VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
DEPARTMENT OF COMPUTER SYSTEMS

POTLAČENÍ DDOS ÚTOKŮ ZNEUŽÍVAJÍCÍCH FRAGMENTACI PAKETŮ

MITIGATION OF DDOS ATTACKS EXPLOITING PACKET FRAGMENTATION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **PATRIK NEŠPOR**
AUTHOR

VEDOUCÍ PRÁCE **Ing. JAN KUČERA**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačových systémů (UPSY) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Nešpor Patrik**
Program: Informační technologie
Název: **Potlačení DDoS útoků zneužívajících fragmentaci paketů**
Mitigation of DDoS Attacks Exploiting Packet Fragmentation
Kategorie: Počítačové sítě

Zadání:

1. Seznamte se s problematikou útoků typu odepření služby a zařízením vyvíjeným v rámci sdružení CESNET pro ochranu před těmito útoky.
2. Nastudujte problematiku útoků zneužívajících fragmentaci paketů a možnosti jejich potlačení.
3. Navrhněte přístup, který pro potlačení DDoS útoku kombinuje limitování provozu a existující heuristiku top-n. V rámci návrhu uvažujte i heuristickou metodu pro odhad čísel portů u fragmentovaných paketů.
4. Navržený přístup implementujte.
5. Vyhodnoťte implementované řešení z hlediska dosažených vlastností.
6. V závěru diskutujte výsledky a možnosti dalšího pokračování práce.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění bodů 1 až 3 zadání.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Kučera Jan, Ing.**
Konzultant: Krobot Pavel, Ing., CESNET
Vedoucí ústavu: Sekanina Lukáš, prof. Ing., Ph.D.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 28. května 2020
Datum schválení: 25. října 2019

Abstrakt

Bakalářská práce se zabývá vývojem nových mitigačních strategií, které mají za úkol potlačit DDoS útoky zneužívající fragmentaci paketů. Analyzuje fragmentovaný provoz, přičemž klade důraz především na pořadí fragmentovaných paketů. Na základě výsledků z analýzy implementuje novou heuristickou metodu pro odvození čísel portů. Součástí práce je také měření latence a propustnosti nově implementovaných funkcionalit. Nové strategie jsou společně s heuristickou metodou zaintegrovány v projektu, jenž je vyvíjen sdružením CESNET.

Abstract

This bachelor's thesis deals with the development of new mitigation strategies that are designed to suppress DDoS attacks that exploit packet fragmentation. It analyses fragmented traffic primarily emphasizing the order of fragmented packets. Based on the result of the analysis, it implements a new heuristic method for port numbers inference. The work also measures the latency and throughput of newly implemented functionalities. All new methods are integrated in a project developed by the CESNET association.

Klíčová slova

Fragmentované pakety, Zneužití fragmentovaných paketů, Mitigační strategie, Odvození čísel portů, DDoS, DDoS mitigace

Keywords

Packet fragmentation, Exploiting packet fragmentation, Mitigation strategies, Ports inference, DDoS, DDoS mitigation

Citace

NEŠPOR, Patrik. *Potlačení DDoS útoků zneužívajících fragmentaci paketů.* Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Jan Kučera

Potlačení DDoS útoků zneužívajících fragmentaci paketů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Jana Kučery. Další informace mi poskytl pan Ing. Pavel Krobot ze sdružení CESNET. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Patrik Nešpor
27. května 2020

Poděkování

Rád bych poděkoval mému vedoucímu práce panu Ing. Janu Kučerovi za pevné nervy při konzultacích, ochotu a vůbec za umožnění pracovat na tomto tématu. Dále bych chtěl poděkovat panu Ing. Pavlu Krobotovi za to, že se mě ujal ihned po mém nástupu ke sdružení CESNET, vždy mě navedl správnou cestou a také za velmi přátelský přístup. V neposlední řadě bych rád poděkoval mé nejbližší rodině, která mi umožnila studovat a byla mi vždy oporou.

Obsah

Kapitola 1

Úvod

Distributed denial-of-service (DDoS) útoky představují v novodobých počítačových sítích stále rostoucí hrozbu. Hlavním cílem DDoS útoků je zamezit zpracování požadavků všem legitimním uživatelům, kteří se snaží komunikovat s napadeným síťovým zařízením. Z důvodu neustálé snahy útočníků modifikovat a zdokonalovat útočné metody, dosáhla problematika DDoS bodu, kdy je velmi obtížné počítat s veškerými riziky přicházejícího útoku.

V současnosti existuje široká škála DDoS útoků. Část z nich zneužívá ke svému cíli fragmentované pakety. Koncept fragmentovaných paketů trpí řadou nedostatků, kterých může potencionální útočník snadno zneužít. Typickou strategií útočníka je generování falešných fragmentovaných paketů, které nelze defragmentovat, a přijaté fragmenty tak po určitou dobu zabírají paměť oběti. Následkem útoku může dojít k úplnému vyčerpání prostředků cílového zařízení, tudíž legitimní uživatelé nadále nemohou využívat služeb síťového prvku.

Cílem bakalářské práce je potlačení DDoS útoků zneužívajících fragmentaci paketů. Obrana proti fragmentovaným útokům je řešena implementací nových mitigačních strategií vhodných pro fragmentovaný provoz. Dalším krokem k ochraně proti fragmentovaným útokům je vytvoření heuristické metody zvané Ports Inference modul. Tato metoda slouží k přiřazení chybějících čísel zdrojového a cílového portu. Čísla portů jsou obsažena vždy pouze v prvním fragmentu datagramu. Další fragmenty čísla již neobsahují. V ideálním případě je ale možné čísla portů u všech následujících fragmentů s využitím heuristiky odvodit. Smyslem odvození čísel portů fragmentovaných paketů je především možnost filtrace fragmentovaného provozu a definice pravidel specifikující rozsah portů. Nové mitigační strategie jsou společně s heuristickou metodou pro odvození čísel portů zaintegrovány v projektu, který je vyvíjen v rámci sdružení CESNET.

