolu, který je zapouzdřen;
- **kontrolní součet** – vypočítaný kontrolní součet z GRE hlavičky a dat;
- **klíč** – obsahuje číslo, které bylo vloženo společně se zapouzdřením. Přijímací strana je může použít k ověření zdroje paketu;
- **sekvenční číslo** – obsahuje číslo, které bylo vloženo společně se zapouzdřením. Přijímací strana je může použít k určení pořadí, v jaké byly pakety byly pakety odeslány.

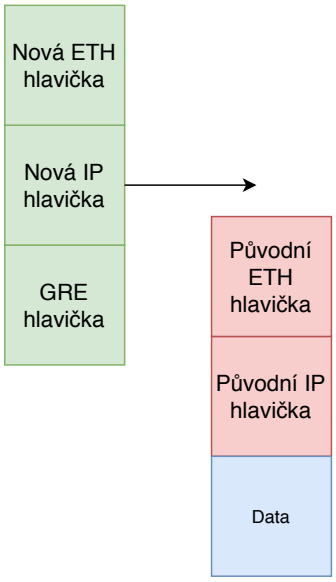
Detekování přítomnosti GRE tunelu v paketu se provádí pomocí informaci v IP hlavičce, která uvádí, který protokol následuje za IP hlavičkou. Pokud se v této části nachází hodnota 47, následuje za IP hlavičkou právě GRE hlavička. V GRE hlavičce se následně obdobně ověří, zda se nejedná o PPTP tunel či o GRETAP tunel. To se provádí pomocí informace o verzi hlavičky. Pokud je verze hlavičky 0, jedná se o GRE či GRETAP tunel. Jestli je verze hlavičky 1, jedná se o PPTP. Podoba GRE paketu je znázorněna na obrázku 4.3.



Obrázek 4.3: Paket obsahující GRE zapouzdření

4.1.1 GRETAP

GRE protokol nám umožňuje také zapouzdřit ethernetové rámce, namísto IP hlavičky. V takovém případě nazýváme tento tunel GRETAP. GRETAP nám tedy nepřidá do paketu pouze IP hlavičku a GRE hlavičku, ale i další ethernetovou hlavičku. Toho se využívá při detekci GRETAP tunelu. Ten se detekuje tak, že GRE hlavička musí být ve verzi 0 a zároveň musí být následující protokol ethernet. Podoba GRETAP paketu je znázorněna na obrázku 4.4.

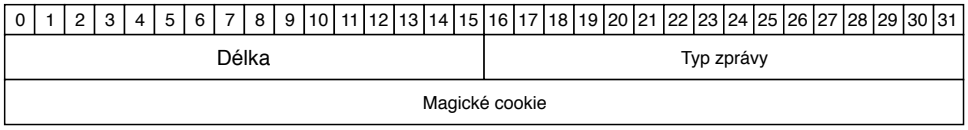


Obrázek 4.4: Paket obsahující GRETAP zapouzdření

4.2 PPTP

Point-to-Point Tunneling Protocol (PPTP) [3] je metoda pro realizaci virtuálních privátních sítí (VPN). Protokol pro řízení používá protokol TCP a pro zapouzdření PPP paketů používá protokol GRE.

PPTP používá ke komunikace ve výchozím nastavení TCP spojení na portu 1723. Toto spojení je používáno k iniciování a řízení GRE tunelu. PPTP používá pro tunelování pomocí GRE nestandardní upravenou verzi hlavičky. Do hlavičky byly zavedeny nové položky: potvrzovací číslo, id volání a délka dat. Do hlavičky také přibyl jeden příznak, pro indikaci, zda je v hlavičce potvrzovací číslo uvedeno. Z hlavičky naopak ubyly informace o kontrolním součtu, klíč a offset. Hlavička samotného protokolu PPTP, která se používá při iniciování a řízení spojení, je zobrazena na obrázku 4.5.

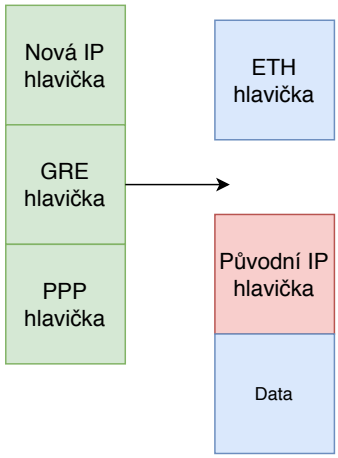


Obrázek 4.5: PPTP hlavička

Popis položek z hlavičky PPTP:

- **délka** – velikost PPTP zprávy v bytech;
- **typ zprávy** – může nabývat hodnot 1 nebo 2. Zprávy typu 2 nicméně nejsou definovány;
- **magické cookie** – hodnota tohoto pole je vždy nastavena na 0x1A2B3C4D.

Detekování přítomnosti PPTP tunelu se provádí obdobně, jako u protokolu GRE. Stejně jako GRE tunel, musí i PPTP tunel za IP hlavičkou obsahovat GRE hlavičku. Zároveň se musí ještě ověřit, že se jedná o upravenou GRE hlavičku, která se používá právě při tunelování pomocí PPTP. To se ověří podle informace v GRE hlavičce, kde je uložena informace o verzi hlavičky. V případě PPTP musí být hodnota tohoto pole nastavena na 1. Podoba PPTP zapoudření je znázorněna na obrázku 4.6.

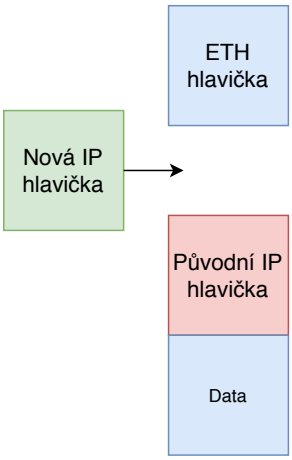


Obrázek 4.6: Paket obsahující PPTP zapouzdření

4.3 IPIP

IP in IP (IPIP) je tunelovací protokol, který zapouzdřuje IP paket jiným IP paketem. Po zapouzdření se přidá nová IP hlavička, která bude mít odlišené hodnoty zdrojové a cílové adresy. V těchto adresách je uveden vstupní a výstupní bod tunelu.

IPIP tunel je možné realizovat libovolnou kombinací IP protokolů. IPv4 protokol může zapouzdřovat IPv6 protokol, stejně tak i naopak. Možné je i zapouzdřovat IP protokol stejným typem IP protokolu. Způsob, jakým jsou data zapouzdřena je znázorněn na obrázku 4.7.



