VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

AUTOMATIZOVANÁ DETEKCE TRANSPORTNÍHO PROTOKOLU V ZACHYCENÉ SÍŤOVÉ KOMUNIKACI

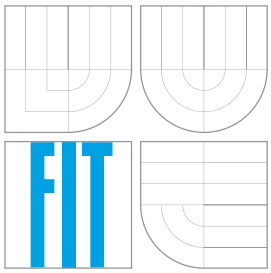
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

ZBYNĚK LAZÁREK

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

AUTOMATIZOVANÁ DETEKCE TRANSPORTNÍHO PROTOKOLU V ZACHYCENÉ SÍŤOVÉ KOMUNIKACI

AUTOMATIC TRANSPORT PROTOCOL DETECTION IN CAPTURED COMMUNICATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — ZBYNĚK LAZÁREK
AUTHOR

VEDOUCÍ PRÁCE — Ing. JAN PLUSKAL
SUPERVISOR

BRNO 2015

Abstrakt

Cílem této bakalářské práce je vytvořit knihovnu v jazyce C#, která bude schopna automatizovaně detekovat transportní protokoly v zachycené komunikaci. Tato knihovna dokáže rozpoznat transportní protokol v tunelovém provozu bez znalosti tunelovacího protokolu. Práce popisuje jedinečné signatury TCP/IP protokolů, na kterých je automatizovaná detekce založená. Dále je popsán způsob, jakým detekce pracuje. V závěru práce je automatizovaná detekce podrobena testům, které mají za úkol prověřit její výkonnost a účinnost.

Abstract

The aim of this bachelor thesis is to create a library in C#, which will be able to automatically detect transport protocols in a captured network traffic. This library can recognize a transport protocol in tunnel traffic without the knowledge of tunneling protocol. The thesis describes the unique signatures of TCP/IP protocols on which the automatic detection is based. Furthermore, it is described how the detection works. The conclusion is subjected to automatic detection tests, which are designed to review its performance and efficiency.

Klíčová slova

automatizovaná detekce, transportní protokol, síť, komunikace, IPv4, IPv6, TCP, UDP

Keywords

automatic detection, transport protocol, network, communication, IPv4, IPv6, TCP, UDP

Citace

Zbyněk Lazárek: Automatizovaná detekce transportního
protokolu v zachycené síťové komunikaci, bakalářská práce, Brno, FIT VUT v Brně, 2015

Automatizovaná detekce transportního protokolu v zachycené síťové komunikaci

Prohlášení

Prohlašuji, že jsem tuto semestrální práci vypracoval samostatně pod vedením Ing. Jana Pluskala

.....
Zbyněk Lazárek
19. května 2015

Poděkování

Zde bych rád poděkoval Ing. Janu Pluskalovi za ochotu a přátelský přístup při vedení mé práce.

Obsah

1 Úvod **3**

2 Analýza síťové komunikace **4**
2.1 ISO/OSI model . . . 4
2.2 TCP/IP model . . . 5
2.3 Síťová vrstva . . . 6
2.3.1 IPv4 . . . 6
2.3.2 IPv6 . . . 7
2.4 Transportní vrstva . . . 8
2.4.1 TCP . . . 9
2.4.2 UDP . . . 10
2.5 Tunelovací protokoly . . . 11
2.5.1 GRE . . . 11
2.5.2 6in4 . . . 12
2.5.3 Teredo . . . 13

3 Klasické rozpoznání protokolů **14**
3.1 Příklady klasického rozpoznání . . . 14
3.1.1 Rozpoznání TCP/IP . . . 14
3.1.2 Rozpoznání v tunelovacím protokolu GRE . . . 15
3.2 Problémy klasického rozpoznání . . . 15

4 Signatury protokolů **17**
4.1 Signatury IP . . . 17
4.1.1 Signatury IPv4 . . . 17
4.1.2 Signatury IPv6 . . . 18
4.2 Signatury TCP a UDP . . . 19
4.2.1 Signatury TCP . . . 19
4.2.2 Signatury UDP . . . 19

5 Postup automatizované detekce **21**
5.1 Obecný postup . . . 21
5.2 Detekce v IPv4 . . . 21
5.3 Detekce v IPv6 . . . 22
5.4 Příklad detekce . . . 23

6 Testování **25**
6.1 Výkonnost automatizované detekce 25
6.1.1 Doba zpracování přeneseného souboru 25
6.1.2 Doba zpracování jednoho rámce 27
6.2 Úspěšnost automatizované detekce 29

7 Závěr **31**

Kapitola 1

Úvod

Detekce transportního protokolu v neznámém tunelovacím protokolu je složitá činnost, která vyžaduje podrobnou znalost běžně využívaných protokolů. Pokud není známo umístění transportního protokolu v zachycených datech, je nutné vyhledávat na základě vlastností, kterými se od sebe protokoly liší. Tyto vlastnosti mohou být například pořadí pevně daných hodnot v hlavičkách protokolů nebo sekvence hodnot, které se v jiných protokolech nevyskytují.

Cílem této bakalářské práce je implementovat knihovnu pro automatizovanou detekci transportního protokolu v zachycené komunikaci. V kapitole nazvané Analýza síťové komunikace jsou probrány vrstvové modely ISO/OSI a z něj vycházející model TCP/IP. Po prozkoumání vrstvových modelů jsou popsány protokoly, jejichž znalost je nezbytná k vytvoření automatizované detekce. Jmenovitě se jedná o protokoly IPv4, IPv6, TCP a UDP. Velký důraz je kladen na strukturu hlaviček jednotlivých protokolů. Výše zmíněné protokoly jsou využívány v tunelovacích protokolech GRE, 6in4 a Teredo. U těchto protokolů je popsán princip jejich činnosti a motivace k jejich provozu.

Následující kapitola přibližuje klasický způsob, jakým jsou síťové protokoly běžně zpracovány. Detailně je popsáno zpracování TCP/IP protokolů a protokolu GRE. Jsou zde uvedeny případy, ve kterých není klasické rozpoznání schopno detekovat transportní protokol a je nutné použít automatizovanou detekci.

Správná funkčnost automatizované detekce je podmíněna objevením jedinečných signatur protokolů IPv4, IPv6, TCP a UDP. Signaturami jsou myšleny vlastnosti, které daný protokol vždy obsahuje. Například pořadí daného protokolu nebo hodnoty polí v jeho hlavičce, které odpovídají daným kritériím.

Po zjištění všech potřebných signatur TCP/IP protokolů je přistoupeno k detailnímu zpracování automatizované detekce. Je popsán obecný postup, kterým detekce pracuje. Rozebrána je posloupnost kroků při zpracování protokolů následující za IPv4 nebo IPv6 protokolem. Pro názornost je uvedeno, jak automatizovaná detekce krok po kroku pracuje.

V závěru práce je automatizovaná detekce transportního protokolu podrobena testům, které mají za úkol odhalit její výkonnost a úspěšnost. Výkonnost automatizované detekce je srovnána s klasickým způsobem zpracování transportních protokolů. Úspěšnost automatizované detekce je zkoumána jak na zachycené běžné komunikaci, tak na uměle vytvořených datech obsahujících mimo jiné i vícenásobné zapouzdření transportního protokolu.

Kapitola 2

Analýza síťové komunikace

Ke vzniku Internetu vedla snaha přenášet informace z jednoho místa na druhé. Aby se přenosu dalo dosáhnout, bylo nutné definovat pravidla, podle kterých se přenos řídí. Tato pravidla jsou obsažena ve vrstvových modelech.

V dnešní době se využívají převážně dva vrstvové modely. Model ISO/OSI (International Organization for Standardization/Open Systems Interconnection) slouží jako referenční model, který popisuje síťovou architekturu. Výrazně jednodušší než model ISO/OSI je model TCP/IP. Dále jsou popsány jednotlivé vrstvy modelu ISO/OSI. Jejich grafické znázornění je na obrázku 2.1.

2.1 ISO/OSI model

Model ISO/OSI se skládá ze sedmi vrstev. Každá vrstva definuje způsob, jak komunikovat se stejnou vrstvou mezi komunikujícími procesy. Daná vrstva používá pro přenos nižší vrstvu, aniž by znala, jak probíhá přenos na této nižší vrstvě.

1. **Fyzická vrstva** (physical layer) definuje elektrické a fyzické vlastnosti datového spoje, vztah mezi zařízením a přenosovým médiem, topologii síťové infrastruktury. Jednotka se kterou pracuje fyzická vrstva se nazývá bit.

2. **Linková vrstva** (data link layer) popisuje virtuální spoj mezi dvěma zařízeními. Kontroluje, zda na fyzické vrstvě došlo k chybě. Určuje, jak se adresují jednotlivá zařízení v síti. Přenášená jednotka se nazývá rámec (frame).

7. Application layer	Aplikační vrstva
6. Presentation layer	Presentační vrstva
5. Session layer	Relační vrstva
4. Transport layer	Transportní vrstva
3. Network layer	Síťová vrstva
2. Datalink layer	Linková vrstva
1. Physical layer	Fyzická vrstva

Obrázek 2.1: ISO/OSI model

3. **Síťová vrstva** (network layer) poskytuje schopnost přenášet různě velké bloky dat mezi zařízeními, která jsou připojena do stejné sítě. Jednotkou pro přenos po síťové vrstvě je datagram.

4. **Transportní vrstva** (transport layer) zajišťuje přenos různě velkých bloků dat mezi zařízeními, které se mohou nacházet v různých sítích.

5. **Relační vrstva** (session layer) vytváří a udržuje logické spojení mezi navzájem komunikujícími aplikacemi.

6. **Presentační vrstva** (presentation layer) poskytuje reprezentaci dat, která není závislá na komunikujících aplikacích.

7. **Aplikační vrstva** (application layer) přímo komunikuje s uživatelskou aplikací.

2.2 TCP/IP model

TCP/IP model je nejrozšířenějším modelem současnosti. Spojuje fyzickou a linkovou vrstvu z OSI modelu do vrstvy fyzického rozhraní. Prezentační a relační vrstvu spojuje do vrstvy aplikační. Pro každou vrstvu je definována sada protokolů.

Dále jsou popsány vrstvy TCP/IP modelu. Graficky jsou znázorněny na obrázku 2.2.

1. **Vrstva fyzického rozhraní** (network interface layer) zapouzdřuje data z vyšších vrstev do rámců, které předává na fyzické médium.

2. **Internetová vrstva** (internet layer) vytváří IP datagramy, které se snaží doručit k cíli. Pro přenos datagramů jsou využity protokoly z rodiny IP (IPv4 [1], IPv6 [2], ICMPv4 [3], ICMPv6 [4]).

3. **Transportní vrstva** (transport layer) slouží k přenosu dat mezi aplikací běžící na zdrojovém zařízení a aplikací běžící na cílovém zařízení. Spojení je jednoznačně určeno číslem zdrojového a cílového portu. Data přenášená po transportní vrstvě jsou zabalena do paketů (packets).

4. **Aplikační vrstva** (application layer) řeší kódování dat, pořadí zasílaných zpráv a reprezentaci dat. Na aplikační vrstvě spolu komunikují přímo procesy.