Práce je členěna do logických celků. V kapitole 2 je krátce popsána síťová architektura na referenčním modelu TCP/IP. Dále je v kapitole blíže představena problematika fragmentovaného provozu, včetně fragmentace v protokolu IPv6 a shrnutí problémů vznikajících s fragmentací. Následující kapitola 3 pojednává o fragmentovaném provozu v DDoS útocích a možných způsobech ochrany. V kapitole 4 je představeno zařízení vyvíjené sdružením CESNET pro ochranu proti DDoS útokům. Kapitola se také zmiňuje o hlavních detekčních modulech, které zařízení využívá. Kapitola 5 se zabývá návrhem a implementací nových funkcionalit vytvářených v rámci této bakalářské práce a analýzou fragmentovaného provozu. V poslední kapitole 6, před samotným závěrem, je měřením otestována latence a propustnost nových strategií a Ports Inference modulu. Rekapitulaci dosažených výsledků a možným pokračováním v budoucnu se zabývá kapitola 7.

Kapitola 2

Principy počítačových sítí

Kapitola se věnuje teoretickým znalostem, jež je nutné zmínit pro přiblížení problematiky zneužití fragmentovaného provozu v DDoS útocích. Teoretická část bude základem pro pochopení praktického úseku bakalářské práce. V úvodní části kapitoly si popíšeme síťovou architekturu na referenčním modelu TCP/IP. Následuje popis zaměřený na fragmentaci provozu, kde si ukážeme, jak je fragmentace a defragmentace prováděna. Mimo jiné také zjistíme, jaké položky IP hlavičky jsou věnovány právě fragmentaci a jaká úskalí s sebou fragmentace přináší. Závěr kapitoly pojednává o způsobech, jimiž lze fragmentaci předcházet.

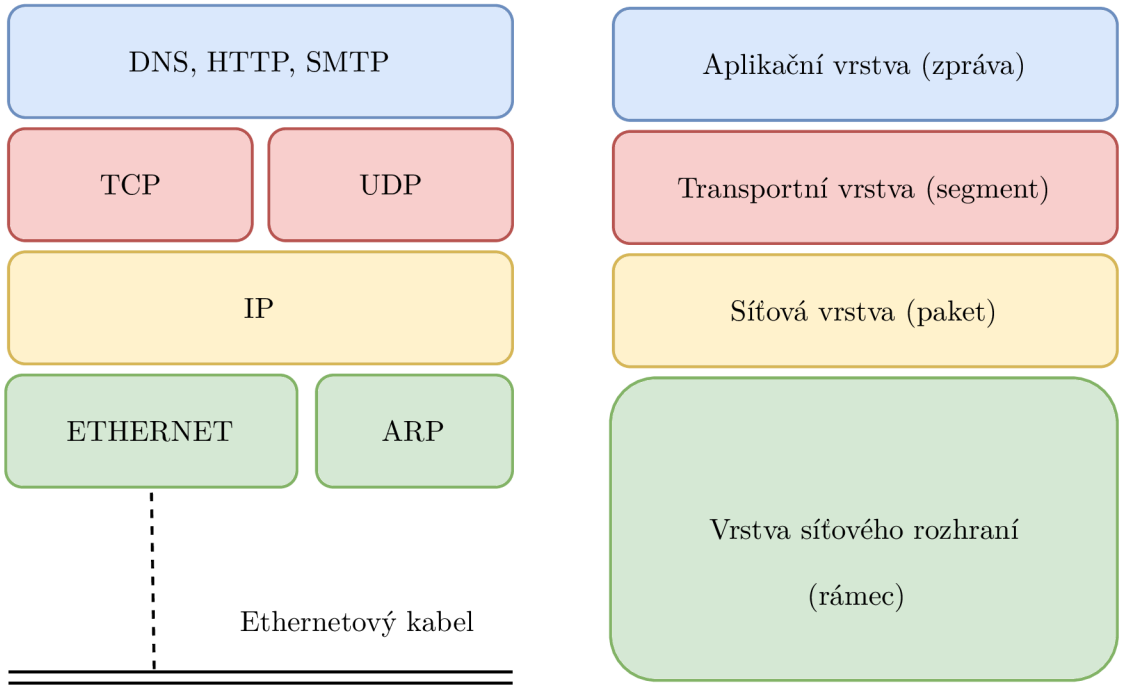
2.1 Síťová architektura

Internet je počítačová síť, která propojuje miliardy výpočetních zařízení skrze celý svět [20]. Jedním z klíčových faktorů, stojících za úspěchem Internetu, je jeho samotná architektura, která je navržena tak, aby dokázala reagovat na rychlé a podstatné změny. Internet původně pochází ze sítě zvané „Arpanet“ z pozdějších 60. let 20. století. Tato síť spojovala univerzity a vládní instituce za účelem výzkumu a umožňovala mezi nimi vzdálený přístup k datům a spouštění programů. World Wide Web komercializoval Internet v 90. letech 20. století a zpřístupnil jej mimo univerzity a vládní instituce. Dnes je Internet nejdůležitějším komunikačním médiem (nepočítaje telefonní síť), obsluhujícím několik miliard uživatelů [23].

V počítačové vědě je často z důvodu snazšího představení a porozumění komplexním síťovým interakcím využíváno konceptu rozdělení na síťové vrstvy. Existují dva nejrozšířenější referenční modely – OSI (Open System Interconnect) a TCP/IP. Architektury obou zmíněných referenčních modelů jsou podobné, avšak jejich jednotlivé vrstvy se liší [7]. Vzhledem k tomu, že TCP/IP model oproti OSI modelu popisuje Internet mnohem více tak, jak jej dnes známe, bude dále princip síťové architektury vysvětlen právě na referenčním modelu TCP/IP. Internet je založen na dvojici základních síťových protokolů. Počítače jsou v Internetu spojeny sítí (pomocí TCP protokolu – Transmision Control Protocol), za účelem výměny dat obsažených v paketech (použitím IP protokolu – Internet Protocol) [21].

Na levé straně obrázku 2.1 je vykreslena logická struktura protokolů rozdělených do vrstev uvnitř počítače připojeného k Internetu. Každý počítač komunikující za pomocí internetových technologií využívá podobné logické struktury, která stojí za rozhodováním v chování počítače na Internetu [29].