Obrázek 4.7: IPIP zapouzdření

Detekce IPIP tunelu se provádí pomocí informace v IP hlavičce, která uvádí, který protokol se nachází za touto IP hlavičkou. Pokud se v tomto poli nachází hodnota 4 (IPv4) či 41 (IPv6), tak se v tomto paketu nachází IPIP zapouzdření.

4.4 L2TP

Layer 2 Tunneling Protocol (L2TP) [12] je tunelovací protokol, určený k použití pro podporu VPN. Tento protokol má původ ve dvou starších tunelovacích protokolech. Vychází z protokolů Layer 2 Forwarding Protocol (L2F) a z protokolu PPTP.

Celý L2TP paket je zapouzdřen do protokolu UDP. Protokol sám o sobě neposkytuje žádné šifrování ani zabezpečení. Z toho důvodu se často používá ve spolupráci s protokolem IPsec, který tyto vlastnosti zajišťuje. Tato kombinace protokolů se nazývá L2TP/IPsec. Hlavička protokolu je zobrazena na obrázku 4.8.

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
T	L	x		S	x	O	P	x				Verze				Délka (volitelné)															
ID Tunelu																ID relace													
Ns (volitelné)																Nr (volitelné)													
Velikost offsetu (volitelné)																Vyplnění offsetu (volitelné)													
Data																															

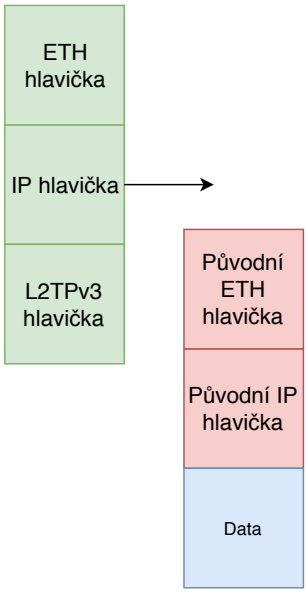
Obrázek 4.8: L2TP hlavička

Popis položek z hlavičky L2TP:

- **T, L, x, S, x, O, P** – příznaky a rezervované bity;
- **verze** – verze protokolu;
- **délka** – velikost zprávy v bajtech;
- **ID tunelu** – jednoznačný identifikátor tunelu pro řízení spojení;
- **ID relace** – jednoznačný identifikátor relace v tunelu;
- **ns** – pořadové číslo pro data nebo řídící zprávu;
- **nr** – pořadové číslo, které je očekávané v příští řídící zprávě, která má být přijata;
- **velikost offsetu** – určuje počet bajtů za L2TP hlavičkou, kde se očekává začátek užitečných dat;
- **vyplnění offsetu** – vyplnění pole offsetu do velikost 32 bitů.

V roce 2005 byla vydána nová verze protokolu L2TPv3 [6]. Tato verze přinesla nové zabezpečovací funkce, upravené zapouzdření a možnost přenášet data přes jiné sítě, než přes sítě IP. Dále přibyla možnost zapouzdřit pouze protokol IP bez nutnosti zapouzdření UDP.

Možnosti detekování L2TP tunelu závisí na zvolené verzi L2TP. Jestliže je použita standardní verze protokolu, je potřeba ověřovat, zda se za protokolem UDP nenachází L2TP hlavička. Stejným způsobem lze detekovat i L2TP tunel ve verzi 3, který používá zapouzdření právě protokolu UDP. Pokud zapouzdřuje protokol IP, detekuje se L2TP tunel právě pomocí IP hlavičky. V IP hlavičce je uložena informace o následujícím protokolu. Pokud bude hodnota této informace rovna 115, nachází se v paketu L2TP zapouzdření. Podoba L2TPv3 paketu, který zapouzdřuje IP, je znázorněna na obrázku 4.9.

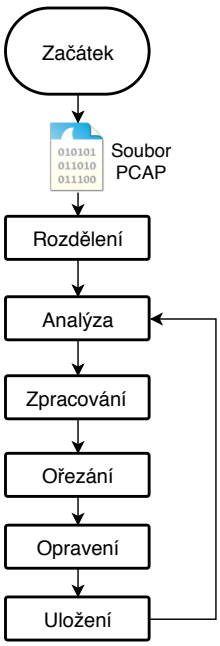


Obrázek 4.9: Paket obsahující L2TPv3 IP zapouzdření

Kapitola 5

Návrh extraktoru

Tato kapitola se věnuje návrhu aplikace, která je cílem této bakalářské práce. Cílem je extrakce tunelovaného provozu. Tohoto cíle se dosáhne za pomocí odstraňování různých vrstev v paketu. Návrh aplikace je rozdělen do několika jednotlivých částí. Celkem je těchto částí šest a každá část zastává jinou činnost při extrakci tunelovaných dat. Základní princip aplikace a její části jsou znázorněny na obrázku 5.1.



Obrázek 5.1: Hlavní část programu. Aplikace je rozdělena celkem na 5 částí.

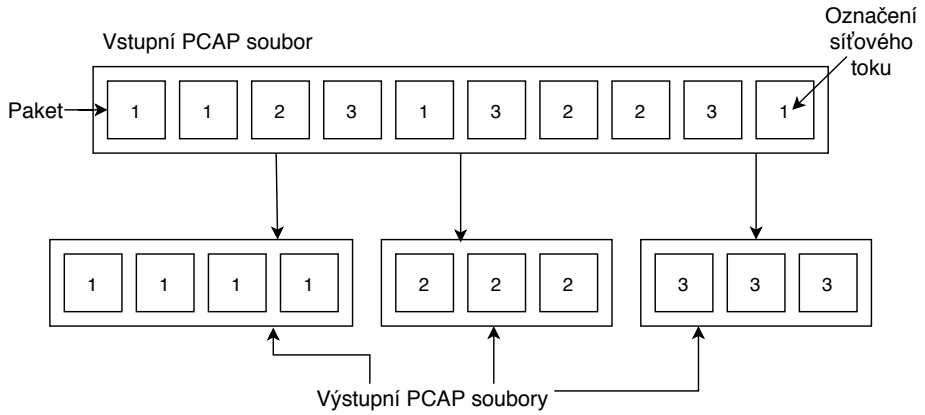
Vstupem aplikace je PCAP soubor. První věc, kterou bude aplikace s daným souborem provádět je rozdělení souboru na jednotlivé zachycené síťové toky do samostatných PCAP souborů. Činnost této části aplikace je popsána v části 5.1. Po rozdělení souboru se bude každý síťový tok zpracovávat samostatně. Každý takový PCAP soubor projde analýzou, kde se bude zjišťovat, zda daný síťový tok je tunelovaný. Činnost analyzátoru je popsána v části 5.2. Pokud analyzátor detekuje tunelovaný provoz, daný PCAP se zpracuje, tzn. dojde k označení částí, které se mají z paketu odstranit. Tomuto procesu se věnuje část 5.3. Samotný proces odstraňování částí paketu je popsán v části 5.4.