Application layer	Aplikační vrstva
Transport layer	Transportní vrstva
Internet layer	Internetová vrstva
Network interface layer	Vrstva fyzického rozhraní

Obrázek 2.2: TCP/IP model

2.3 Síťová vrstva

Jak už bylo naznačeno výše, síťová vrstva zajišťuje směrování dat od jednoho zařízení k druhému v rámci různých sítí. Pro směrování mezi koncovými zařízeními jsou v dnešní době využívány protokoly IPv4 a IPv6.

IP datagram obsahuje hlavičku, která je následována daty. Hlavička mimo jiné obsahuje IP adresu zdrojového zařízení a IP adresu cílového zařízení. Pomocí cílové IP adresy se IP datagram dostane skrz Internet od zdroje k cíli.

2.3.1 IPv4

Na obrázku 2.3 je zobrazen IPv4 datagram. Jednotlivé části jsou popsány dále.

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	Version				IHL				DSCP						ECN		Total length															
32	Identification																Flags			Fragment Offset											
64	Time to live								Protocol								Header checksum															
96	Source IP address																															
128	Destination IP address																															
160	Options								Padding																							
	Data																															

Obrázek 2.3: IPv4 datagram

- **Version** (4 bity) je verze IP protokolu. Pro IPv4 je tato hodnota vždy rovna 4.
- **Internet Header Length** (4 bity) je délka hlavičky v počtu 32-bitových slov. Minimální délka hlavičky je 160 bitů, což udává, že nejmenší číslo v IHL je 5.
- **Type of Service** (8 bitů) je typ služby, který je v IP datagramu obsažen. Typy služeb hrají roli v oblasti, která se nazývá Diferenciované služby.
- **Total Length** (32 bitů) je délka IP datagramu včetně hlavičky a dat. Číslo představuje délku v bytech. Snadno se dá odvodit, že minimální délka IP datagramu je 20 bytů (délka hlavičky) a maximální délka je 65535 bytů (maximální rozsah 32-bitového čísla).
- **Identification** (32 bitů) je číslo, které slouží ke správnému uspořádání fragmentovaných datagramů v případě, že nepřicházejí ve správném pořadí.
- **Flags** (3 bity)
  - **bit 0**: Je rezervován. Musí být nastaven na 0.
  - **bit 1**: (Don't fragment) Pokud je nastaven na 1, zpráva nemůže být rozdělena do více datagramů. Pokud je nastaven na 0, posílaná zpráva může být rozdělena do více datagramů.
  - **bit 2**: (More fragments) Pokud je nastaven na 1, následující datagramy obsahují další část zprávy. Pokud je nastaven na 0, jedná se o poslední část zprávy.

Číslo	Protokol
0	HOPOPT - IPv6 Hop-by-Hop Option
1	ICMP
6	TCP
17	UDP
43	IPv6-Route - Routing Header for IPv6
44	IPv6-Frag - Fragment Header for IPv6
59	IPv6-NoNxt - No Next Header for IPv6
60	IPv6-Opts - Destination Options for IPv6

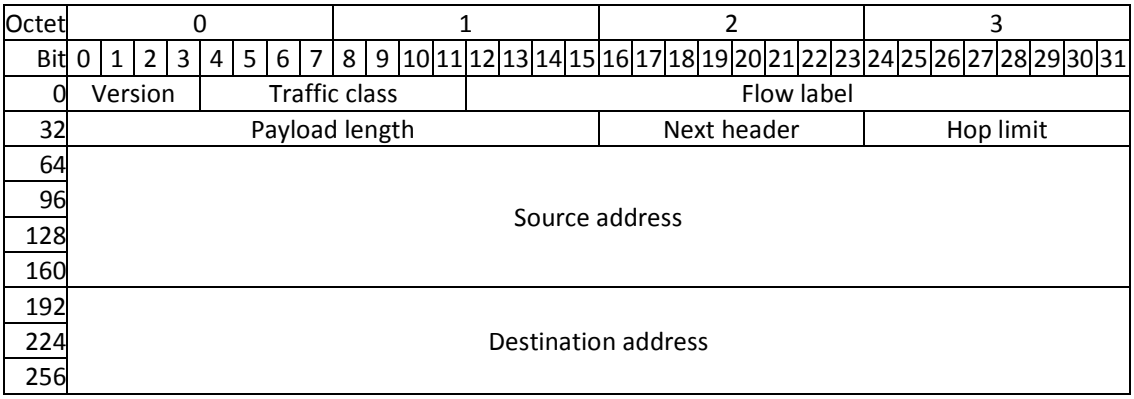
Tabulka 2.1: Čísla přiřazená vybraným protokolům v hlavičkách protokolů IPv4 a IPv6

- **Time to Live** (8 bitů) určuje maximální počet aktivních prvků, přes které může datagram projít. Každý síťový prvek, který zpracuje hlavičku datagramu, musí hodnotu TTL snížit. Pokud se hodnota TTL dostane na nulu, musí se datagram zničit.
- **Protocol** (8 bitů) udává, který protokol je zapouzdřen v IP datagramu. Vybraná čísla přiřazená protokolům jsou zobrazena v tabulce 2.1, která vychází z RFC 790 [5] a z RFC 2460 [2].
- **Header checksum** (16 bitů) udává kontrolní součet hlavičky. Při každé změně TTL se musí kontrolní součet přepočítat. Pokud během přenosu dojde k chybě v rámci IP hlavičky, pomocí kontrolního součtu se chyba odhalí.
- **Source address** (32 bitů) je zdrojová adresa.
- **Destination address** (32 bitů) je cílová adresa.
- **Options** jsou volitelné položky v hlavičce.
- **Padding** je výplň, která zajišťuje zarovnání hlavičky na násobky 32 bitů.

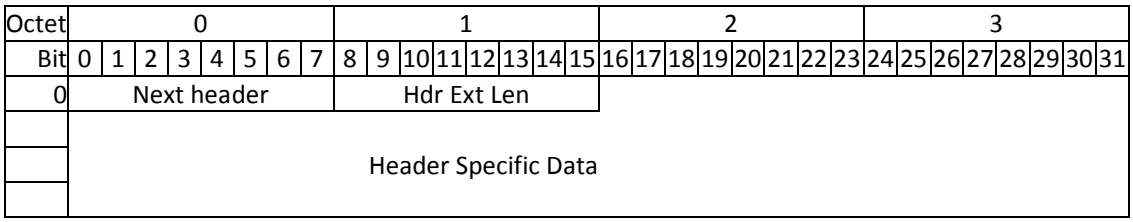
2.3.2 IPv6

Na obrázku 2.4 je zobrazen IPv6 datagram. Jednotlivé části jsou popsány dále.

- **Version** (4 bity) udává verzi IP protokolu. Pro IPv6 je tato hodnota vždy rovna 6.
- **Traffic Class** (8 bitů) má stejný význam jako ToS v IPv4 hlavičce. Využívá se například v oblasti s názvem Diferenciované služby.
- **Flow Label** (20 bitů) slouží k označení posloupnosti datagramů, které mají speciální význam.
- **Payload Length** (16 bitů) udává délku dat za hlavičkou (payload) v bytech. Do Payload Length se počítají i rozšířené hlavičky.
- **Next Header** (8 bitů) udává typ hlavičky, který bezprostředně následuje za IPv6 hlavičkou.



Obrázek 2.4: Struktura IPv6 hlavičky



Obrázek 2.5: Struktura IPv6 rozšiřující hlavičky

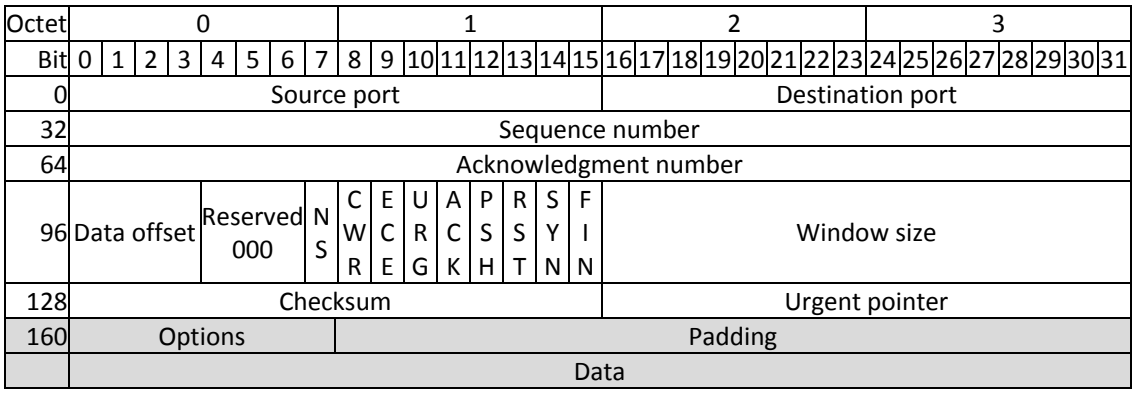
- **Hop Limit** (8 bitů) je číslo, které se dekrementuje o jedničku pokaždé, když je datagram přeposlán síťovým zařízením. Pokud hodnota klesne na 0, datagram je zahozen.
- **Source Address** (128 bitů) udává adresu zdroje.
- **Destination Address** (128 bitů) udává adresu cíle.

Rozšiřující informace o internetové vrstvě nejsou jako v případě IPv4 vkládány do hlavičky, ale slouží k tomu rozšiřující hlavičky [6], které se vkládají za IPv6 hlavičku. Jejich počet není omezen. Na obrázku 2.5 je zobrazen jednotný formát, který by měly rozšiřující hlavičky dodržet. Povinné části rozšiřující hlavičky jsou znázorněny níže.

- **Next header** udává číslo dalšího protokolu. Číslování se řídí tabulkou 2.1.
- **Hdr Ext Len** udává délku rozšiřující hlavičky v 8 bytech, do kterých se nepočítá prvních osm bytů. Například číslo 2 značí délku 24 bytů.
- **Header Specific Data** obsahuje data rozšiřující hlavičky.

2.4 Transportní vrstva

Úkolem transportní vrstvy je vytvořit logické spojení mezi dvěma procesy. Adresování procesů je založeno na číslech portů. Jednoznačné spojení je vytvořeno přidělením jedinečného čísla portu na každém zařízení. Čísla portů jsou rozdělena do tří bloků.



Obrázek 2.6: Struktura TCP paketu

1. **Rezervované porty** (0-1023) jsou přidělovány organizací IANA (Internet Assigned Numbers Authority).
2. **Registrované porty** (1024-49151) jsou často využívány pro firemní protokoly. Seznam registrovaných portů je také poskytován organizací IANA.
3. **Dynamické porty** (49452-65535) jsou většinou využívány pro účely, kdy na čísle portu nezáleží. Například klientský proces čeká na dynamickém portu odpověď od serveru.

Některé služby vyžadují od přenosu dat po síti jiné vlastnosti než ostatní služby. Některé potřebují spolehlivý přenos dat, jiné vyžadují co nejmenší režii při přenosu nebo co nejrychlejší doručení zprávy. Proto jsou v TCP/IP modelu nadefinovány protokoly TCP [7] a UDP [8], které splňují požadavky zmíněných typů služeb.