Na pravé straně obrázku 2.1 je k vidění referenční TCP/IP model. Referenční model je umístěn tak, aby jeho jednotlivé vrstvy (stručně popsány za obrázkem) odpovídaly logické struktuře nalevo. Mimo jiné je u každé vrstvy uveden také název datové jednotky zvané PDU (Process Data Unit).



(a) Základní TCP/IP síťový uzel.

(b) Vrstvy TCP/IP modelu, s názvy PDU.

Obrázek 2.1: Srovnání TCP/IP síťového uzlu s referenčním TCP/IP modelem.

Aplikační vrstva. Aplikační vrstva je na samém vrcholu TCP/IP modelu. Definuje, jak jednotlivé služby pracují a jakým způsobem mohou být tyto služby použity. Aplikační vrstva obsahuje všechny vysokoúrovňové protokoly, jako jsou DNS (Domain Name System), HTTP (Hypertext Transfer Protocol), SSH (Secure Shell) apod.

Transportní vrstva. Pro přenos uživatelských dat se nejčastěji využívá dvou transportních protokolů. Prvním je TCP (Transmission Control Protocol) a druhým UDP (User Datagram Protocol). Oba protokoly rozloží zprávu, kterou chce aplikace odeslat do jednotlivých paketů, jež jsou následně odeslány požadovanému příjemci. Na straně příjemce mají protokoly transportní vrstvy za úkol z přijatých paketů předat payload[1].

Nejvýznamnějším rozdílem mezi protokoly je jejich spolehlivost. Zatímco TCP protokol je spolehlivý, UDP nikoliv. TCP protokol shromáždí veškeré přijaté pakety a uspořádá je do správného pořadí, čímž umožňuje sestavení původní, originální zprávy. V případě potřeby si TCP vyžádá opětovné odeslání ztracených nebo poškozených paketů. UDP pouze přijme příchozí paket a předá jeho payload vyšší aplikační vrstvě. O veškeré chyby nebo špatné uspořádání dat se musí dále postarat samotná aplikace.

[1]Payload – V počítačových sítích a telekomunikaci obsahuje přenosová jednotka odeslaná od zdroje k cíli hlavičku a skutečná data, která se mají přenést. Zmiňovaná přenášená data se nazývají payload [9].

UDP je díky své nespolehlivosti rychlejší, a proto se využívá u aplikací pro streamování videa a audia, kde je občasná chyba či ztráta přijatelnější nežli pomalý, ale spolehlivý TCP přenos. Obecně by se dalo říci, že je UDP protokol navržen pro aplikace, které striktně nevyžadují, aby byly přijaté pakety v původním pořadí. Z tohoto důvodu bývá UDP považován za „bezespojový“ (anglicky connection-less) protokol. TCP protokol je anglicky označován jako „connection-oriented“ protokol. TCP spojení je identifikováno IP adresami a čísly portů, které využívají obě koncové stanice. Datový tok je u TCP spojení seskupován do jednotlivých segmentů. Pořadí segmentů je určováno sekvenčními čísly. Proces ustanovení TCP spojení má za úkol zjistit, zda obě strany komunikace skutečně existují, výměnu volitelných parametrů (velikost okna, velikost paketu, ...), alokaci transportních zdrojů apod. [32]

Síťová vrstva. Vrstva zodpovědná za přenos a směrování dat v síti. Internet používá IP (Internet Protokol) jako síťovou vrstvu. Každý uzel má svou adresu, nazývanou IP adresa, a data jsou odesílána jako IP pakety. Princip fungování IP protokolu je jednoduchý: paket obsahuje zdrojovou adresu, cílovou adresu a payload a je odesílán z jednoho uzlu v síti na další, dokud nedosáhne svého cíle. Nastane-li situace, kdy se paket po cestě ztratí, IP protokol ztrátu neodhalí a paket nikdy nedosáhne cíle. Jestliže uzel není schopen přeposlat paket na další uzel normální cestou, vynasnaží se najít alternativní cestu k cíli. Proto je IP protokol označován jako „best-effort delivery“ protokol. První verzí IP byla IPv4, nasazená roku 1983. Dodnes je IPv4 nejpoužívanější IP verzí. Novější verzí IP protokolu je IPv6. Hlavní motivací pro vytvoření IPv6 byl nedostatek IP adres v IPv4 [22].

Vrstva síťového rozhraní. Definuje protokoly a hardware potřebný k připojení hosta do fyzické sítě a doručení dat skrze tuto síť. Pakety jsou ze síťové vrstvy odeslány níže do vrstvy síťového rozhraní a následně do fyzické sítě. Cílová destinace může být jiný host v síti, totožný host nebo směrovač zodpovědný za další směrování paketu [1]. Nejznámějšími protokoly, které v TCP/IP modelu spadají pod vrstvu síťového rozhraní, jsou Ethernet a WiFi.

2.2 Fragmentace provozu

Ne všechny protokoly ve vrstvě síťového rozhraní jsou schopny přenášet pakety o stejných velikostech. Některé z protokolů dokaží přenést datagramy relativně velké, jiné zase poměrně malé. Například ethernetové rámce mohou obsahovat data o velikosti až 1500 bajtů, zatímco rámce využívané u rozlehlých síťových linek (protokol X.25) nepřenesou více než 576 bajtů [34]. Zajímavostí je, že ethernetové rámce jsou schopny dosahovat mnohem větších velikostí než standardních 1500 bajtů. Rámce se nazývají „Jumbo Frame“ a využívají se v lokálních sítích s přenosovou rychlostí alespoň 1 Gb/s. Velikost takových rámců může dosahovat i 9000 bajtů (jelikož rámce nejsou definovány v IEEE 802.3 pro Ethernet, velikosti se mohou lišit) [28]. Maximální velikost dat, kterou je možno vyslat síťovým rozhraním, se nazývá maximální přenosová jednotka (anglicky Maximum Transmission Unit, MTU). Seznam vybraných linek a jejich velikostí MTU znázorňuje tabulka 2.1. Jelikož je každý IP datagram zapouzdřený v rámci, v němž je dále vysílán z jednoho směrovače na druhý, určuje MTU pevné ohraničení velikosti IP datagramu. Samotné omezení velikosti IP datagramu by však nepředstavovalo problém. Problém nastává ve chvíli, kdy linky, spojující směrovače mezi příjemcem a odesílatelem, využívají odlišné protokoly linkové vrstvy a tím pádem i odlišnou velikost MTU.