Cíle aplikace se dosáhnou za pomocí odstraňování různých vrstev paketu. Při odstranění některé vrstvy ale můžeme narazit na několik problémů. Vrstvy modelu jsou totiž vzájemně propojeny a odstraněním některé z nich může porušit správnost informací v ostatních vrstvách. Typické informace, které již nemusí při odstranění platit jsou například délka datagramu, následující protokol a kontrolní součet. Po odstranění některé vrstvy z paketu je nutné tyto informace opravit, aby tyto informace byli i nadále korektní. Způsob opravování je popsán v části 5.5.

Po každém odstraněném zapouzdření se vytvoří nový PCAP soubor, obsahující datový tok bez daného zapouzdření. Takto nově vytvořený soubor se opět zanalyzuje pro případ, že původní soubor obsahoval více zapouzdření. Aplikace tedy bude z PCAP souboru postupně odstraňovat zapouzdření a vytvářet nové PCAP soubory. Způsob vytváření nových souborů je popsán v části 5.6.

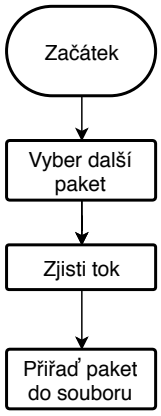
5.1 Předzpracování

Předzpracování je první věc, která se provede se vstupním PCAP souborem. Cílem této části aplikace je rozdělit PCAP soubor, který je vstupem programu, do více PCAP souborů. Soubor bude rozdělen tak, aby každý síťový tok byl v samostatném souboru. Každé takto vytvořené soubory se poté budou zpracovávat samostatně. Způsob rozdělení souboru je znázorněn na obrázku 5.2.



Obrázek 5.2: Znázornění, jak jednotlivé toky budou ze vstupního souboru rozděleny. Každé číslo v obrázku značí číslo toku.

Tato část aplikace bude procházet vstupní soubor paket po paketu a jednotlivé pakety přidělovat k daným síťovým tokům. Výstupem této části aplikace bude *1 až N* PCAP souborů, v závislosti na počtu síťových toků, které vstupní soubor obsahuje. Základní princip této části aplikace je znázorněn na obrázku 5.3.



Obrázek 5.3: Jednotlivé části předzpracování, kde se provádí rozdělení vstupního souboru na menší části, kdy bude každý síťový tok uložen v jednom souboru.

Pro určení síťového toku bude nutné ze vstupního souboru získat informace o síťovém toku, konkrétně zdrojové a cílové IP adresy, protokol a popřípadě čísla zdrojového a cílového portu. Tyto informace se budou získávat po převedení vstupního souboru do formátu PDML. Pro zjištění těchto informací je potřeba v PDML souboru nalézt v každém paketu IP hlavičku a hlavičku buď UDP nebo TCP protokolu, pokud je daný paket obsahuje. Z těchto hlaviček je poté nutno vytáhnout tyto informace. Jak jsou tyto informace uložené v PDML souboru je znázorněno na obrázku 5.4.

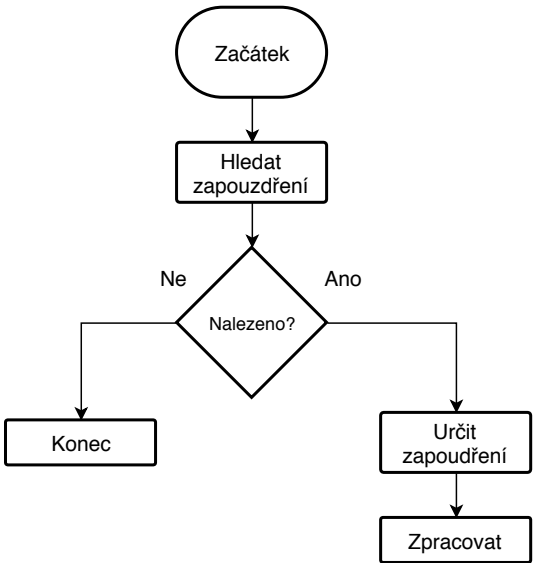
```
  <field name="ip.frag_offset" showname="Fragment offset: 0" size="2" pos="40" show="0" value="4000"/>
  <field name="ip.ttl" showname="Time to live: 64" size="1" pos="42" show="64" value="40"/>
  <field name="ip.proto" showname="Protocol: TCP (6)" size="1" pos="43" show="6" value="06"/>
  <field name="ip.checksum" showname="Header checksum: 0x9a8e [validation disabled]" size="2" pos="44" show="0x0
  <field name="ip.checksum.status" showname="Header checksum status: Unverified" size="0" pos="44" show="2"/>
  <field name="ip.src" showname="Source: 10.10.10.1" size="4" pos="46" show="10.10.10.1" value="0a0a0a01"/>
  <field name="ip.addr" showname="Source or Destination Address: 10.10.10.1" hide="yes" size="4" pos="46" show="1
  <field name="ip.src_host" showname="Source Host: 10.10.10.1" hide="yes" size="4" pos="46" show="10.10.10.1" val
  <field name="ip.host" showname="Source or Destination Host: 10.10.10.1" hide="yes" size="4" pos="46" show="10.1
  <field name="ip.dst" showname="Destination: 10.10.10.2" size="4" pos="50" show="10.10.10.2" value="0a0a0a02"/>
  <field name="ip.addr" showname="Source or Destination Address: 10.10.10.2" hide="yes" size="4" pos="50" show="1
  <field name="ip.dst_host" showname="Destination Host: 10.10.10.2" hide="yes" size="4" pos="50" show="10.10.10.2
  <field name="ip.host" showname="Source or Destination Host: 10.10.10.2" hide="yes" size="4" pos="50" show="10.1
  <field name="" show="Source GeoIP: Unknown" size="4" pos="46" value="0a0a0a01"/>
  <field name="" show="Destination GeoIP: Unknown" size="4" pos="50" value="0a0a0a02"/>
</proto>
<proto name="tcp" showname="Transmission Control Protocol, Src Port: 47890, Dst Port: 6669, Seq: 1, Ack: 1, Len:
  <field name="tcp.srcport" showname="Source Port: 47890" size="2" pos="54" show="47890" value="bb12"/>
  <field name="tcp.dstport" showname="Destination Port: 6669" size="2" pos="56" show="6669" value="1a0d"/>
  <field name="tcp.port" showname="Source or Destination Port: 47890" hide="yes" size="2" pos="54" show="47890" v
  <field name="tcp.port" showname="Source or Destination Port: 6669" hide="yes" size="2" pos="56" show="6669" val
  <field name="tcp.stream" showname="Stream index: 0" size="0" pos="54" show="0"/>
  <field name="tcp.len" showname="TCP Segment Len: 11" size="1" pos="66" show="11" value="80"/>
  <field name="tcp.seq" showname="Sequence number: 1    (relative sequence number)" size="4" pos="58" show="1" va
```

Obrázek 5.4: Znázornění, jak jsou v PDML souboru uloženy informace, které potřebujeme k určení síťového toku.