2.4.1 TCP

TCP (Transmission Control Protocol) je protokol, který zajišťuje spolehlivý přenos dat. To znamená, že data předávaná aplikační vrstvě jsou bezchybná a ve správném pořadí jako byla odeslána. K ověření úspěšného doručení dat přijímací strana posílá informace o přijetí paketu. Pokud by odesílatel před odesláním dalšího paketu musel čekat na potvrzení předešlého, vznikala by nevyužitá časová mezera. Pro urychlení celého procesu se neposílá pouze jeden paket, ale více paketů najednou v takzvaném okně (window). Po potvrzení přijetí všech paketů z daného okna, může být odesláno další okno. Pokud síť v důsledku zahlcení začne pakety zahazovat, TCP protokol toto zahlcení rozpozná v podobě nepotvrzených paketů. Na zahazování paketů odesílatel reaguje zmenšením velikosti okna odesílaných paketů a opětovným zasláním zahozených dat.

Na obrázku 2.6 je zobrazen TCP paket. Jednotlivé části jsou popsány dále.

- **Source Port** (16 bitů) označuje číslo zdrojového portu.
- **Destination Port** (16 bitů) označuje číslo cílového portu.
- **Sequence Number** (32 bitů) udává pořadové číslo prvního bytu dat v daném paketu.
- **Acknowledgment Number** (32 bitů) je hodnota, která je očekávána v poli Sequence Number v příštím paketu, pokud je pole ACK nastaveno na 1.

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	Source port																Destination port															
32	Length																Checksum															
64	Data																															

Obrázek 2.7: UDP paket

- **Data Offset** (4 bity) je délka hlavičky v 32-bitových slovech.
- **Reserved** (3 bity) vždy obsahuje hodnotu 0.
- **Flags** (9 bitů) obsahuje řídící příznaky.
- **Window** (16 bitů) je počet bytů, který odesílatel tohoto paketu očekává.
- **Checksum** (16 bitů) je kontrolní součet hlavičky a dat.
- **Urgent Pointer** určuje, kde končí urgentní data.
- **Options** (násobky 8 bitů) jsou rozšiřující pole, která mohou následovat za hlavičkou.
- **Padding** zajišťuje zarovnání hlavičky na násobky 32 bitů.
- **Data** jsou data obsažená v TCP paketu.

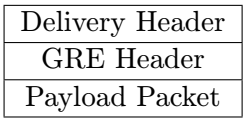
K zajištění spolehlivého přenosu je nutné vyvinout větší úsilí, než u nespolehlivého přenosu. Proto si některé služby vystačí s protokolem UDP, který spolehlivý přenos nezajišťuje.

2.4.2 UDP

UDP (User Datagram Protocol) slouží k rychlé a jednoduché komunikaci. Nezajišťuje spolehlivý přenos. O správné přeuspořádání a potvrzování paketů se musí postarat aplikační vrstva.

Na obrázku 2.7 je zobrazen UDP paket. Jednotlivé části jsou popsány dále.

- **Source Port** (16 bitů) udává zdrojový port. Tato položka slouží k případné odpovědi na UDP paket.
- **Destination Port** (16 bitů) udává cílový port.
- **Length** (16 bitů) udává délku hlavičky a dat v bytech. Z toho vyplývá, že nejnižší možná hodnota v poli Length je 8.
- **Checksum** (16 bitů) udává kontrolní součet hlavičky a dat. Pokud odesílatel nevygeneruje kontrolní součet, potom je hodnota pole Checksum rovna 0.
- **Data** jsou data obsažená v UDP paketu.



Obrázek 2.8: GRE zapouzdření

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	Source port																Destination port															
32	Length																Checksum															
64	Data																															

Obrázek 2.9: GRE hlavička

2.5 Tunelovací protokoly

Tunelovací protokol je protokol umožňující přenášet prostřednictvím síťového spojení jiná síťová spojení. Tento postup porušuje sled ISO/OSI vrstev. Příkladem může být tunelovací protokol 6in4, kde je IPv6 protokol přenášen v protokolu IPv4. Tedy třetí vrstva modelu ISO/OSI následuje za třetí vrstvou. Protokolem 6in4 je umožněno využívat IPv6 protokol v síti, která IPv6 nepodporuje.

Další motivací používat tunelovací protokoly může být potřeba obejít restrikce sítě. Pokud nastavení sítě umožňuje komunikaci pouze přes HTTP protokol, lze požadované protokoly zapouzdřit do HTTP protokolu. Tímto způsobem se obejde restrikce sítě.

K vytvoření virtuálního připojení do vzdálené lokální sítě se využívá technik nazvaných VPN (Virtual Private Network). Tyto techniky umožňují vytvořit spojení se vzdálenou lokální sítí na druhé vrstvě ISO/OSI přes vyšší vrstvy ISO/OSI. Například přes čtvrtou vrstvu, kde jsou využívány protokoly TCP a UDP.

Pokud zařízení umístěné za NATem potřebuje získat přístup do IPv6 sítě, nelze využít protokolů na způsob 6in4 tunelu. K tomuto účelu jsou navrženy specializované protokoly. Například lze využít Teredo tunel od společnosti Microsoft.

2.5.1 GRE

GRE (Generic Routing Encapsulation) [9] je tunelovací protokol vytvořený společností Cisco Systems. Cílem protokolu je vytvořit virtuální spojení typu point-to-point, které dokáže zapouzdřit protokoly na internetové vrstvě do jiných protokolů na internetové vrstvě.

Na obrázku 2.8 je znázorněno zapouzdření paketu. Hlavička GRE paketu je znázorněna na obrázku 2.9. Jednotlivé položky jsou popsány dále.

- **Checksum Present** (bit 0): Pokud je nastaven na 1, je pole Checksum přítomno a obsahuje validní hodnotu.

  Pokud je nastaven příznak Checksum Present nebo příznak Routing Present na 1, musí být v hlavičce přítomny obě pole Checksum a Offset.

- **Routing Present** (bit 1): Pokud je nastaven na 1, jsou pole Offset a Routing přítomny a obsahují validní hodnoty.

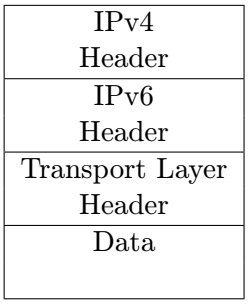
Pokud je nastaven příznak Checksum Present nebo příznak Routing Present na 1, musí být v hlavičce přítomny obě pole Checksum a Offset.

- **Key Present** (bit 2): Pokud je nastaven na 1, je pole Key přítomno a obsahuje validní hodnotu. Pokud je nastaven na 0, pole Key v hlavičce přítomno není.
- **Sequence Number Present** (bit 3): Pokud je nastaven na 1, je pole Sequence Number přítomno. Pokud je nastaven na 0, pole Sequence Number v hlavičce přítomno není.
- **Strict Source Route** (bit 4): Nastavuje se pouze ve speciálních případech.
- **Recursion Control** (bity 5-7): Obsahuje číslo přidaného zapouzdření, které je dovolené. Výchozí hodnota by měla být 0.
- **Version Number** (bity 13-15): Vyjadřuje verzi protokolu GRE. Pokud se jedná o základní verzi protokolu, musí být tato hodnota nastavena na 0.
- **Protocol Type** (16 bitů): Obsahuje typ protokolu, který je do GRE zapouzdřen. Protokoly typu IP jsou označeny hodnotou 0800.
- **Offset** (16 bitů): Udává počet bytů od začátku pole Routing. Toto pole je přítomno, pokud jsou příznaky Routing Present nebo Checksum Present nastaveny na 1. Obsahuje validní hodnotu pouze tehdy, pokud je příznak Routing Present nastaven na 1.
- **Checksum** (16 bitů): Obsahuje kontrolní součet GRE hlavičky a zapouzdřených dat. Toto pole je přítomno, pokud jsou příznaky Routing Present nebo Checksum Present nastaveny na 1. Obsahuje validní hodnotu pouze tehdy, pokud je příznak Checksum Present nastaven na 1.
- **Key** (32 bitů): Obsahuje číslo, které vložil účastník vytvářející tento GRE paket. Může být využito k autentizaci zdroje. Pole Key je přítomno, pokud je pole Key Present nastaveno na 1.
- **Sequence Number** (32 bitů): Obsahuje pořadové číslo paketu, podle kterého je možné pakety po přijetí seřadiť. Pole Sequence Number je přítomno, pokud je příznak Sequence Number Present nastaven na 1.
- **Routing** (proměnná velikost): Pole je přítomno, pokud je příznak Routing Present nastaven na 1.

2.5.2 6in4

Klíčem k úspěšnému nasazení IPv6 protokolu je kompatibilita s IPv4 protokolem, který je využíván ve většině současných sítích. Pro tento účel byl definován protokol 6in4 [10]. Hlavním úkolem protokolu 6in4 je zapouzdřit IPv6 datagramy do IPv4 datagramů. Ukázka zapouzdření je znázorněna na obrázku 2.10 IPv4 hlavička je bezprostředně následována IPv6 hlavičkou. Pro rozpoznání tunelu 6in4 je do pole Protokol v IPv4 hlavičce vložena hodnota 41, která značí tunel 6in4. Pokud tedy chce zařízení z IPv4 sítě komunikovat se zařízením v IPv6 síti, zapouzdří IPv6 datagram protokolem 6in4. Tento datagram je doručen k síťovému zařízení, které je vybaveno jak IPv4 konektivitou tak IPv6 konektivitou. Toto zařízení odstraní IPv4 hlavičku a pošle datagram dále jako normální IPv6 datagram.

Postupem času, kdy se přestane využívat protokol IPv4, vymizí také protokol 6in4.



Obrázek 2.10: Zapouzdření IPv6 do IPv4

2.5.3 Teredo

Teredo [11] je tunelovací protokol vyvinutý společností Microsoft. Jeho cílem je zpřístupnit zařízením umístěným za jedním nebo několika IPv4 Network Address Translations (NATy) IPv6 konektivitu. Princip spočívá v zapouzdření IPv6 datagramu do IPv4 UDP paketu. Tímto zapouzdřením je zajištěna velká propustnost skrz NAT zařízení. Ke správné činnosti služby Teredo jsou zapotřebí následující typy zařízení.

- **Teredo Client** je zařízení disponující IPv4 konektivitou. Toto zařízení vyžaduje spojení se zařízením nativně připojeným do IPv6 sítě.
- **Teredo Server** je zařízení disponující IPv4 a zároveň IPv6 konektivitou. Slouží jako pomocník při poskytování IPv6 konektivity Teredo klientům. Pomáhá zjistit, za jakým typem NATu se klient nachází. Teredo Server naslouchá na portu 3544.
- **Tered Relay** je zařízení disponující IPv4 a zároveň IPv6 konektivitou. Slouží jako IPv6 router, přes který jsou přenášena veškerá data mezi Teredo klientem a zařízením z IPv6 sítě.

Při analýze jednotlivých paketů je velmi obtížné detekovat, že jde o protokol Teredo, protože UDP porty jsou vybírány dynamicky.

Protokol Teredo je koncipován jako dočasný protokol. V momentě, kdy budou klienti disponovat IPv6 konektivitou, přestane být Teredo protokol potřeba.