Velikost MTU	Typ linky
65535	PPP maximum
1500	Ethernet
1480	PPPoE
1460	L2TP
576	X.25

Tabulka 2.1: Ukázkový seznam linek a jejich MTU velikostí.

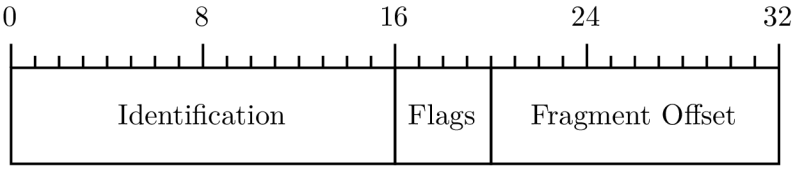
Jak je vyřešen problém odlišných protokolů, včetně jejich velikostí MTU vyskytujících se mezi příjemcem a odesílatelem IP datagramu? Řešení spočívá v rozdělení dat IP datagramu na dva nebo více menších datagramů, zaobalení takto rozdělených datagramů do linkového rámce a následné odeslání odchozí linkou. Datagramy, které vznikly rozložením jednoho celkového datagramu, se nazývají fragmenty.

Příchozí fragmenty je zapotřebí znovu sestavit do jednoho celku a vytvořit tak původní datagram ještě předtím, než dosáhnou transportní vrstvy na straně příjemce, jelikož transportní protokoly UDP a TCP počítají s kompletními, nerozfragmentovanými datagramy z nižší síťové vrstvy. Z důvodu zachování jednoduchosti síťového jádra se vývojáři IPv4 rozhodli přenechat práci defragmentace datagramu raději koncovému zařízení než síťovým směrovačům.

Poté, co cílový host obdrží řadu datagramů ze stejného zdroje, musí ověřit, zda některý z datagramů není ve skutečnosti fragmentem a není tedy součástí jednoho většího datagramu. V případě, že je obdržený datagram skutečně fragmentem, musí příjemce po obdržení posledního chybějícího fragmentu sestavit původní datagram [20]. Není-li uvedeno jinak, získané informace, týkající se fragmentace paketů, jsou získany ze zdrojů [26], [29], [31] a [20].

2.2.1 Přehled fragmentace v protokolu IP

Proces fragmentace a následné defragmentace musí být implementován tak, aby byl z fragmentů na straně odesílatele složen opět původní datagram na straně příjemce. Podrobnější popis fragmentace a defragmentace, včetně ukázkové implementace, se nachází v kapitolách 2.2.2 a 2.2.3. Pro zajištění korektního procesu designéři IPv4 vložili několik položek do IP hlavičky (znázorněno na obrázku 2.2):

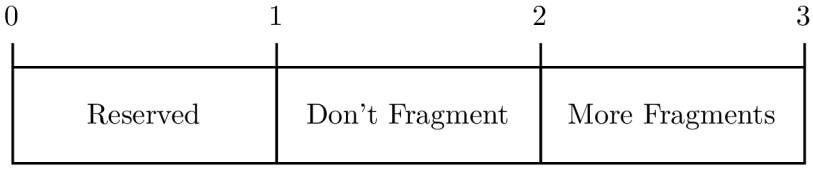


Obrázek 2.2: Část IP hlavičky pro účely fragmentace.

- **Identification** – Příjemce fragmentu využívá položky Identification z IP hlavičky k tomu, aby rozeznal, zda se jedná o fragment patřící k požadovanému datagramu a případně se tak nemísily fragmenty jednoho datagramu s fragmenty patřícími k jinému datagramu. Hodnota položky Identification musí být unikátní vzhledem ke zdrojové a cílové IP adrese a protokolu. To znamená, že se nesmí opakovat během maximální „životnosti" datagramu (anglicky Maximum Datagram Lifetime, MDL).

Typickou hodnotou pro MDL jsou 2 minuty a souvisí s časovým úsekem doporučeným pro vypršení (anglicky timeout) defragmentace [31].

- **Flags** – Sestává ze tří bitových položek, které jsou k vidění na obrázku 2.3:



Obrázek 2.3: Jednotlivé položky Flags v IP hlavičce.

  - První ze tří bitů je rezervován pro budoucí účely a nabývá vždy hodnoty nula.
  - Je-li nastaven **Don't Fragment** (DF) bit, není možné za žádných okolností na datagramu provádět proces fragmentace. To znamená, že v případě, kdy je zapotřebí rozdělit datagram na menší části, aby byl schopen dorazit do cílového zařízení, je tento datagram zahozen a přenos není možné vykonat. Naopak hodnota nula fragmentaci povoluje.
  - **More Fragments** (MF) bit je nastaven, jestliže se nejedná o poslední fragment. Vynulovaný bit označuje u fragmentovaných paketů poslední fragment v pořadí.

- **Fragment Offset** – Číselná hodnota měřená v osmibajtových jednotkách, udává příjemci pozici fragmentu v původním datagramu. Fragment Offset a délka získaná z hlavičky společně určují, jakou část původního datagramu daný fragment zabírá. První fragment nabývá vždy hodnoty Fragment Offset nula.

  Tabulka 2.2 znázorňuje veškeré kombinace bitů Fragment Offset a More Fragments v závislosti na typu fragmentu.

	More fragments	**Fragment Offset**
Nefragmentovaný paket	0	0
První fragment	1	0
Prostřední fragmnet	1	Velikost předchozích fragmentů bez hlavičky, měřených v osmi bajtech
Poslední fragment	0	

Tabulka 2.2: Kombinace bitů pro různé typy fragmentů.