5.2 Analyzátor

Tato část aplikace má na starost hledání zapouzdření. Na vstup této části aplikace přijde PCAP soubor, ve kterém se bude hledat zapouzdření. Dojde také k určení, jaké zapouzdření se v daném souboru vyskytuje, jak bylo popsáno v kapitole 4 při jednotlivých protokolech.

Na základě typu zapouzdření se poté vybere správný modul pro zpracování. Pokud v daném souboru nebude žádné zapouzdření nalezeno, nebude se s tímto souborem dále pracovat. Základní princip této části aplikace je znázorněn na obrázku 5.5. Pro analýzu vstupního souboru bude využit externí nástroj.



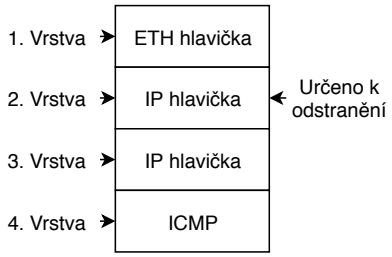
Obrázek 5.5: V této části aplikace se bude v souboru hledat zapouzdření. Při nalezení zapouzdření se bude soubor dále zpracovávat.

Hledání zapouzdření bude prováděno následujícím způsobem:

- **IPIP** – za IP hlavičkou se nachází další IP hlavička;
- **GRE** – za IP hlavičkou následuje GRE hlavička, která je ve verzi 0 a za kterou opět následuje IP hlavička;
- **GRETAP** – za IP hlavičkou následuje GRE hlavička, která je ve verzi 0 a za kterou následuje ethernetová hlavička;
- **PPTP** – za IP hlavičkou následuje GRE hlavička, která je ve verzi 1;
- **L2TP** – za UDP nebo IP hlavičkou se nachází L2TP hlavička.

5.3 Zpracování určitého protokolu

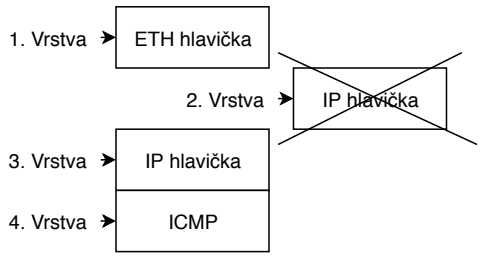
Pokud se v PCAP souboru nalezlo zapouzdření, dojde ke zpracování tohoto souboru. V této části aplikace se pouze označí hlavičky, které mají být z daného souboru odstraněny. Hlavičky jsou označovány za pomocí čísel. Každé hlavičce bude přiděleno číslo, v závislosti na pořadí, ve kterém se v daném paketu nachází. Označen tedy bude seznam hlaviček, které jsou určeny k odstranění. Tento seznam se bude skládat z minimálně jedné odstraňované hlavičky, v závislosti na tom, jaký tunel byl použit. Seznam hlaviček, určených k odstranění, a soubor se dále pošlou do funkce, která má ořezávání souboru na starost. Na obrázku 5.6 je znázorněn paket a jeho jednotlivé vrstvy. Znázorněn je paket, který obsahuje IPIP zapouzdření při ICMP zprávě.



Obrázek 5.6: Znázornění jednotlivých vrstev paketu spolu s označením hlavičky, určené k odstranění.

5.4 Ořezávání

V této části aplikace se provede ořezání o označené hlavičky. Vstupem této části je seznam hlaviček, určených k odstranění a soubor, ze kterého mají být tyto hlavičky odstraněny. Po odstranění všech označených hlaviček se upravený soubor pošle do funkce, která z daného souboru odstraní chyby, které mohli vzniknout při odstranění některých hlaviček. Na obrázku 5.7 je znázorněno, jakým způsobem bude z paketu odstraněno zapouzdření. Obrázek znázorňuje paket, který obsahuje IPIP zapouzdření.



Obrázek 5.7: Zobrazení způsobu odstranění jedné vrstvy zapouzdření z paketu při použití tunelu IPIP.

5.5 Opravování

Při odstranění hlaviček z paketu může dojít k chybám v obsahu u zbývajících hlaviček. Tato část aplikace má za úkol tyto chyby nalézt a opravit. Typickými chybami, ke kterým může při odstranění hlaviček dojít, jsou například informace o následující hlavičce a kontrolní součet. Po opravení těchto chyb se upravený soubor uloží jako nový a bude znovu poslán k analýze, jestli neobsahuje další zapouzdření.