Kapitola 3

Klasické rozpoznání protokolů

Při klasickém rozpoznávání typu protokolu je nutné nahlédnout do příslušných polí v předcházejícím protokolu. Pokud je paket zabalen do IPv4, jedná se o pole Protocol. U zapouzdření do IPv6 jde o pole Next header. V těchto polích je hodnota vycházející z tabulku 2.1, která určuje následující protokol.

Některé tunelovací protokoly ve své hlavičce také uvádějí následující protokol. Tyto protokoly lze rozpoznávat a zpracovávat klasickými postupy. Příkladem je protokol GRE, u kterého je typ následujícího protokolu umístěn do pole Protocol Type.

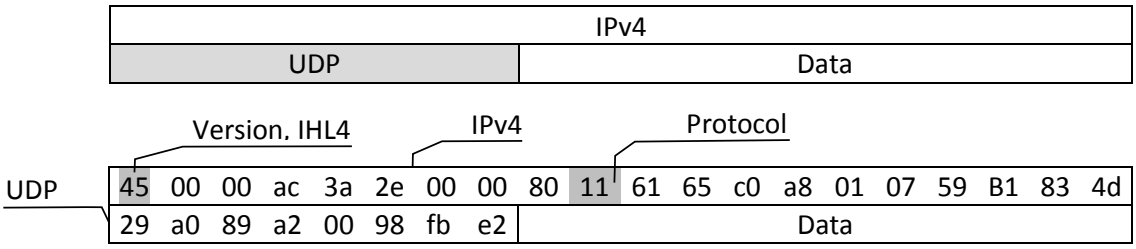
3.1 Příklady klasického rozpoznání

Postup klasického rozpoznání je vždy stejný. Nejdříve se podle prvních čtyř bitů zjistí typ IP protokolu. Podle typu protokolu jsou určeny pozice a velikostí polí, která jsou potřebné k dalšímu rozpoznávání. Ke zjištění offsetu protokolu následujícího za IP je nutné zjistit velikost IP hlavičky. Velikost IPv4 hlavičky je dána polem IHL, které leží za prvními čtyřmi bity v hlavičce určujícími verzi protokolu. IPv6 hlavička má pevně danou velikost hlavičky 40 bytů. Je však následována rozšiřujícími hlavičkami, které jsou vkládány mezi IPv6 hlavičky a následující protokol. Každá rozšiřující hlavička obsahuje číslo následující rozšiřující hlavičky nebo číslo protokolu. Tyto hlavičky je nutné zpracovat a dostat se až k následujícímu protokolu.

Dále je zjištěn typ protokolu za IP. Typ je uložen v poli Protocol nebo NextHeader. Podle typu je protokol zpracován. Jedná-li se o protokol, který ve své hlavičce nemá informaci o dalším protokolu, například protokoly UDP a TCP. Je zpracování ukončeno. Pokud je informace v hlavičce obsažena, například GRE nebo IP, celý proces se zjišťováním velikosti hlavičky a typu následujícího protokolu se opakuje.

3.1.1 Rozpoznání TCP/IP

Pro znázornění klasického rozpoznání TCP/IP protokolů je použit příklad IPv4 protokolu následovaný UDP protokolem. Jedná se o nejjednodušší typ zpracování. První čtyři bity obsahují číslo 4, což určuje IPv4. Velikost je určena následujícími čtyřmi bity v poli IHL. Typ protokolu UDP je dán hodnotou 11 hexadecimálně (17 dekadicky) v poli Protocol. UDP hlavička neobsahuje informace o typu přenášených dat, proto zde klasické rozpoznávání končí.

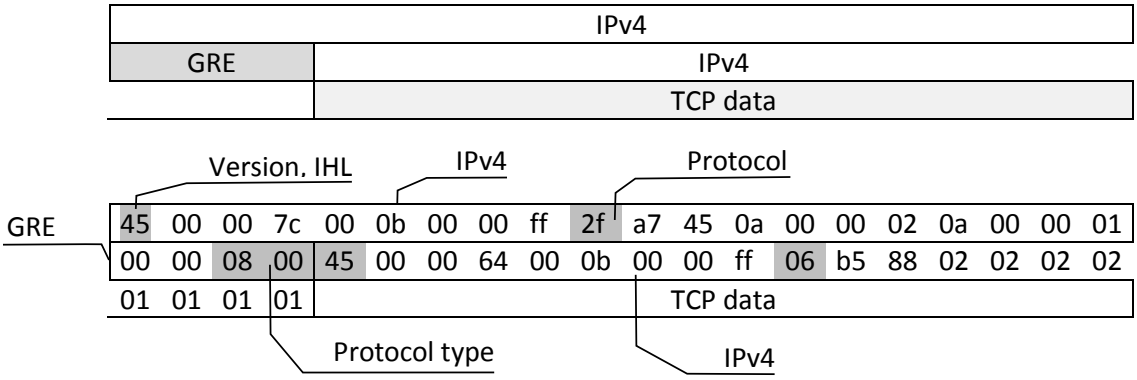


Obrázek 3.1: Ukázka IPv4 protokolu následovaného UDP protokolem

3.1.2 Rozpoznání v tunelovacím protokolu GRE

Klasické rozpoznání transportního protokolu v tunelovacím protokolu je znázorněno na příkladu GRE zapouzdření.Na obrázku 3.2 je znázorněno zapouzdření IPv4 protokolu do tunelovacího protokolu GRE. Zvýrazněna jsou pole, která jsou využívána k rozpoznání protokolů.

První čtyři bitu určují verzi IPv4. V poli Protokolu je uloženo číslo 2f v hexadecimální soustavě (47 v desítkové soustavě). Toto číslo určuje, že následující protokol bude GRE. V protokolu GRE je v poli Protocol type vloženo číslo 0800 hexadecimálně. Toto číslo podle dokumentu [13] určuje, že dalším protokolem bude IPv4. Postup zpracování IPv4 zapouzdřeného do GRE je identický s dosavadním postupem.



Obrázek 3.2: Ukázka zapouzdření IPv4 protokolu do tunelovacího protokolu GRE

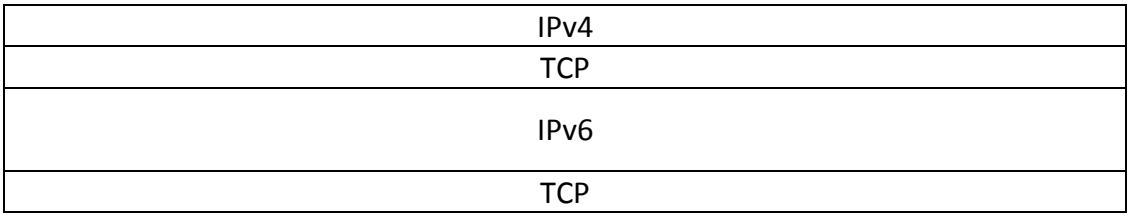
3.2 Problémy klasického rozpoznání

Klasické rozpoznání selhává v případech, kdy je za rozpoznané protokoly vložen jeden nebo více protokolů, který ve své hlavičce neobsahují informace o typech následujících protokolů. Data následovaná za těmito protokoly není možné bez znalosti kontextu komunikace zpracovat.

Pokud se rozpoznávaný paket vyskytuje za TCP nebo UDP protokolem, není jej možné určit, protože TCP ani UDP hlavička neobsahuje informace o následujícím protokolu. Z tohoto důvodu se rozpoznávání stává nemožným.

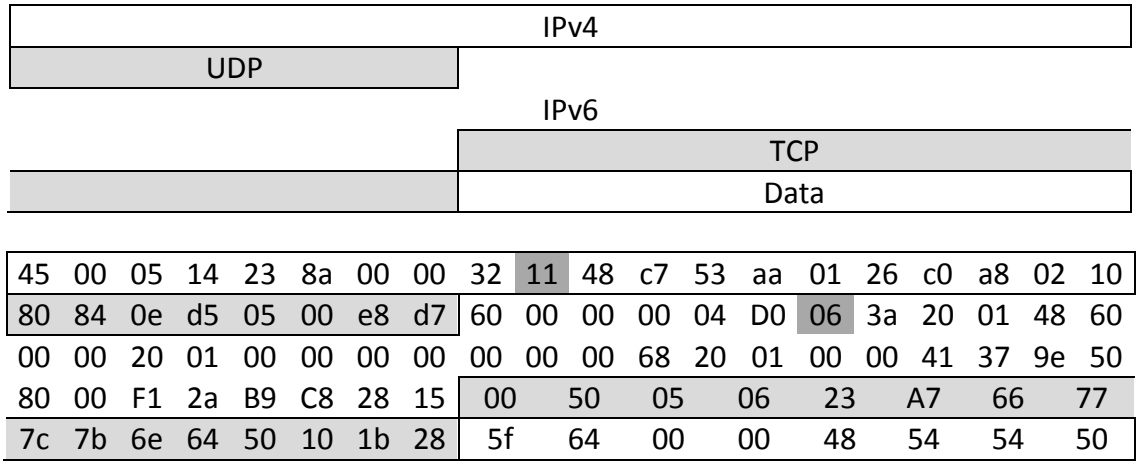
Na obrázku 3.3 je znázorněno zapouzdření IPv6 komunikace do TCP protokolu. Klasickým rozpoznáváním lze určit pouze protokol IPv4 a za ním následující protokol TCP. IPv6 není možné zjistit, protože TCP hlavička neobsahuje informaci o typu přenášených dat.

I přesto, že první čtyři bity obsahují číslo 6, které odpovídá IPv6 protokolu, není možné s jistotou prohlásit data za IPv6 protokol. V prvních čtyřech bitech se číslo šest mohlo náhodně vyskytnout z přenášených dat.



Obrázek 3.3: Ukázka zapouzdření IPv6 komunikace do TCP protokolu

Například tunelovací protokol Teredo je přenášen v UDP paketu. Protože UDP hlavička neobsahuje informace o následujícím paketu, bez kontextu není možné ze zachycené komunikace určit, o který tunelovací protokol se jedná. V tomto případě je nutné použít automatizovanou detekci. Na obrázku 3.4 jsou zvýrazněny byty, které určují typ následujícího protokolu.



Obrázek 3.4: Ukázka zapouzdření IPv6 v protokolu Teredo

Kapitola 4

Signatury protokolů

Pro automatizovanou detekci transportních protokolů je nutné nejprve nalézt IP protokol, který jim předchází. Následně pomocí signatur a kontrolního součtu zjistit, jestli zkoumaná část dat odpovídá transportnímu protokolu.

Signaturami se rozumí hodnoty určených polí, které musí splňovat určité podmínky odpovídající zkoumanému protokolu. Například při hledání TCP protokolu, následujícího za IPv4 hlavičkou, musí pole Protocol v IPv4 hlavičce obsahovat hodnotu 6.

4.1 Signatury IP

Hlavičky IPv4 a IPv6 se značně liší. Liší se také jejich signatury, podle kterých jsou rozpoznatelné. IPv6 hlavička postrádá oproti IPv4 hlavičce některá pole. Nejzásadnějším rozdílem je však absence kontrolního součtu v IPv6 hlavičce.