Shrňme si podstatné informace, které jsme doposud zjistili, abychom mohli postoupit k ukázkovému příkladu procedury fragmentace a defragmentace. Položka hlavičky Identification je použita společně se zdrojovou a cílovou IP adresou a číslem protokolu k jednoznačné identifikaci fragmentu v rámci datagramu. Flag More Fragments je nastaven v momentě, kdy je paket fragmentován a jsou očekávány další fragmenty (tj. nejedná se o poslední fragment). Fragment Offset udává pozici fragmentu vztahující se k původnímu nerozfragmentovanému datagramu, přičemž jsou fragmenty počítány v osmibajtových jednotkách. Strategie fragmentace je navržena tak, aby nefragmentovaný datagram obsahoval u položek určených k řízení fragmentace hodnotu nula (MF = 0, FO = 0). Jestliže je IP datagram fragmentován, musí být jeho data zarovnána na osmibajtové hodnoty. Tento formát umožňuje

$$2^{13} = 8192$$

osmibajtových fragmentů, což ve výsledku znamená 65 536 bajtů. Výsledná hodnota je tedy rovna maximální velikosti datagramu.

Dle [26] musí být síťové zařízení schopno přenést datagram o velikosti 68 bajtů bez jakékoliv fragmentace. Toto opatření vyplývá z faktu, že internetová hlavička může nabývat až 60 bajtů a minimální velikost fragmentu je 8 bajtů. Dále udává, že každá cílová stanice musí přijmout datagram o velikosti alespoň 576 bajtů, ať už v podobě jednoho datagramu, či několika fragmentů.

Fragmentací mohou být ovlivněny následující položky IP hlavičky:

1. Options – její velikost je proměnná, závisí na počtu přidaných položek,
2. More Fragments flag,
3. Fragment Offset,
4. Internet Header Length – čtyřbitová položka udávající velikost internetové hlavičky,
5. Total Length – dvoubajtová položka reprezentující velikost paketu (IP hlavička + payload),
6. Header Checksum – hodnota vypočítaná na základě položek IP hlavičky (slouží pro ověření bezchybovosti přenosu).

Je-li nastaven Don't Fragment (DF) bit, není povoleno daný datagram fragmentovat, i za cenu toho, že bude celý datagram zahozen. Takové opatření je využíváno například v případě, kdy příjemce nemá dostatečné prostředky k tomu, aby mohl provádět defragmentaci.

2.2.2 Proces fragmentace

Následuje textový popis algoritmu 1, který symbolicky vystihuje proces fragmentace IP datagramu. V textovém popisu i v algoritmu se vyskytují zkratky blíže popsané v tabulce 2.3

Jestliže je celková délka TL menší nebo rovna MTU, není potřeba provádět proces fragmentace a datagram je možné dále zpracovat (řádek 1). V opačném případě se ošetří hodnota DF (Don't Fragment), je-li nastavena, dojde k zahození datagramu (řádek 2). Není-li DF nastaven, musí být datagram rozdělen na dva fragmenty. První fragment, který bude nabývat maximální možné velikosti, a druhý fragment o velikosti zbytku datagramu. U prvního fragmentu dojde ke zkopírování internetové hlavičky (řádek 6) a následně se do proměnných s předchozími hodnotami uloží hodnoty aktuální (řádky 7, 8 a 9). Po přiřazení hodnot proměnným se prvnímu fragmentu přiřadí povolené množství dat (řádek 10), nastaví se MF informující o tom, že budou následovat další fragmenty z datagramu a vypočítá se hodnota celkové délky TL součtem velikostí hlavičky a dat (řádek 11). Zbývá přepočítat Checksum hodnotu (řádek 12) a předat fragment k dalšímu zpracování.

U druhého fragmentu se selektivně zkopírují položky hlavičky (některé hodnoty z položky Options zkopírovány nejsou). Fragmentu se přidělí veškerá zbývající data (řádek 15). Odečtením nezkopírovaných položek od OIHL se vypočítá aktuální velikost internetové hlavičky IHL (řádek 16). Dojde k odvození délky TL odečtením velikosti dat prvního fragmentu od OTL společně s velikostí hlavičky (řádek 17). Posledním přiřazením dojde k úpravě nové FO a MF hodnoty (řádek 18). Opět se přepočítá Checksum fragmentu k ověření správnosti přenosu (řádek 19). V případě, že je výsledný fragment stále příliš velký, opakuje se celá procedura do té doby, než poslední fragment splní požadovanou velikost MTU.

Zkratka	Význam
FO	Fragment Offset
IHL	Velikost internetové hlavičky
DF	Don't Fragment
MF	More Fragments
TL	Celková délka
OFO	Předchozí hodnota Fragment Offset
OIHL	Předchozi velikost internetové hlavičky
OMF	Předchozí hodnota More Fragments
OTL	Předchozí celková velikost
NFB	Počet bloků fragmentu
MTU	Maximální přenosová jednotka

Tabulka 2.3: Zkratky využívané v proceduře fragmentace a jejich význam.