5.6 Uložení výsledků

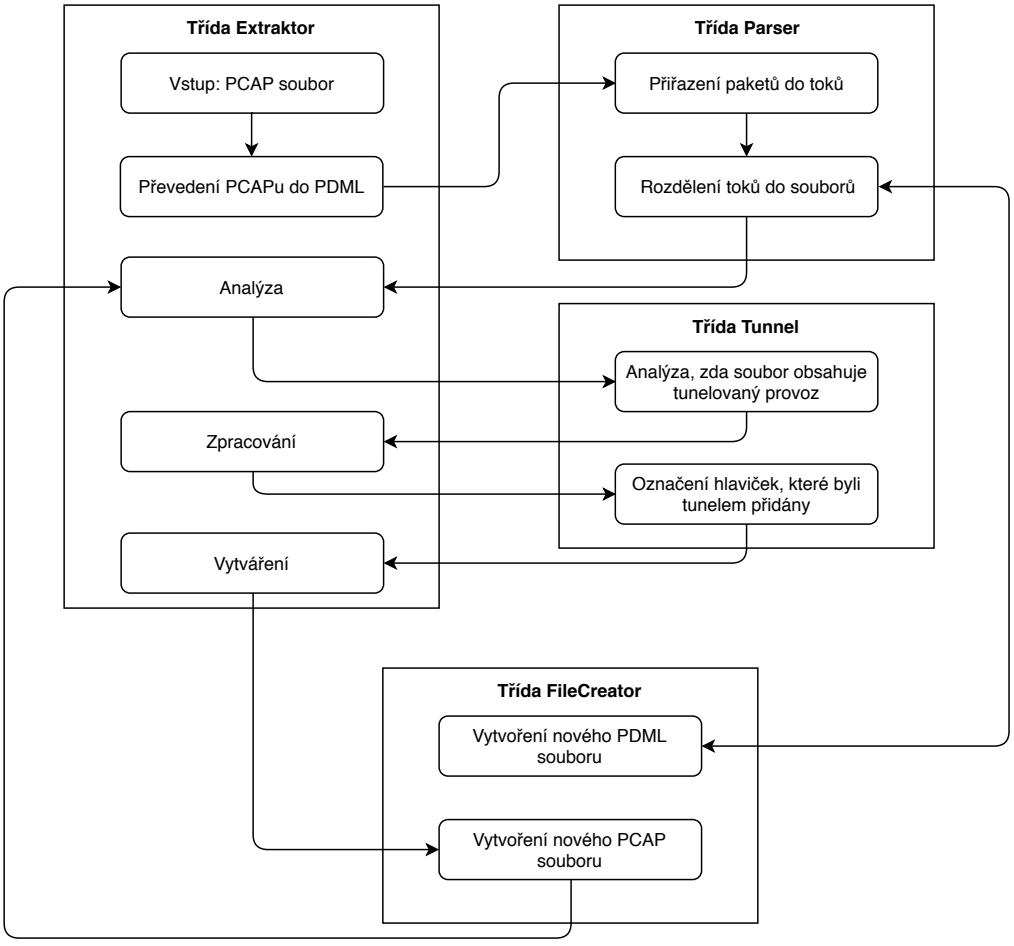
Po provedení všech dílčích kroků je potřeba uložit výsledek procesu do nového PCAP souboru. Nové PCAP soubory vznikají z každého samostatného toku a po odstranění každého zapouzdření. Počet vytvořených souborů tedy záleží na počtu síťových toků ve vstupním PCAP souboru a na počtu jednotlivých zapouzdření v každém síťovém toku. Názvy souboru se postupně pojmenovávají, podle toho jak jsou v programu vytvářeny. Pro nově vytvořené soubory se nejdříve za původní název souboru dodá číslo toku, podle pořadí, v jakém se

daný síťový tok ve vstupní souboru nachází. Před toto číslo se ještě dopíše do názvu *_flow*. Do názvu souboru se tedy přidá *_flowX*, kde X je číslo síťového toku. Po odstranění zapouzdření z daného síťového toku se do názvu ještě dodá informace o tom, kolik zapouzdření bylo z tohoto toku odebráno a před toto číslo se do názvu dopíše *_encap*. Do názvu se celkově tedy dopíše *_encapX*, kde X je počet odstraněných zapouzdření. Příklad nově vytvořeného souboru po odstranění jednoho tunelu: *gre_flow1_encap1.pcap*.

Kapitola 6

Implementace

Tato kapitola je zaměřená na popis implementace aplikace, která je cílem této bakalářské práce. Aplikace je tvořena několika třídami, které jsou jednotlivě popsány v sekci 6.2. Každá třída má na starost určitou část aplikace, podle návrhu z kapitoly 5. Dále je tu popsaná knihovna *ElementTree*, která byla při vytváření aplikace hodně využívána. Tato knihovna je popsána v sekci 6.1. Pro implementaci byl využit programovací jazyk Python 3. Základní funkčnost aplikace je znázorněna na obrázku 6.1.



Obrázek 6.1: Diagram, znázorňující funkčnost aplikace

6.1 Knihovna ElementTree

Tato knihovna slouží pro práci s XML soubory v jazyce Python. Knihovna obsahuje několik funkcí, které umožňují snadnou práci se soubory ve formátu XML, ať už se jedná o procházení těchto souborů nebo jejich vytváření. Já jsem tuto knihovnu využíval pro práci se soubory ve formátu PDML, do kterého jsem vstupní soubor převáděl. V aplikaci bylo nutné tyto soubory analyzovat a vytvářet.

Tato knihovna je v této práci hodně využívaná, kvůli práci s PDML soubory. PDML soubory jsou přepis PCAP souboru do formátu XML, který tato knihovna čte a zpracovává. Většina práce prováděné aplikací je právě čtení informací z těchto souborů. Aby byla práce s těmito soubory co nejsnadnější, bylo potřeba nalézt vhodné nástroje, který k tomuto účelu slouží. A právě tato knihovna práci s těmito soubory velice usnadňuje.

6.2 Třídy

V této části jsou popsány třídy, které byly vytvořeny pro navrženou aplikaci. Extrakci tunelovaných dat mají na starost celkem čtyři třídy. Hlavní a vstupní třídou aplikace je třída *Extraktor*. Třídou, která má na starost rozdělení vstupního souboru do jednotlivých toků, je třída *Parser*. Další třídou je třída *Tunnel*, která má na starost analyzovaní, zda se v daném souboru nachází tunelovaný provoz a stará se o jeho určení. Poslední popsanou třídou je třída *FileCreator*, která slouží k vytváření nových souborů typu PCAP a PDML.

6.2.1 Třída Extraktor

Tato třída je hlavní třídou aplikace. Třída má na starost volání ostatních částí aplikace a obsahuje metody pro předzpracování a analýzu vstupního PCAP souboru.

Třída se také stará o převedení PCAP souboru do souboru typu PDML, se kterým se bude dále pracovat. PDML soubor je vlastně XML soubor, který popisuje PCAP soubor. Díky tomuto převedení se bude se vstupním souborem lépe pracovat. K tomuto převedení se využívá aplikace TShark, která je popsána v sekci 3.2.2.

6.2.2 Třída Parser

Tato třída má na starost rozdělení vstupního souboru na menší soubory, obsahující pouze jeden síťový tok. Tato třída tedy implementuje funkci předzpracování. Třída nejdříve přiřadí všechny pakety ze souboru k toku, ke kterému tento paket patří. Pro každý síťový tok je tedy vytvořen seznam paketů, které k tomuto toku patří.

Tato třída již pracuje se vstupním souborem převedeným do formátu PDML. V něm se z každého paketu zjistí potřebné informace pro určení síťového toku. Pro tento účel je potřeba v souboru nalézt informace o zdrojové a cílové IP adrese, protokolu následující za IP hlavičkou a čísla zdrojového a cílového portu.

6.2.3 Třída Tunnel

Tato třída má na starost zjištění, zda se v daném síťovém toku nachází tunelovaný provoz. Dále také určuje druh tunelu a označuje hlavičky, které byli do daného síťového toku přidány při použití tohoto tunelu. Takto označené hlavičky jsou určeny k následnému odstranění ze souboru. Označené hlavičky se ze souboru odebíraly až při vytváření nového souboru, který již nemá obsahovat toto zapouzdření.