4.1.1 Signatury IPv4

Na obrázku 4.1 jsou zvýrazněny části hlavičky, podle kterých je možné určit IPv4 hlavičku. Níže jsou rozebrány hodnoty, které musí zvýrazněné části obsahovat, aby se mohlo jednat o IPv4 hlavičku následovanou TCP nebo UDP protokolem.

- **Version** (hodnota 4) Pole udává verzi IP protokolu. Protokol IPv4 je značen číslem 4.
- **IHL Internet Header Length** (hodnota v rozmezí 5 - 15) Pole udává velikost hlavičky bez přenášených dat. Jedná se o velikost určující násobky 32 bitů.

<table>
<tr><td>Octet</td><td colspan="8">0</td><td colspan="8">1</td><td colspan="8">2</td><td colspan="8">3</td></tr>
<tr><td>Bit</td><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td>0</td><td colspan="4">Version</td><td colspan="4">IHL</td><td colspan="6">DSCP</td><td colspan="2">ECN</td><td colspan="16">Total length</td></tr>
<tr><td>32</td><td colspan="16">Identification</td><td colspan="3">Flags</td><td colspan="13">Fragment offset</td></tr>
<tr><td>64</td><td colspan="8">Time to live</td><td colspan="8">Protocol</td><td colspan="16">Header checksum</td></tr>
<tr><td>96</td><td colspan="32">Source IP address</td></tr>
<tr><td>128</td><td colspan="32">Destination IP address</td></tr>
</table>

Obrázek 4.1: Signatury v IPv4 hlavičce

Minimální hodnota 5 je odvozena ze skutečnosti, že součet všech povinných polí v hlavičce je 20 bytů (160 bitů). 5 x 32 = 160.

Maximální hodnota 15 je dána velikostí IHL pole (4 bity).

- **Total length** (hodnota v rozmezí 28 - 65 535) Pole udává celkovou velikost IPv4 paketu v bytech zahrnující jak hlavičku tak následující data.

  Minimální hodnota 28 bude použita pro nejmenší možnou IPv4 hlavičku (20 bytů) následovanou UDP paketem, který bude obsahovat pouze hlavičku (8 bytů) bez dat. Hodnota 40 bude použita pro nejmenší možnou IPv4 hlavičku (20 bytů) následovanou TCP paketem, který bude obsahovat pouze nejmenší možnou hlavičku (20 bytů) bez dat.

  Maximální hodnota 65 535 je dána velikostí Total length pole (16 bitů).

- **Protocol** (hodnota 6 nebo 17) Pole udává číslo protokolu, který následuje za IPv4 hlavičkou. Z tabulky 2.1 je patrné, že TCP protokol má přiřazeno číslo 6 a protokol UDP číslo 17.

- **Header checksum** Pole udává kontrolní součet, který je počítán z IPv4 hlavičky [1].

4.1.2 Signatury IPv6

Na rozdíl od IPv4 hlavičky má IPv6 hlavička podstatně méně signatur, podle kterých ji lze rozpoznat. Největším rozdílem je absence kontrolního součtu. Nutnost provádět kontrolní součet tak závisí na TCP nebo UDP protokolu.

Na obrázku 4.2 jsou zvýrazněny signatury, které naznačují přítomnost IPv6 hlavičky. Konečné rozhodnutí, zda se jedná o IPv6 hlavičku či nikoli, se však z důvodu nepřítomného kontrolního součtu může provést až po inspekci paketů ležících za IPv6 hlavičkou.

Jednotlivé signatury jsou popsány níže.

- **Version** (hodnota 6) Pole udává verzi IP protokolu. IPv6 má přiřazenu hodnotu 6.

- **Payload length** (hodnota v rozmezí 8 - 65535) Pole udává velikost dat za IPv6 hlavičkou včetně rozšířených hlaviček

  Minimální hodnota 8 udává samotnou UDP hlavička (8 bytů) bez dat.

  Hodnota 20 udává nejmenší možnou TCP hlavička (20 bytů) bez dat.

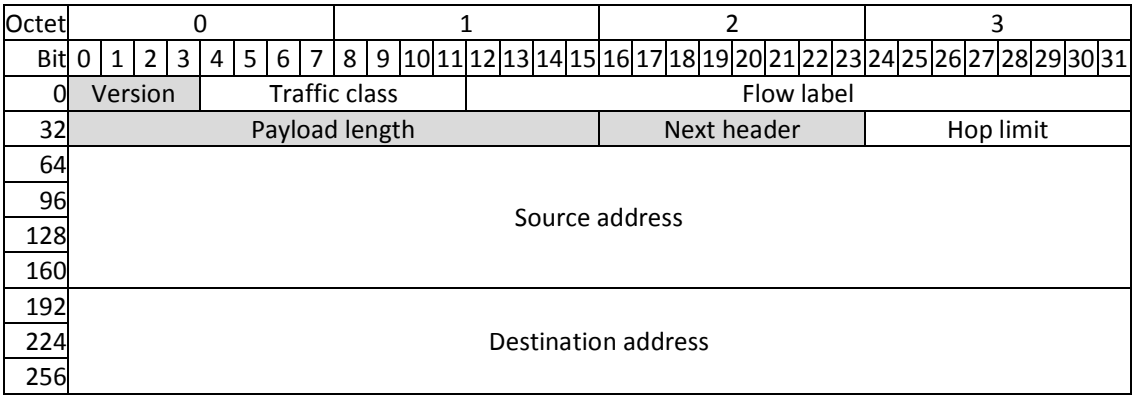
  Maximální hodnota 65 535 je dána rozsahem pole Payload length (16 bitů).

- **Next Header** (hodnota 6 nebo 17) Pole udává číslo následující hlavičky. Přehled čísel je uveden v tabulce 2.1.

  Hodnota 6 udává následující protokol TCP.

  Hodnota 17 udává následující protokol UDP.

  Pokud je v poli jiná hodnota než 6 nebo 17, můžou mezi transportním protokolem a IPv6 hlavičkou ležet rozšířené hlavičky. V tomto případě je potřeba přeskákat přes všechny rozšířené hlavičky až k transportnímu protokolu.



Obrázek 4.2: Signatury v IPv6 hlavičce

4.2 Signatury TCP a UDP

Vzhledem k tomu, že se kontrolní součet protokolů TCP a UDP dopočítává z informací, které jsou obsaženy v IP hlavičce, je nutné nejprve nalézt IP protokol. Po nalezení IP protokolu je zapotřebí zkontrolovat signatury transportních protokolů, které jsou popsány dále.

4.2.1 Signatury TCP

V případě protokolu TCP téměř nezáleží na verzi IP paketu, do kterého je zapouzdřen. Drobná odlišnost je pouze v dopočítání kontrolního součtu [7]. Ostatní signatury se nemění.

Na obrázku 4.3 jsou zvýrazněny signatury TCP hlavičky. Jejich popis je uveden níže.

- **Data offset** (hodnota v rozmezí 5 - 15) Pole udává velikost TCP hlavičky. Velikost je udávána v 32 bitových slovech.

  Minimální hodnota 5 je dána nejmenší možnou velikostí TCP hlavičky.

  Maximální hodnota 15 je dána velikostí pole Data offset - 4 bity.

- **Reserved** (hodnota 0) Pole o velikosti 3 bity je rezervováno. Musí vždy obsahovat hodnotu 0.

- **Checksum** Pole udává kontrolní součet, který je počítán z TCP paketu a z informací obsažených v IP hlavičce. Postup výpočtu kontrolního součtu je uveden v dokumentu [7].

4.2.2 Signatury UDP

Protokol UDP má rozdílný přístup k počítání kontrolního součtu v závislosti na verzi IP paketu, ve kterém je zapouzdřen [8]. Při zapouzdření do IPv4 je pole checksum nepovinné. Pokud je UDP paket přenášen v paketu IPv6, je pole checksum povinné.

Na obrázku 4.4 jsou zvýrazněny signatury UDP hlavičky. Jejich popis je uveden níže.

- **Length** (hodnota v rozmezí 8 - 65535) Pole udává celkovou velikost UDP paketu v bytech. Do celkové velikosti se počítá hlavička a data.

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	Source port																Destination port															
32	Sequence number																															
64	Acknowledgment number																															
96	Data offset				Reserved 000				NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window size														
128	Checksum																Urgent pointer															

Obrázek 4.3: Signatury v TCP hlavičce

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	Source port																Destination port															
32	Length																Checksum															

Obrázek 4.4: Signatury v UDP hlavičce

Minimální hodnota 8 je dána UDP hlavičkou, která je velká 8 bytů.

Maximální hodnota 65535 je dána velikostí pole Length 16 bitů.

- **Checksum** Pole udává kontrolní součet, který je počítán z UDP paketu a z informací obsažených v IP hlavičce. Postup výpočtu kontrolního součtu je uveden v dokumentu [8].

  Kontrolní součet v IPv4 je nepovinný. Pro IPv6 je přítomnost kontrolního součtu povinná.

Kapitola 5

Postup automatizované detekce

Tato kapitola popisuje zpracování zapouzdřených protokolů na základě TCP, UDP a IP signatur. Hlavním úkolem automatizované detekce je nalézt v neznámých datech začátek protokolu IP následovaný TCP nebo UDP protokolem. Při pozitivním nálezu je tento blok dat předán ke zpracování klasickým metodám. Automatizovaná detekce také umožňuje rekurzivně hledat zapouzdřené transportní protokoly v již zapouzdřených protokolech. Popsaná metoda není závislá na určitých typech tunelovacích protokolů. Je tedy schopná nalézt transportní protokol v jakémkoli tunelovacím protokolu.

5.1 Obecný postup

Automatizovaný algoritmus prochází postupně pole bytů následujících za poslední známou hlavičkou v rámci. Od zkoumaného bytu je vytvořeno imaginární okno, které obsahuje byty potřebné k pojmutí dvojice TCP/IP protokolů. Imaginární okno může měnit svoji velikost například na základě pole Internet Header Length v IPv4 hlavičce.

Podle prvních čtyř bitů (pole Version IP hlavičky) v imaginárním okně je rozhodnuto, zda se jedná o IPv4 nebo o IPv6. Pokud verze odpovídá jedné z verzí IP protokolů, začnou se ostatní hodnoty testovat na signatury TCP/IP protokolů. Při první neshodě je porovnání zastaveno a okno se posune na další byt v pořadí. Pokud všechny hodnoty odpovídají signaturám některé z dvojic TCP/IP protokolů, je proveden kontrolní součet. Pokud kontrolní součet odpovídá, začne se právě nalezený blok bytů považovat za regulérní TCP/IP protokoly a je předán klasickým metodám ke zpracování. Algoritmus nekončí prohledávat pole bytů po prvním pozitivním nálezu. Při nálezu pokračuje kontrola bytů za oblastí nalezených protokolů.

Níže je popsán postup testování signatur jednotlivých TCP/IP protokolů. Pro přehlednost je vypsán seznam kroků. Prní část každého kroku znázorňuje hlavičku protokolu, ve které se nachází zkoumané pole. Druhá část je název zkoumaného pole. Třetí část znázorňuje pravidlo, vůči kterému je hodnota testována.

5.2 Detekce v IPv4

Nejprve jsou zkontrolovány IPv4 signatury. Pokud všechny odpovídají, kontrola přejde na signatury TCP a UDP protokolu. Postupné seznamy kroků kontroly jsou zobrazeny níže.