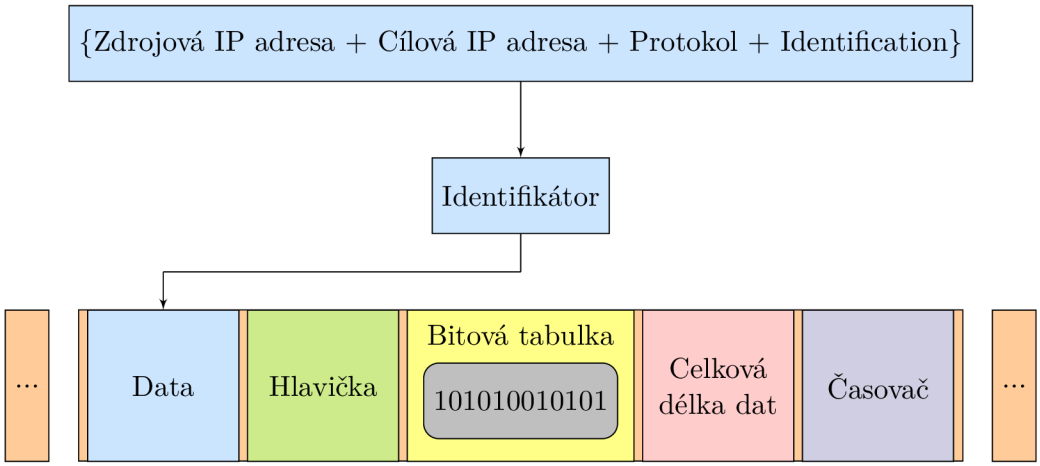
Algorithm 1: Procedura fragmentace.

```
1  if TL <= MTU then
2  |  Předej daný datagram k dalšímu zpracování;
3  else if DF == 1 then
4  |  Zahoď datagram;
5  else
6  |  Zkopíruj původní internetovou hlavičku;
7  |  OIHL ← IHL; OTL ← TL;
8  |  OFO ← FO; OMF ← MF;
9  |  NFB ← (MTU − IHL ∗ 4)/8;
10 |  Připoj prvních NFB ∗ 8 bajtů dat;
11 |  MF ← 1; TL ← (IHL ∗ 4) + (NFB ∗ 8);
12 |  Přepočítej Checksum hodnotu;
13 |  Předej fragment k dalšímu zpracování;
14 |  Zkopíruj potřebné položky z internetové hlavičky;
15 |  Připoj zbývající data;
16 |  IHL ← (((OIHL ∗ 4) − (délka nezkopírovaných položek)) + 3)/4;
17 |  TL ← OTL − NFB ∗ 8 − (OIHL − IHL) ∗ 4);
18 |  FO ← OFO + NFB; MF ← OMF;
19 |  Přepočítej Checksum;
20 |  Předej fragment k otestování; Konec procedury;
```

2.2.3 Proces defragmentace

Stejně jako v podkapitole 2.2.2 bude následovat textový popis defragmentace, popisující algoritmus 2. Zkratky využívané v procesu defragmentace jsou vysvětleny v tabulce 2.4.

Na první pohled může být proces defragmentace a s tím související alokace paměti poněkud matoucí. Pro přibližnou představu o alokované paměti pro jeden datagram slouží obrázek 2.4. Vrchní část obrázku znázorňuje klíčové položky, které společně vytváří unikátní identifikátor pro každý datagram. Identifikátor poté odkazuje na alokovanou paměť datagramu určenou k defragmentaci. Na obou stranách od alokované paměti je oranžovými bloky znároněno další alokované místo patřící jiným datagramům.



Obrázek 2.4: Znázornění alokované paměti pro účely defragmentace.

Pro každý datagram je alokována samostatná paměť. O tom, kterému datagramu náleží alokovaná paměť, rozhoduje identifikátor. Identifikátor je vytvořen spojením zdrojové a cílové IP adresy, protokolu a položky Identification (řádek 1). Jsou-li bity datagramu More Framgents a Fragment Offset rovny nule (řádek 2), jedná se o kompletní datagram a není třeba dále řešit defragmentaci. Pokud je potřeba, proběhne uvolnění veškeré paměti přiřazené k defragmentaci tohoto datagramu (řádek 4). Daný datagram tak může být předán k dalšímu zpracování (řádek 5). Jestliže zatím nebyl zpracován žádný fragment se stejným identifikátorem (řádek 7), je zapotřebí alokovat paměť pro celý datagram (řádek 8). Alokovaná paměť se skládá z části pro data, hlavičku, tabulky obsahující bity bloků jednotlivých fragmentů, bloku celkové délky dat a časovače. Proběhne nastavení časovače na nejnižší možnou hodnotu (řádek 9) a vynuluje se délka dat TDL (řádek 10). Data fragmentu jsou umístěna do části paměti s daty, a to vzhledem k offsetu datagramu a jeho délce (řádek 11). Kromě části s daty se také nastaví jednotlivé bity v bitové tabulce, které odpovídají offsetu daného fragmentu (řádek 12). Nastavením bitů určíme, jakou část původního datagramu jsme již přijali.

Pokud se jedná o poslední fragment (hodnota More Fragments se rovná nule, řádek 13), je vypočítána celková délka dat TDL (řádek 14). Jedná-li se o první fragment (takový fragment, který má hodnotu Fragment Offset rovnu nule, řádek 15), vloží se hlavička tohoto fragmentu na místo v paměti reprezentující data hlavičky (řádek 16). V případě, že je doručený fragment poslední chybějící částí datagramu (zjistí se testováním nastavených bitů v tabulce bloků fragmentu, řádek 17), přepočítá se nejprve výsledná hodnota TL (řádek 18) a výsledný datagram je přeposlán k dalšímu zpracování (řádek 19). Nesplní-li se ani jedna podmínka, je porovnáním hodnot aktuálního časovače a položky TTL přiřazena větší z nich do aktuálního časovače (řádek 22). Proces defragmentace datagramu je tímto krokem u konce.

Může také nastat případ, kdy vyprší časovač a všechny prostředky přiřazené konkrétnímu identifikátoru jsou uvolněny. Počáteční interval časovače je inicializován na nejnižší možnou hodnotu, a to z toho důvodu, aby se interval navýšil až ve chvíli, kdy je příchozí hodnota TTL obdrženého fragmentu vyšší. Maximální velikost intervalu časovače odpovídá maximální hodnotě TTL (přibližně 4,24 minuty). Doporučená počáteční hodnota časovače je 15 sekund, doba se však může měnit v závislosti na předchozích zkušenostech s protokolem.

Zkratka	Význam
FO	Fragment Offset
IHL	Velikost internetové hlavičky
MF	More Fragments
TTL	Time To Live
NFB	Počet bloků fragmentu
TL	Celková délka
TDL	Celková délka dat
BUFID	Identifikátor paměti
RCVBT	Bitová tabulka přijatých fragmentů
TLB	Nejnižší hodnota časovače

Tabulka 2.4: Zkratky využité v proceduře defragmentace a jejich význam.

Algorithm 2: Procedura defragmentace.