Zjištění a určení tunelu je prováděno pomocí informace o následující hlavičce, která je uložená v IP hlavičce. Pokud nám následující hlavička značí, že bylo použito tunelování, tuto skutečnost třída potvrdí a označí hlavičky, které je potřeba z paketu odstranit. Na obrázku 6.2 je znázorněno, jak je v PDML souboru uložena IPv6 hlavička a kde se nachází informace o následující hlavičce.

```
<proto name="ipv6" showname="Internet Protocol Version 6, Src: 2001:db8:0:f101::1, Dst: 2001:db8:0:f101::2" size="
  <field name="ipv6.version" showname="0110 .... = Version: 6" size="1" pos="14" show="6" value="6" unmaskedvalue=
  <field name="ip.version" showname="0110 .... = Version: 6 [This field makes the filter match on &quot;ip.version
  <field name="ipv6.tclass" showname=".... 0000 0000 .... .... .... .... .... = Traffic Class: 0x00 (DSCP: CS0, EC
    <field name="ipv6.tclass.dscp" showname=".... 0000 00.. .... .... .... .... .... = Differentiated Services Cod
    <field name="ipv6.tclass.ecn" showname=".... .... ..00 .... .... .... .... .... = Explicit Congestion Notifica
  </field>
  <field name="ipv6.flow" showname=".... .... .... 1010 1101 1000 0010 0000 = Flow Label: 0xad820" size="4" pos="1
  <field name="ipv6.plen" showname="Payload Length: 104" size="2" pos="18" show="104" value="0068"/>
  <field name="ipv6.nxt" showname="Next Header: IPv6 (41)" size="1" pos="20" show="41" value="29"/>
  <field name="ipv6.hlim" showname="Hop Limit: 64" size="1" pos="21" show="64" value="40"/>
  <field name="ipv6.src" showname="Source: 2001:db8:0:f101::1" size="16" pos="22" show="2001:db8:0:f101::1" value=
  <field name="ipv6.addr" showname="Source or Destination Address: 2001:db8:0:f101::1" hide="yes" size="16" pos="2
  <field name="ipv6.src_host" showname="Source Host: 2001:db8:0:f101::1" hide="yes" size="16" pos="22" show="2001:
  <field name="ipv6.host" showname="Source or Destination Host: 2001:db8:0:f101::1" hide="yes" size="16" pos="22"
  <field name="ipv6.dst" showname="Destination: 2001:db8:0:f101::2" size="16" pos="38" show="2001:db8:0:f101::2" v
  <field name="ipv6.addr" showname="Source or Destination Address: 2001:db8:0:f101::2" hide="yes" size="16" pos="3
  <field name="ipv6.dst_host" showname="Destination Host: 2001:db8:0:f101::2" hide="yes" size="16" pos="38" show="
  <field name="ipv6.host" showname="Source or Destination Host: 2001:db8:0:f101::2" hide="yes" size="16" pos="38"
  <field name="" show="Source GeoIP: Unknown" size="16" pos="22" value="20010db80000f101000000000000001"/>
  <field name="" show="Destination GeoIP: Unknown" size="16" pos="38" value="20010db80000f101000000000000002"/>
</proto>
```

Obrázek 6.2: Obrázek, znázorňující IPv6 hlavičku v souboru PDML se zvýrazněnou informací o následující hlavičce, kterou třída tunnel potřebuje vědět k určení zapouzdření.

6.2.4 Třída FileCreator

Tato třída má na starost vytváření nových souborů, které budou vznikat ze vstupního souboru. Třída obsahuje metody pro vytváření souborů typu PCAP a PDML. Vytváření souborů typu PDML je použito při předzpracování, kdy je potřeba vstupní soubor rozdělit na několik menších souborů, obsahující pouze jeden síťový tok. K tomuto účelu byla využita knihovna *ElementTree*, která je popsána v samostatné sekci 6.1. Při rozdělování souboru se označí ty pakety, které patří k danému síťovému toku, a poté se z těchto paketů vytvoří samostatný PDML soubor, obsahující pouze jeden síťový tok.

K vytváření nových PCAP souborů dochází po předzpracování a po odstranění každého zapouzdření, které daný síťový tok obsahuje. Z každého síťového toku se tedy vytvoří nový PCAP soubor, ze kterého se poté vytvoří více PCAP souborů, v závislosti na tom, kolik zapouzdření daný síťový tok obsahuje. Pokud síťový tok žádné zapouzdření neobsahuje, žádné další soubory se z daného síťového vytvářet nebudou. PCAP soubory se vytváří z informací, uložených v PDML souboru.

V této třídě také dochází k odstraňování označených hlaviček a k opravování zbývajících hlaviček, pokud v nich došlo k nějaké chybě. Označené hlavičky se při vytváření nového souboru přeskočí a do nového souboru se tak nezapíší. K přeskakování namísto fyzického odstranění hlavičky ze souboru dochází kvůli přílišné výkonnostní náročnosti fyzického odstraňování hlaviček ze souboru. Stejně tak dochází v této fázi aplikace k opravování hlaviček, pokud je to zapotřebí.

6.3 Spouštění aplikace

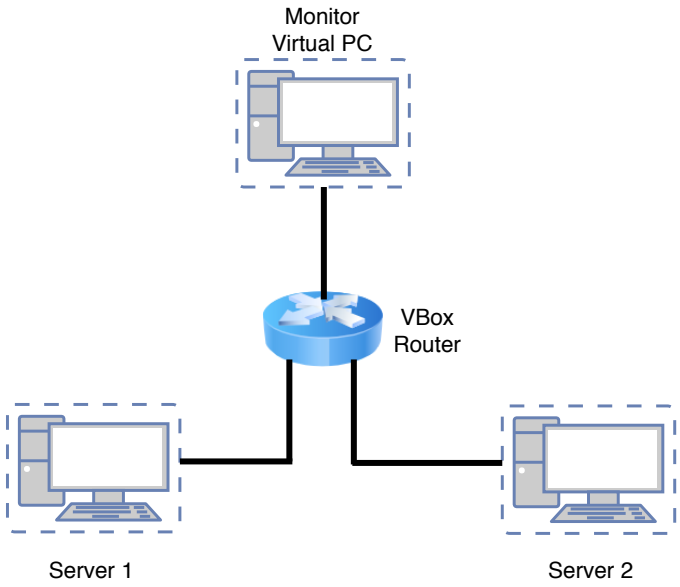
Aplikace se spouští z příkazové řádky a ke spuštění je zapotřebí mít na počítači nainstalované aplikace TShark a Python 3. Aplikace vyžaduje ke spuštění zadání dvou argumentů příkazové řádky a to cestu k vstupnímu souboru a cestu ke složce, do které se budou ukládat výstupní soubory. Oba tyto argumenty jsou povinné. Dále aplikace nabízí možnost vytvářet nové soubory až po odstranění všech zapouzdření. Tato možnost se aktivuje pomocí volitelného argumentu příkazové řádky. Argumenty jsou tedy následující:

- `-h, --help` – Vypíše nápovědu
- `-f, --file [SOUBOR]` – Cesta ke vstupnímu souboru
- `-d, --dir [SLOŽKA]` – Cesta k výstupnímu adresáři
- `-a, --all` – Vytvoření nového souboru až po odstranění všech zapouzdření

Kapitola 7

Testování

Pro získávání tunelovaných dat bylo nutné vytvořit vhodné testovací prostřední pro jejich získávání. Rozhodl jsem, že toto prostředí bude virtualizované. Toto rozhodnutí bylo způsobeno nedostatečným počtem fyzických počítačů, na kterých bych mohl testovací data vytvářet. Na vytváření virtuálních počítačů jsem použil aplikaci Oracle VM VirtualBox. Celkem jsem pro vytváření dat využil tři virtuální stroje. Dva stroje byly využity k vytváření tunelů a jeden stroj byl využit k monitorování sítě. Monitorování sítě bylo prováděno pomocí aplikace Wireshark a zachycená síťová komunikace zde také byla ukládaná do PCAP souborů. Testovací topologie je znázorněna na obrázku 7.1. Všechny tři virtuální stroje využívali operační systém Linux, konkrétně linuxuvou distribuci Ubuntu. V části 7.1 je popsán balíček nástrojů, který jsem pro vytváření testovacích dat využíval. V části 7.2 je popsáno prostředí, ve kterém byly testovací data vytvářena. Dále je v části 7.3 popsáno, jakým způsobem jsem testoval aplikaci a v sekci 7.4 jsou zobrazeny výkonnostní testy aplikace.



Obrázek 7.1: Testovací topologie

Virtuální stroje *Server 1* a *Server 2* byly použity k vytváření tunelů. Tunely jsem vytvářel právě mezi těmito dvěma stroji. Třetí stroj, nazvaný *Monitor Virtual PC*, jsem

používal k zachytávání paketů na síti. Zachytávání paketů jsem prováděl pomocí aplikace Wireshark a síťové karty, která byla přepnutá do promiskuitního režimu.

7.1 Iproute2

Iproute2 je balíček nástrojů pro ovládání a monitorování různých aspektů sítí v Linuxovém jádře, jako jsou například směrování, síťové rozhraní a tunely. Tento balíček obsahuje několik nástrojů, mezi nejvýznamnější nástroje patří *ip* a *tc*. Nástroj *ip* slouží ke konfiguraci IPv4 a IPv6 sítí, zatímco nástroj *tc* slouží k ovládání síťového provozu. Tento balíček byl využit pro vytváření tunelů.

Cílem balíčku *iproute2* je nahradit starší balík nástrojů *net-tools*, které již není nadále vyvíjen. Všechny nástroje obsahují detailní informace a dokumentaci. Jak byly nástroje z balíčku *net-tools* nahrazeny nástroji z balíčku *iproute2* je ukázáno v tabulce 7.1.

Účel	**net-tools**	**iproute2**
Konfigurace dat a spojení	ifconfig	ip addr, ip link
Směrovací tabulky	route	ip route
Sousedé	arp	ip neight
VLAN	vconfig	ip link
Tunely	iptunnel	ip tunnel
Multicast	ipmaddr	ip maddr
Statistiky	nestat	ss

Tabulka 7.1: Nahrazení nástrojů *net-tools* nástroji *iproute2*

Jak lze vidět z tabulky, pro vytváření tunelů jsem tedy využíval nástroj *ip tunnel*. Tunely jsem vytvářel pomocí následujícího příkazu:

```
$ ip tunnel add NAZEV mode MOD remote ADRESA1 local ADRESA2
```

, kde:

NAZEV je název síťového rozhraní, reprezentující tunel;

MOD je typ tunelu, který chceme vytvořit (ipip, gre, sit, ipip6, ip6gre);

ADRESA1 je IP adresa vzdáleného síťového zařízení;

ADRESA2 je IP adresa lokálního síťového zařízení.

Kompletní popis tohoto příkazu lze nalézt na jeho manuálových stránkách [1].

7.2 Prostředí

Všechny stroje, které byly použity k vytváření tunelů, používaly operační systém Ubuntu ve verzi *Server*. Tato verze Ubuntu nenabízí žádné grafické uživatelské prostředí, pouze příkazový řádek. Verze obsahuje řadu užitečných nástrojů, určených pro administrátory sítí. Výběr této verze byl učiněn z důvodů omezeného výkonu na mém fyzickém počítači a z důvodu, že tyto virtuální stroje grafické uživatelské rozhraní vůbec nepotřebovaly. Poslední virtuální stroj, určený k zachytávání paketů, už používal klasickou verzi Ubuntu s uživatelským rozhraním. Na tomto stroji již bylo grafické uživatelské rozhraní nutné, protože na něm běžela aplikace Wireshark.

7.3 Funkční testování

Funkčnost aplikace jsem testoval na mnou vytvořených PCAP souborech. Bylo potřeba otestovat všechny podporované tunelovací protokoly a jejich různé kombinace. Každý tunelovací protokol jsem testoval na několika souborech. Celkově jsem funkčnost testoval zhruba na 40 souborech.

Bylo také nutné otestovat nějaké extrémní případy, kdy síťový tok obsahoval velký počet různých zapouzdření. Pro tento účel byl vytvořen soubor, obsahující pět různých tunelů. Cílem tohoto testu bylo zaručení, že si aplikace poradí i s takovými případy.

Pro zjišťování, zda vše funguje podle očekávání, jsem nově vytvořené soubory otevíral v aplikaci Wireshark a zjišťoval, zda se soubor v pořádku otevře a jestli jsou všechna data správně uložena. Jestliže při tomto testování došlo k nějaké chybě, nastalo zjišťování, kde k chybě došlo. K tomu jsem používal čtení části souboru v binární podobě, pomocí kterého jsem zjišťoval, kde přesně se v souboru nachází chyba. Pro tento účel jsem vytvářel pouze části souboru, kdy jsem soubor postupně vytvářel hlavičku po hlavičce. Díky tomu se soubor v binární podobě lépe četl a snáze se tak došlo k odhalení a opravení chyby.