TCP

1. IPv4 Version = 4
2. IPv4 IHL >= 20
3. IPv4 Protocol = 6
4. IPv4 Total length >= 40
5. TCP Reserved = 0
6. TCP Data offset >= 5

UDP

1. IPv4 Version = 4
2. IPv4 IHL >= 20
3. IPv4 Protocol = 17
4. IPv4 Total length >= 28
5. UDP Length > 8

Pokud všechny hodnoty odpovídají, je vypočítán kontrolní součet IPv4 hlavičky. Při shodě vypočítaného kontrolního součtu s hodnotou v poli Header checksum je nález považován za validní.

5.3 Detekce v IPv6

Po zkontrolování verze IP protokolu je zkoumána přítomnost rozšiřujících hlaviček. Pokud žádné rozšiřující hlavičky nejsou přítomny, leží transportní protokol bezprostředně za IPv6 hlavičkou. Při výskytu jedné nebo více rozšiřující hlaviček je potřena projít až na jejich konec, kde leží transportní protokol. Pro procházení skrze rozšířené hlavičky se využívá skutečnost, že všechny rozšiřující hlavičky mají stejný formát znázorněný na obrázku 2.5 popsaný v dokumentu [6], ze kterého je patrné, že u všech rozšiřující hlaviček leží číslo následujícího protokolu v 1. bytu a délka rozšiřující hlavičky vždy v 2. bytu. Z tohoto důvodu je možné procházet i neznáme hlavičky. U rozšířené hlavičky se kontroluje, zda pole Next header obsahuje číslo TCP nebo UDP protokolu. Pokud neobsahuje, začne se zkoumat další rozšiřující hlavička. Při nalezení poslední rozšiřující hlavičky, za kterou leží transportní protokol, se přistoupí ke zkoumání TCP nebo UDP signatur. Postupný seznam kroků zkoumajících signatury je popsán níže.

TCP

1. IPv6 Version = 6
2. IPv6 Next header = 6
3. IPv6 Payload length >=20
4. TCP Reserved = 0
5. TCP Data offset >= 5

UDP

1. IPv6 Version = 6
2. IPv6 Next header = 17
3. IPv6 Payload length >=8
4. UDP Length >= 8

Pokud všechny hodnoty odpovídají, je vypočítán kontrolní součet TCP nebo UDP paketu. Při shodě vypočítaného kontrolního součtu s hodnotou v poli Checksum je nález považován za validní.

5.4 Příklad detekce

Pro názornost jsou na obrázku 5.1 znázorněna uměle vytvořená data, která poslouží jako příklad automatizované detekce. Níže je popsán postup, kterým se data zpracují.

bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	IPv4																			
20	TCP																			
40	Data																			
60	IPv4																			
80	UDP								Data											

bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	45	00	00	64	00	01	00	00	40	06	d2	71	01	01	01	01	7f	00	00	01
20	00	50	4e	fd	bd	f6	64	8b	8a	1d	43	9b	50	10	00	7b	8c	b6	00	00
40	13	17	06	45	00	00	39	11	22	60	aa	70	05	8b	50	94	08	f6	54	32
60	45	00	00	28	4d	7b	40	00	38	11	95	7b	4d	4b	4b	2a	c0	a8	01	10
80	29	a0	89	a2	00	14	4a	fe	99	98	97	96	95	94	93	92	91	90	89	88

Obrázek 5.1: Uměle vytvořený rámec pro názornou ukázku automatizované detekce

1. Klasická metoda zpracuje prvních 40 bytů, ve kterých rozpozná IPv4 protokol následovaný protokolem TCP. Zbylých 60 bytů není schopna rozpoznat, proto je předá automatizované detekci.
2. Od 40. bytu jsou byty testovány na signatury protokolů IPv4 nebo IPv6. První je hledána verze 4 nebo 6. Verze 4 odpovídá prvním čtyřem bitů v bytu 43.
3. Hodnota druhé poloviny 43. bytu je testována na signaturu IHL. Hodnota 5 odpovídá.
4. Hodnota v 52. bytu je testována na signaturu Protocol. Zde je očekáváno číslo 6 nebo 17 (dekadicky). Hodnota neodpovídá signatuře, protože je zde hodnota 5. Další pole IPv4 není třeba kontrolovat. Automatizovaná detekce se vrátí na 44. byte.
5. Od 44. bytu je opět hledána verze IPv4 nebo IPv6. Obsah 49. bytu odpovídá verzi IPv6. Mohlo by se tedy jednat o začátek IPv6 hlavičky.

6. Další kontrolovanou signaturou pro IPv6 je Next header v bytu číslo 55. Pokud by se hodnota rovnala číslu 6 nebo 17, přešlo by se k testování signatur TCP nebo UDP protokolu. Obsah 55. je však roven 94 hexadecimálně (148 dekadicky). Hodnota 148 může značit číslo některé nestandardní rozšířené hlavičky. Podle struktury rozšířených hlaviček, která je znázorněna na obrázku 2.5, je zkontrolováno pole Hdr Ext Len v rozšířené hlavičce ležící na 56. bytu. Toto pole určuje délku rozšířené hlavičky v násobcích 8 bytů. Hodnota 8 by určovala délku rozšířené hlavičky na 64 bytů. Takto velká hlavička by se do kontrolovaného rámce nevešla. Proto se detekce vrátí na 50. byte.

7. Od 50. bytu je opět hledána verze IPv4 nebo IPv6. Byte s číslem 60 obsahuje v prvních čtyřech bytech hodnotu 4. Mohlo by se tedy jednat o IPv4 protokol.

8. Druhá polovina 50. bytu obsahuje číslo 5. Tyto čtyři biti představují pole IHL, které udává délku IPv4 hlavičky na přijatelných 20 bytů.

9. Hodnota 69. bytu je testována na signaturu Protocol. Hodnota 11 hexadecimálně (17 dekadicky) odpovídá protokolu UDP.

10. Po součtu velikosti IPv4 hlavičky 20 bytů a nejmenší možné velikosti UDP protokolu 8 bytů vychází číslo 28. Velikost obou hlaviček se do procházených dat vejde. Detekce může pokračovat.

11. Hodnota v 84. a 85. bytu udává pole Length v UDP hlavičce. Velikost 14 hexadecimálně (20 dekadicky) přesně odpovídá velikosti dat od začátku UDP hlavičky do konce zpracovávaného rámce. Tato signatura také odpovídá.

12. Posledním krokem je vypočítání kontrolního součtu IPv4 hlavičky. Pokud bude pole Checksum, ležící na 70. a 71. bytu, obsahovat korektní hodnotu, automatizovaná detekce nalezla IPv4 protokol následovaný transportním protokolem UDP. Blok dat od 60. bytu předá klasické metodě ke zpracování.

Kapitola 6

Testování

Tato kapitola zkoumá vlastnosti obou metod vytvořených v rámci této práce, které rozpoznávají a zpracovávají transportní protokoly v zapouzdřených tunelovacích protokolech. Je zde rozebírána otázka rychlosti zpracování paketů obou metod v závislosti na protokolech a velikosti přenášených dat. Další část řeší míru úspěšnosti automatizované detekce při různých variantách zapouzdření.

Pro účely testování byla vytvořena data, která obsahují velké množství náhodných hodnot, aby bylo odhaleno co nejvíce vlastností při zpracovávání zachycených dat. Pro každý test byl vygenerován soubor o velikosti 5 MB, který obsahovat náhodné kombinace číselných hodnot. Tento soubor se následně přenášel přes testované protokoly. Po zpracování zachycené komunikace byla vždy získaná data porovnána s originálním souborem, aby byly zjištěny případné rozdíly.

6.1 Výkonnost automatizované detekce

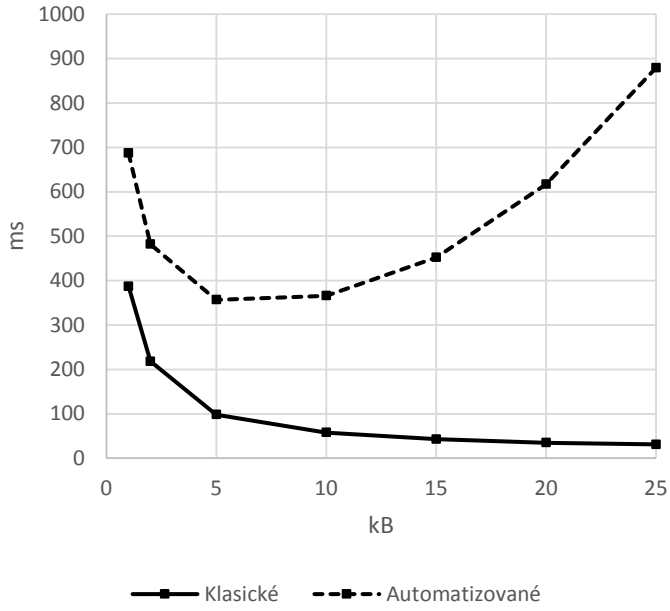
Při testování výkonnosti klasické a automatizované detekce byl vytvořen soubory s náhodnými hodnotami. Data ze souboru byla následně po určitých velikostech rozdělena a za pomoci Python knihovy Scapy byla zapouzdřena do testovaných protokolů. Pro měření výkonnosti automatizované detekce byla data posílána v protokolech 6in4 a GRE. Přijatá data byla zpracována stokrát klasickou a stokrát automatizovanou metodou. Výsledné doby zpracování byly zprůměrovány.

6.1.1 Doba zpracování přeneseného souboru

První část testování výkonnosti se zaměřila na dobu potřebnou ke zpracování zachycené komunikace obsahující kompletní data z předem připraveného souboru. Zkoumána byla doba zpracování na základě rozdílné velikosti dat v rámci. Zpracovávány byly rámce, které obsahovaly data o velikosti od 1 kB po 25 kB. Při zvětšování velikosti přenášených dat v jednom paketu se snižuje počet rámců potřebných k přenesení celého souboru.

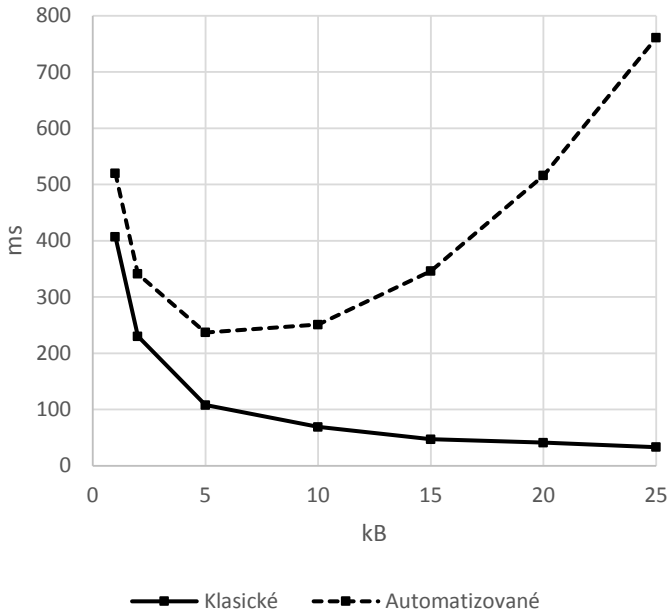
Klasická detekce se zvětšováním velikosti rámců vykazovala rychlejší zpracování celého souboru. Příčinou je snižující se celkový počet rámců nutný ke zpracování a také fakt, že klasická metoda neprochází data byte po bytu.