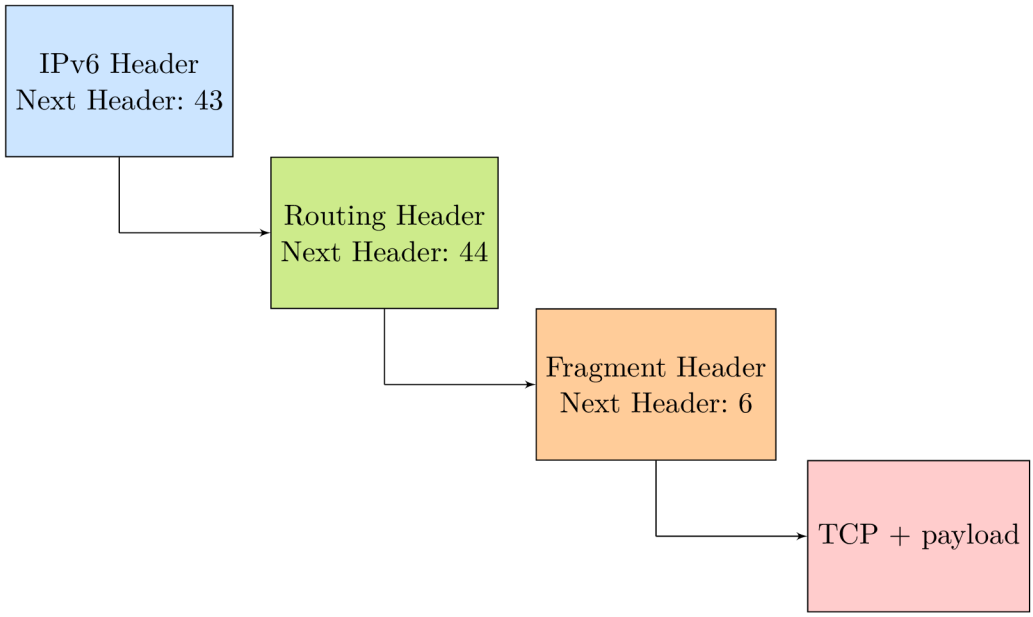
```
1  BUFID ← zdrojová a cílová IP adresa, protokol, identifikátor;
2  if FO == 0  MF == 0 then
3      if paměť s BUFID je alokována then
4          Uvolni veškerou paměť s tímto BUFID;
5          Předej datagram k dalšímu kroku;
6          Konec procedury;
7      else if paměť s BUFID není alokována then
8          Alokuj prostředky potřebné k defragmentaci BUFID;
9          Časovač ← TLB;
10         TDL ← 0;
11         Vlož data z fragmentu do alokované paměti s BUFID od bajtu
            FO * 8 do bajtu (TL − (IHL * 4)) + FO * 8;
12         Nastav RCVBT bity od FO do FO + ((TL − (IHL * 4) + 7)/8);
13         if MF == 0 then
14             TDL ← TL − (IHL * 4) + (FO * 8);
15         if FO == 0 then
16             Vlož hlavičku do alokované paměti na místo hlavičky;
17         if TDL ≠ 0 a jsou nastaveny všechny RCVBT od 0 do (TDL + 7)/8 then
18             TL ← TDL + (IHL * 4);
19             Předej datagram k dalšímu zpracování;
20             Uvolni veškerou paměť pro danou BUFID;
21             Konec procedury;
22         TIMER ← MAX(TIMER, TTL);
23         Vyčkej na další fragment, nebo dokud nevyprší časovač TIMER;
   // Časovač vypršel:
24 Uvolni veškerou paměť pro BUFID;
25 Konec procedury;
```

2.2.4 Fragmentace v protokolu IPv6

Internet Protokol verze 6 (IPv6) je novou verzí internetového protokolu, navrženou jako nástupce IP verze 4. Nový protokol s sebou přinesl řadu změn a vylepšení, ačkoliv pro

nás budou klíčové pouze změny v hlavičce protokolu ovlivňující způsob řešení fragmentace v IPv6.

V IPv6 jsou volitelná data obsahující informace patřící do internetové vrstvy kódována v separátních hlavičkách vložených mezi IPv6 hlavičku a hlavičku vyšší transportní vrstvy. Takových hlaviček rozšiřujících IPv6 hlavičku je pouze omezené množství a každá z nich je identifikována odlišnou, tzv. Next Header hodnotou. Next Header hodnota se vkládá do hlaviček za účelem identifikace následující hlavičky. IPv6 paket může obsahovat několik nebo žádnou z rozšiřujících hlaviček, dle potřeby. Pro účely fragmentace existuje samostatná rozšiřující hlavička, z toho důvodu neobsahuje IPv6 hlavička žádné informace týkající se fragmentace. V obrázku 2.5 je zobrazena situace, kdy IPv6 paket obsahuje nejprve IPv6 hlavičku, poté následují dvě rozšiřující hlavičky. Poslední hlavička paketu patří protokolu vyšší transportní vrstvy.



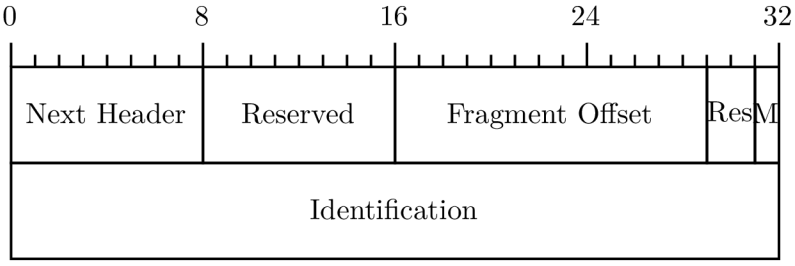
Obrázek 2.5: Příklad provázání IP hlavičky a rozšiřujících hlaviček.

Je-li použito v jednom IPv6 paketu více rozšiřujících hlaviček, RFC2460 [12] určuje bližší doporučení, jak jednotlivé hlavičky uspořádat za sebou.