7.4 Výkonnostní testování

V této části jsou zobrazeny výkonnostní testy aplikace. Toto testování je zaměřeno na časovou a paměťovou náročnost aplikace při různě velkých souborech, obsahujících různý počet síťových toků. Testování probíhalo na virtuálním stroji a za použití aplikace TShark ve verzi 2.5.0. Pro testování byl využit počítač s procesorem AMD FX 6300, který má šest jader pracující na frekvenci 3500 MHz a 8096 MB RAM. Pro virtuální stroj byly k dispozici tři jádra procesoru a vyhrazeno bylo pro něj 2048 MB RAM. Výsledky tohoto testování jsou zobrazeny v tabulce 7.2.

Počet toků	**Počet paketů**	**T1 – Čas převodu [s]**	**T2 – Čas běhu [s]**	**T3 – Čas celkem [s]**	**RAM [MB]**
1	10	0.989	0.077	1.066	3.838
1	50	0.999	0.279	1.279	3.838
1	100	0.801	0.538	1.339	27.787
2	20	0.901	0.145	1.045	16.314
2	200	1.175	0.985	2.160	35.004
7	317	1.212	1.631	2.843	54.387
7	1034	1.073	5.536	6.609	129.511

Tabulka 7.2: Tabulka s výsledky výkonnostního testování

Celkem se měřily tři časy za běhu aplikace. První čas T1 je čas potřebný pro převod PCAP souboru do formátu PDML. Druhý čas T2 je čas běhu aplikace bez času potřebného pro převod souboru. Třetí čas T3 je čas celého běhu aplikace, tedy součet časů T1 a T2. Poslední sledovanou položkou bylo využití paměti RAM při běhu aplikace. Časy jsou měřeny v sekundách, paměť byla měřena v MB.

Z tabulky je patrné, že na čas nutný pro převod vstupního souboru nemá počet paketů ani toků žádný vliv. Aplikaci TShark převod souboru trval vždy přibližně stejnou dobu, která trvala kolem jedné vteřiny. Největší vliv na běh aplikace má počet paketů ve vstupním souboru. Čím větší je počet paketů, tím delší bude běh aplikace a využití systémové paměti.

Kapitola 8

Závěr

Hlavním cílem této bakalářské práce bylo navrhnout a implementovat aplikaci pro extrakci tunelovaných dat do samostatných toků. Aplikace rozděluje zachycenou síťovou komunikaci do samostatných toků a odstraňuje z nich všechny tunelové zapouzdření, které jednotlivé síťové toky obsahují. Zapouzdření je možno odstraňovat ze síťového toku postupně nebo zároveň odstranit všechny zapouzdření, které obsahuje. Aplikace má usnadnit následnou analýzu a diagnostiku této zachycené komunikace. Při odstraňování zapouzdření bylo potřeba také zajistit, aby zbývající hlavičky v souboru obsahovali správné informace, které se mohly při odstraňování stát neplatné.

V rámci bakalářské práce byla nastudovaná síťová architektura TCP/IP a její základní protokoly. Dále bylo zapotřebí se seznámit s možnostmi zachytávání dat na síti a s principem tunelování. V neposlední řade bylo potřeba nastudovat tunelovací protokoly a seznámit se s možnostmi extrakce zapouzdřených dat. Pro funkční testování výsledné aplikace bylo zapotřebí vytvořit několik testovacích souborů, obsahující různé tunelovací protokoly a jejich kombinace.

Aplikace byla implementována v programovacím jazyce Python. Pro testování aplikace byly použité vlastní vytvořené testovací PCAP soubory. Aplikace podporuje tunelovací protokoly GRE, GRETAP, IPIP a L2TPv3. Pro práci se soubory PCAP byl využit jejich převod do formátu PDML za pomocí nástroje TShark, který je nutný pro běh aplikace. V práci jsou také obsaženy výsledky výkonnostního testování aplikace, které bylo provedeno při testování implementace. Při výkonnostním testováním vyšlo najevo, že aplikace je náročná na operační paměť. V budoucnu by tedy bylo vhodné se zaměřit na optimalizaci aplikace.

Literatura

[1] *WWW stránky - manuálové stránky ip-tunnel*. [Online; navštíveno 20.01.2018]. URL http://man7.org/linux/man-pages/man8/ip-tunnel.8.html

[2] Deering, S. E.; Hinden, R. M.: Internet Protocol, Version 6 (IPv6) Specification. RFC 2460, RFC Editor, Prosinec 1998, http://www.rfc-editor.org/rfc/rfc2460.txt.

[3] Hamzeh, K.; Pall, G.; Verthein, W.; aj.: Point-to-Point Tunneling Protocol (PPTP). RFC 2637, RFC Editor, Červenec 1999, http://www.rfc-editor.org/rfc/rfc2637.txt.

[4] Hanks, S.; Li, T.; Farinacci, D.; aj.: Generic Routing Encapsulation (GRE). RFC 1701, RFC Editor, Říjen 1994, http://www.rfc-editor.org/rfc/rfc1701.txt.

[5] Kabelová, A.; Dostálek, L.: *Velký průvodce protokoly TCP/IP a systémem DNS*. Computer Press, Albatros Media a.s., 2012, ISBN 978-80-251-2236-5.

[6] Lau, J.; Townsley, M.; Goyret, I.: Layer Two Tunneling Protocol - Version 3 (L2TPv3). RFC 3931, RFC Editor, Březen 2005, http://www.rfc-editor.org/rfc/rfc3931.txt.

[7] Matoušek, P.: *Síťové aplikace a jejich architektura*. VUTIUM, 2014, ISBN 978-80-214-3766-1.

[8] Postel, J.: User Datagram Protocol. STD 6, RFC Editor, Srpen 1980, http://www.rfc-editor.org/rfc/rfc768.txt.

[9] Postel, J.: Internet Protocol. STD 5, RFC Editor, Září 1981, http://www.rfc-editor.org/rfc/rfc791.txt.

[10] Postel, J.: Transmission Control Protocol. STD 7, RFC Editor, Září 1981, http://www.rfc-editor.org/rfc/rfc793.txt.

[11] Sanders, C.: *Practical Packet Analysis, 2nd Edition*. No Starch Press, 2011, ISBN 978-1-59327-266-1.

[12] Townsley, W.; Valencia, A.; Rubens, A.; aj.: Layer Two Tunneling Protocol "L2TP". RFC 2661, RFC Editor, Srpen 1999, http://www.rfc-editor.org/rfc/rfc2661.txt.

Příloha A

Obsah přiloženého paměťového média

Přiložené CD obsahuje následující položky:

- src/ – Zdrojové kódy výsledné aplikace
- src-tex/ – Zdrojové kódy této dokumentace v LaTeXu
- pcaps/ – Vytvořené testovací soubory ve formátu PCAP
- xnahal01.pdf – Tato dokumentace ve formátu PDF
- README – Popis aplikace