Automatizovaná detekce zpočátku provádí zpracování na základě zmenšujícího se počtu celkových rámců rychleji. Se vzrůstající velikostí dat však doba zpracování nezanedbatelně stoupá. Při velikosti okolo 10 kB se doba zpracování začíná zvětšovat. Tento jev je dán nutností zkoumat signatury protokolů byte po bytu.



Velikost dat v paketu [kB]	Klasické zpracování [ms]	Automatizované zpracování [ms]
1	387	687
2	218	482
5	98	357
10	58	366
15	43	452
20	35	617
25	31	879

Obrázek 6.1: Doba zpracování souboru o velikosti 5 MB přenášeného v protokolu 6in4

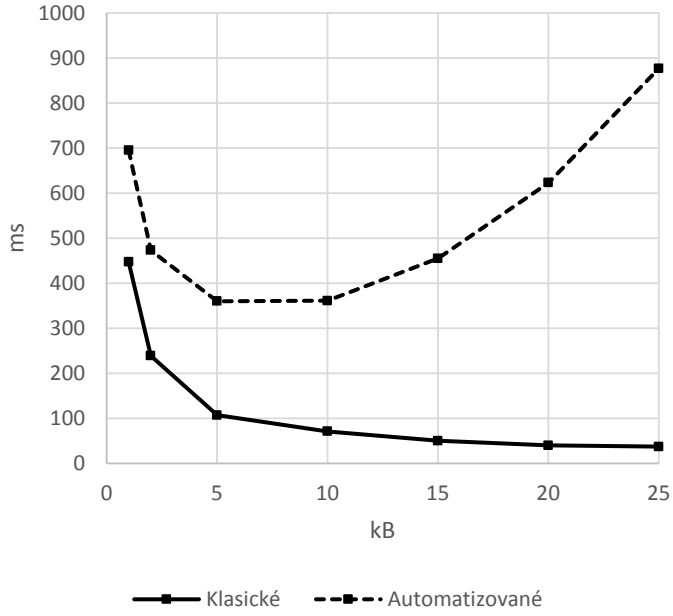


Velikost dat v paketu [kB]	Klasické zpracování [ms]	Automatizované zpracování [ms]
1	407	520
2	230	341
5	108	237
10	69	251
15	47	346
20	41	516
25	33	761

Obrázek 6.2: Doba zpracování souboru o velikosti 5 MB přenášeného v protokolu IPv4 zapouzdřeného do protokolu GRE

Obrázek 6.1 porovnává zpracování klasickou metodou s automatizovanou detekcí. Zobrazuje dobu zpracování zachycené komunikace, ve které je přenesen soubor o velikosti 5 MB. Tento soubor je přenášen v protokolu 6in4. Soubor byl postupně přenesen po blocích o velikostech od 1 kB po 50 kB. Na obrázku je patrná oblast mezi daty o velikosti mezi 5 kB a 10 kB, kde se vliv zpracování velkého počtu paketů vyrovná s velikostí procházených dat a doba zpracování začne narůstat.

Na obrázcích 6.2 a 6.3 je porovnáno zpracování klasickou metodou s automatizovanou



Velikost dat v paketu [kB]	Klasické zpracování [ms]	Automatizované zpracování [ms]
1	447	695
2	239	473
5	107	360
10	71	361
15	50	455
20	40	623
25	37	877

Obrázek 6.3: Doba zpracování souboru o velikosti 5 MB přenášeného v protokolu IPv6 zapouzdřeného do protokolu GRE

detekci. Je znázorněna dobra zpracování zachycené komunikace, která obsahuje přenesený souboru o velikosti 5 MB. Tento soubor je přenášen v tunelovacím protokolu GRE následovaného protokolem IPv4 (obrázek 6.2) nebo IPv6 (obrázek 6.3). Z porovnání obrázků je možné vyčíst, že doba zpracování je také závislá na IP protokolu. Delší doba zpracování protokolu IPv6 je způsobena zkoumáním potenciálních rozšířených hlaviček mezi IPv6 a transportním protokolem.

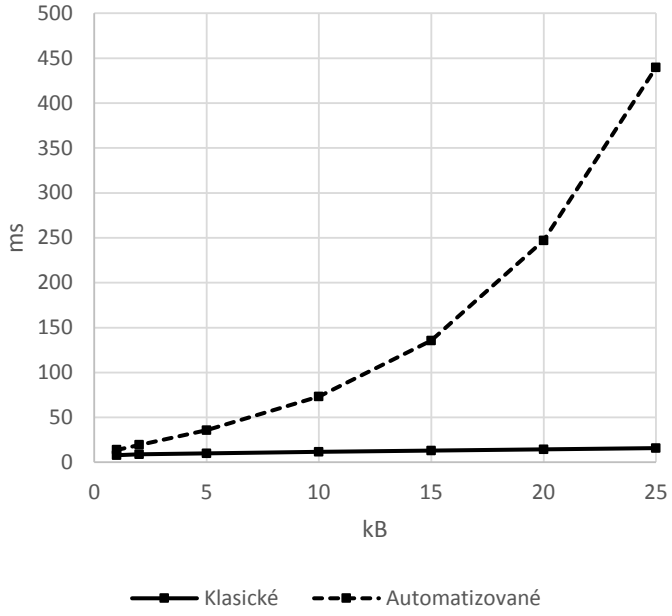
6.1.2 Doba zpracování jednoho rámce

Druhá část testování zkoumala dobu potřebnou ke zpracování jednoho rámce v závislosti na velikosti přenášených dat. Velikost dat přenášených v testovaných rámcích byla od 1 kB po 25 kB.

Na obrázku 6.4 je porovnání rychlosti zpracování jednoho paketu automatizovanou detekci oproti klasickému zpracování. Zpracována jsou náhodná data přenášená protokolem 6in4. Velikost přenášených dat v paketu se pohybuje od 1 kB po 50 kB. Na grafu je patrná značná závislost doby zpracování na velikosti přenášených dat při zpracování automatizovanou detekcí.

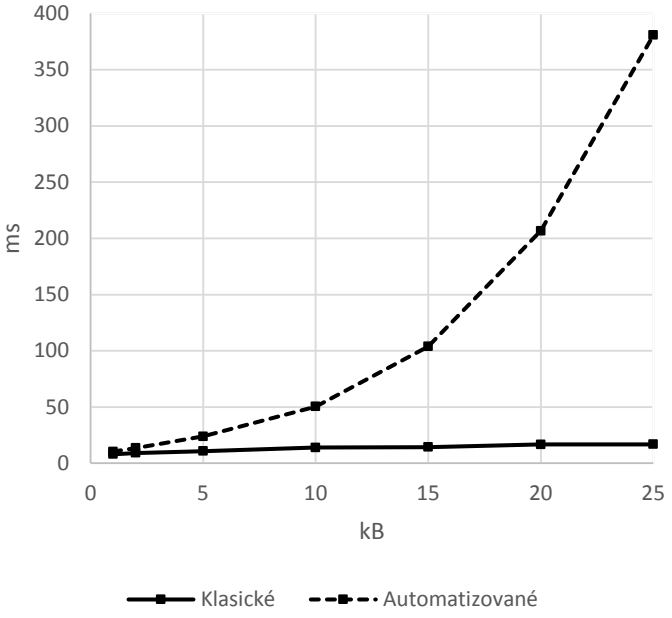
Obrázky 6.5 a 6.6 si kladou za cíl poukázat na výkonnostní dopad v závislosti na zpracovávané verzi IP protokolu. Oba obrázky znázorňují dobu potřebnou ke zpracování jednoho paketu automatizovanou detekcí a klasickým zpracováním. Velikost přenášených dat v paketu se pohybuje v rozmezí od 1 kB po 50 kB. Při porovnání obou obrázků je dobře viditelný časový dopad na výkonnost při nutnosti zpracovat IPv6 protokol kvůli rozšířeným hlavičkám.

Po zhodnocení výsledků je patrné, že klasické detekce netrpí tak markantním snížením rychlosti se zvětšující se velikostí dat jako automatizovaná metoda. Při několikanásobném zvětšení přenášených dat je zpomalení zpracování zanedbatelné. Automatizovaná detekce oproti tomu vykazuje značný nárůst doby zpracování při větším objemu zpracovávaných dat.



Velikost dat v paketu [kB]	Klasické zpracování [ms]	Automatizované zpracování [ms]
1	7,7	13,7
2	8,7	19,2
5	9,8	35,7
10	11,6	73,2
15	13,0	135,5
20	14,3	247,0
25	15,7	439,5

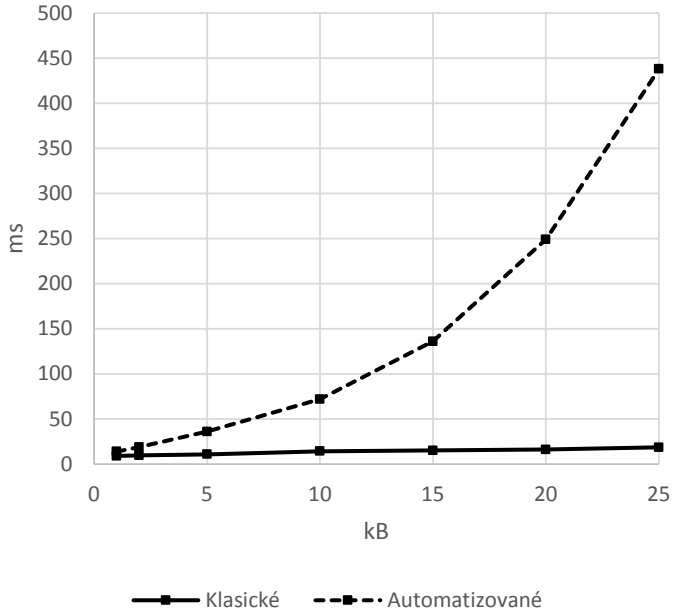
Obrázek 6.4: Doba zpracování paketu přenášeného v protokolu 6in4



Velikost dat v paketu [kB]	Klasické zpracování [ms]	Automatizované zpracování [ms]
1	8,1	10,4
2	9,2	13,6
5	10,8	23,8
10	13,9	50,4
15	14,3	103,8
20	16,7	206,6
25	16,9	380,8

Obrázek 6.5: Doba zpracování paketu přenášeného v protokolu IPv4 zapouzdřeného do protokolu GRE

Například při velikosti dat 15 kB je automatizovaná detekce přibližně desetkrát pomalejší než klasická metoda. V běžné komunikaci velikost dat jen zřídka překročí velikost jednoho kB. Proto při zpracování běžné komunikace bude automatizovaná detekce víceméně stejně výkonná jako klasické rozpoznání.



Velikost dat v paketu [kB]	Klasické zpracování [ms]	Automatizované zpracování [ms]
1	8,9	13,9
2	9,6	18,9
5	10,8	36
10	14,3	72
15	15,1	136
20	16,1	249
25	18,5	438

Obrázek 6.6: Doba zpracování paketu přenášeného v protokolu IPv6 zapouzdřeného do protokolu GRE

6.2 Úspěšnost automatizované detekce

Míra úspěšnosti automatizované detekce byla nejdříve testována na komunikaci, která neobsahovala tunelovací protokol. Jednalo se o zachycenou komunikaci běžných služeb jako Skype, YouTube.com, Mapy.cz nebo Stream.cz. V této části testu bylo zjišťováno, zda automatizovaná metoda nevynechává nebo nepřidává paketu. Obsah dat nebyl zkoumán. Na obrázku 6.7 jsou zobrazeny zkoumané služby. U každé služby je znázorněno, jakou posloupnost protokolu využívá. Pro lepší představu objemu testovaných dat je připojen počet rámců zkoumaných v jednotlivých testech. Počet nalezených transportních hlaviček vždy odpovídal počtu zkoumaných rámců.

Služba	Posloupnost protokolů	Počet rámců
Skype	IPv4 - UDP - Data	2243
Skype	IPv6 - UDP - Data	2220
Steam.cz	IPv4 - TCP - Data	25193
Steam.cz	IPv6 - TCP - Data	6351
YouTube.com	IPv4 - TCP - Data	13751
YouTube.com	IPv6 - TCP - Data	44457
Mapy.cz	IPv4 - TCP - Data	9874
Mapy.cz	IPv6 - TCP - Data	13309

Obrázek 6.7: Přehled zkoumaných služeb neobsahujících tunelovací protokol

Druhá část testování úspěšnosti zkoumala, zda jsou detekovaná data v transportních protokolech identická s daty vyslanými. Pro tyto účely byly generovány soubory s náhodným obsahem, aby bylo otestováno co nejvíce možností. Soubory byly pomocí knihovny Scapy zapouzdřeny. Dále byly prováděny testy na experimentálním sledu zapouzdřených paketů

za účelem otestovat úspěšnost na vícenásobném zapouzdření. Testovány byly posloupnosti protokolů typu IPv4-TCP-IPv4-TCP-Data1-IPv4-UDP-Data2 v různých obměnách transportních protokolů a v různých verzích IP protokolů. Data1 obsahovala data z jednoho souboru a Data2 ze souboru druhého. Na obrázku 6.8 jsou znázorněny rámce využité při testování úspěšnosti vícenásobného zapouzdření. Automaticky detekovaná data z obou nalezených transportních protokolů byla porovnána s originálními soubory. Ve všech případech byla shoda stoprocentní.

Posloupnost protokolů	Velikost Data1 v paketu	Velikost Data2 v paketu	Počet rámců
IPv4-TCP-IPv**4**-**UDP**-Data1-IPv**4**-**UDP**-Data2	1 kB až 25 kB	1 kB až 25 KB	2000 až 50 000
IPv4-TCP-IPv**4**-**TCP**-Data1-IPv**4**-**TCP**-Data2	1 kB až 25 kB	1 kB až 25 kB	2000 až 50 000
IPv4-TCP-IPv**6**-**UDP**-Data1-IPv**4**-**TCP**-Data2	1 kB až 25 kB	1 kB až 25 kB	2000 až 50 000
IPv4-TCP-IPv**6**-**TCP**-Data1-IPv**4**-**UDP**-Data2	1 kB až 25 kB	1 kB až 25 kB	2000 až 50 000
IPv4-TCP-IPv**6**-**UDP**-Data1-IPv**6**-**TCP**-Data2	1 kB až 25 kB	1 kB až 25 kB	2000 až 50 000
IPv4-TCP-IPv**4**-**TCP**-Data1-IPv**6**-**UDP**-Data2	1 kB až 25 kB	1 kB až 25 kB	2000 až 50 000

Obrázek 6.8: Přehled zkoumaných rámců obsahujících vícenásobné zapouzdření

Sada testů obsahovala také smyšlené tunelovací protokoly, které obsahovaly náhodný shluk číselných hodnot různé délky. Tyto shluky hodnot byly vkládány před hledanou dvojici TCP/IP protokolů. Jednalo se o posloupnosti typu IPv6-TCP-X-IPv4-UDP-Data, kde X označuje náhodný řetězec bytů o náhodné délce mezi 1 až 2000 bytů. Cílem těchto testů bylo zjistit, zda si automatizovaná detekce poradí s jakkoli velkým tunelovacím protokolem. Jednotlivé testy jsou znázorněny na obrázku 6.9.

Posloupnost protokolů	Délka náhodného řetězce X
IPv6-TCP-X-IPv4-UDP	1 až 2000 bytů
IPv6-TCP-X-IPv4-TCP	1 až 2000 bytů
IPv6-TCP-X-IPV6-UDP	1 až 2000 bytů
IPv6-TCP-X-IPv6-TCP	1 až 2000 bytů

Obrázek 6.9: Přehled zkoumaných rámců obsahujících náhodný řetězec

Během testování nebyl nalezen případ, ve kterém by automatizovaná detekce selhala. Pokud by při automatizované detekci přeci jen nastala chyba, není problém špatný paket odlišit. Pravděpodobně se budou lišit adresy a porty protokolů.

Kapitola 7

Závěr

Samotným implementacím předchází analýza síťové komunikace. Analyzovány jsou vrstvové modely ISO/OSI a TCP/IP. Detailně jsou popsány protokoly IPv4, IPv6, TCP a UDP. Znalost těchto protokolů a jejich hlaviček je stěžejní pro samotnou realizaci automatizované detekce transportního protokolu. S využitím znalostí o TCP/IP protokolech jsou rozebrány tunelovací protokoly 6in4, GRE a Teredo.

Pro ověření správné funkčnosti automatizované detekce byla nejprve implementována knihovna, která je schopna zpracovat tunelovací protokoly 6in4 a GRE klasickou metodou. Klasická metoda postupně zpracuje hlavičky protokolů až do chvíle, kdy narazí na neznámý protokol nebo na protokol, který neobsahuje informace o následující hlavičce. Implementaci tunelovacího protokolu Teredo nebylo možné v této knihovně realizovat, protože protokol Teredo využívá pro přenos IPv6 protokolu UDP protokol, který neobsahuje informace o typu přenášených dat. Na zpracování transportních protokolů přenášených v určitých typech tunelové komunikace, například v tunelovacím protokolu Teredo, je nutné využít automatizovanou detekci.

Automatizovaná detekce využívá ke své činnosti signatury TCP/IP protokolů. V rámci této práce byly nalezeny signatury protokolů IPv4, IPv6, TCP a UDP. Signaturami se rozumí vlastnosti, které dané protokoly splňují. Jedná se například o pozici hlavičky protokolu, nebo hodnoty jednotlivých polí v ní.

Na základě nalezených signatur byla implementována knihovna schopná automatizované detekce transportního protokolu v tunelovém provozu bez znalosti tunelovacího protokolu. Tato knihovna prochází byte po bytu blok dat, který není klasická metoda schopná zpracovat, a snaží se pomocí signatur nalézt místo, které může odpovídat hledaným protokolům. Pokud je takové místo objeveno, je vypočítán kontrolní součet. Při kladném výsledku je blok dat prohlášen za transportní protokol a je předán klasickým metodám ke zpracování. Automatizovaná metoda je založena pouze na signaturách IP a transportních protokolů. Z tohoto důvodu dokáže nalézt transportní protokol kdekoli v procházených datech. Nezáleží přitom na typu protokolů, ve kterých je transportní protokol zapouzdřen. Díky této vlastnosti dokáže automatizovaná detekce rozpoznat transportní protokol i v komunikaci, která obsahuje vícenásobné zapouzdření.

Výkonnost automatizované detekce byla porovnána s výkonností klasického rozpoznání. K tomuto účelu byla vytvořena sada testů. Výsledky ukázaly, že je automatizovaná metoda při zpracování velkého množství dat přenášených v jednom rámci znatelně pomalejší než klasické zpracování. Při zpracování rámců do 1 kB je výkonnost obou metod srovnatelná. Další sada testů měla za úkol otestovat účinnost automatizované detekce. Tato sada testů se zaměřovala na přenos souboru v transportních protokolech v různých tunelovacích

protokolech. Data z detekovaných transportních protokolů byla následně porovnána s originálními daty. Během testování se nevyskytl ani jediný případ, ve kterém by automatizovaná detekce selhala. Z výsledků testů vyplývá, že automatizovanou detekci lze úspěšně nasadit na rozpoznání TCP/IP protokolů v jakýchkoliv tunelovacích protokolech.

Literatura

[1] POSTEL, J., *Internet Protocol.* RFC 791. [online].
September 1981. [cit. 2015-01-23]
Dostupné z: http://tools.ietf.org/html/rfc791

[2] DEERING, S., *Internet Protocol, Version 6 (IPv6).* RFC 2460. [online].
December 1998. [cit. 2015-01-23]
Dostupné z: https://tools.ietf.org/html/rfc2460

[3] POSTEL, J., *Internet Control Message Protocol.* RFC 792. [online].
September 1981. [cit. 2015-01-23]
Dostupné z: https://tools.ietf.org/html/rfc792

[4] CONTA, A., *Internet Control Message Protocol (ICMPv6) for the Internet Protocol Version 6 (IPv6) Specification.* RFC 4443. [online].
March 2006. [cit. 2015-01-23]
Dostupné z: https://tools.ietf.org/html/rfc4443

[5] POSTEL, J., *Assigned Numbers.* RFC 790. [online].
September 1994. [cit. 2015-01-21]
Dostupné z: http://tools.ietf.org/html/rfc790

[6] KRISHNAN, S., *A Uniform Format for IPv6 Extension Headers.* RFC 6564. [online].
December 1998. [cit. 2015-05-05]
Dostupné z: https://tools.ietf.org/html/rfc6564

[7] POSTEJ, J., *Transmission Control Protocol.* RFC 793. [online].
September 1981. [cit. 2015-01-24]
Dostupné z: https://tools.ietf.org/html/rfc793

[8] POSTEJ, J., *User Datagram Protocol.* RFC 768. [online].
August 1980. [cit. 2015-01-24]
Dostupné z: https://www.ietf.org/rfc/rfc768

[9] HANKS, S., *Generic Routing Encapsulation (GRE).* RFC 1701. [online].
October 1994. [cit. 2015-01-24]
Dostupné z: http://tools.ietf.org/html/rfc1701

[10] NORDMARK, E., *Basic Transition Mechanisms for IPv6 Hosts and Routers.* RFC 4213. [online].
October 2005. [cit. 2015.01.25]
Dostupné z: http://tools.ietf.org/html/rfc4213

[11] HUITEMA, C., *Teredo: Tunneling IPv6 over UDP through Network Address Translations (NATs).* RFC 4382. [online].
February 2006. [cit. 2015.01.25]
Dostupné z: http://www.ietf.org/rfc/rfc4380

[12] MATOUSEK, P., *Síťové aplikace a jejich architektura.*
VUTIUM, Brno, 2014

[13] *IEEE list of EtherType values.* [online].
May 2015. [cit. 2015.05.03]
Dostupné z: http://standards.ieee.org/develop/regauth/ethertype/eth.txt