IPv6 směrovač nově nemůže fragmentovat pakety samostatně, tudíž pokud je paket příliš velký a je zapotřebí jej fragmentovat, směrovač vygeneruje ICMPv6 paket Typu 2, s kódem Packet Too Big (PTB). Vygenerovaný paket je adresován stanici, ze které byl vyslán paket přesahující hodnotu MTU. Zatímco IPv6 směrovač nemá možnost vykonat proces fragmentace, odesílatel paketu ano. Takový způsob řešení přináší značnou nevýhodu oproti předchozí verzi protokolu, kdy stačilo pouze nastavit hodnotu Don't Fragment na nula před odesláním paketu, s čímž odpadala závislost na přijetí ICMP zprávy indikující problémy s MTU. V IPv6 se ale žádný Don't Fragment flag nevyskytuje a jediným způsobem, jak může být fragmentace provedena, je, že odesílatel přijme ICMP PTB zprávu a na oplátku znovu odešle již rozfragmentovaný paket s takovou velikostí, aby odpovídala MTU. Avšak co když jsou všechny ICMP zprávy příjemcem filtrovány? Pak jsou pakety přesahující hodnotu MTU v tichosti zahozeny.

Fragment Header

Fragment Header je využíván ve verzi IPv6 k odeslání paketu většího, než je hodnota MTU příjemce. Jak již bylo zmíněno, na rozdíl od IPv4 probíhá proces fragmentace datagramu přímo ve zdrojové stanici, nikoliv ve směrovači. Fragment Header je identifikován číselnou hodnotou 44 a je uchován v předcházející hlavičce. Položky Fragment Header, viz obrázek 2.6.



Obrázek 2.6: Fragment Header.

- **Next Header** identifikuje typ hlavičky, který bezprostředně následuje za nynější hlavičkou. Využívá stejných hodnot jako položka protokol v IPv4.
- **Reserved** je 8 rezervovaných bitů nabývajících hodnoty nula. Bity jsou u příjemce ignorovány.
- **Fragment Offset** slouží ke stejnému účelu jako v IPv4, tedy udává polohu fragmentu v původním datagramu.
- Dvoubitová položka **Res** značí dva nulové rezervované bity.
- **M flag** je jediným flagem, který zůstal pro účely fragmentace a plní stejnou funkci jako More Fragments flag v IPv4.
- **Identification** se až na dvojnásobnou velikost (32 bitů) oproti předchozí verzi nezměnil.

V nové verzi IPv6 je neustále nastaven Don't Fragment flag a není tak explicitně obsažen v žádné z hlaviček.

2.2.5 Problémy vznikající s fragmentací provozu

Argumenty ve prospěch fragmentace by měly být doposud jasné. Fragmentace umožňuje, aby se protokoly vyšší vrstvy nemusely zajímat o vlastnosti přenosového kanálu, a také zajišťuje odeslání dat ve vhodně zvolené velikosti, jelikož odeslání většího množství dat v každém IP datagramu má za následek minimalizaci režijních nákladů spojených se správou dat.

Fragmentace poskytuje zdrojové stanici způsob, jakým se vypořádat s linkami, které mají odlišné velikosti MTU, bez nutnosti znát cestu, jakou budou pakety směrovány (alespoň tak tomu je u IPv4, kde proces fragmentace probíhá přímo ve směrovačích).

Nevýhody fragmentace se dají podle [17] rozdělit do tří kategorií:

Fragmentace způsobuje neefektivní využití zdrojů

Uvažujeme-li cenu spojenou s každým odeslaným paketem skrze bránu, musíme si uvědomit, že je potřeba také počítat s určitou konstantní výpočetní náročností, spojenou například se směrováním paketu, modifikací hlavičky paketu, výpočtem nové checksum hodnoty nebo zařazením paketu do správné příchozí a odchozí fronty. Tímto způsobem často dochází k neoptimálnímu využití zdrojů směrovače a nadměrnému využití šířky pásma.

Představme si situaci, kdy se zařízení snaží odeslat 1024 bajtů dat napříč směrovačem s velikostí MTU o hodnotě 1007 bajtů, přičemž velikost IP a TCP hlavičky je minimálně 40 bajtů. Celkově tedy potřebujeme odeslat 1064 bajtů velký datagram, který bude směrovačem rozdělen na dva fragmenty, první fragment o velikosti 1000 bajtů (nejbližší hodnota k MTU dělitelná osmi) a druhý fragment velký 84 bajtů. V momentě, kdy budeme chtít odeslat 10 KB dat, odeslání TCP segmentů po 1024 bajtech znamená 10 paketů, každý o velikosti 1064 bajtů. Každý paket bude ovšem s danou hodnotou MTU rozdělen na dva fragmenty, tedy celkově 20 paketů. Kvůli zmíněné vlastnosti stoupá potřeba využití zdrojů a počet paketů je vyšší, než kolik je opravdu potřeba.

Ztráta fragmentu vede ke snížení výkonnosti

Technika defragmentace není zrovna robustní. V okamžiku, kdy je odeslaný paket příliš velký a je potřeba jej fragmentovat, ztráta jediného fragmentu vyžaduje od protokolu vyšší vrstvy opětovné odeslání veškerých dat původního datagramu, i přesto, že většina fragmentů byla v pořádku přijata a defragmentována. Zmíněný problém může vést ke znatelnému snížení výkonnosti, než jakého by bylo možné dosáhnout, kdyby odeslané pakety nevyžadovaly fragmentaci.

K zahození fragmentů může dojít například v okamžiku, kdy nastane vlivem vysokého síťového provozu k přetížení a síťové uzly mají za úkol provoz určitým způsobem limitovat. Zahozením jediného fragmetu však nedojde k ulehčení situace, ale naopak ještě k většímu zhoršení, jelikož zahozením fragmentu musí dojít k retransmisi celého datagramu, což znamená více příchozích fragmentů. Ve výsledku tak dochází k nárůstu provozu a nižší propustnosti.

Efektivní defragmentace je obtížná

Vzhledem k pravděpodobnosti počtu ztracených fragmentů a informací obsažených v IP hlavičce existuje mnoho situací, ve kterých nabývá proces defragmentace horších výsledků, než je požadováno.

V IP hlavičce se nenachází žádná položka, která by u fragmentovaných paketů indikovala, kolik fragmentovaných paketů ještě bude následovat nebo jaká je velikost celkového datagramu. Jediným způsobem, jak se orientovat v rámci celkového datagramu, jsou položky Fragment Offset, které udávají pozici fragmentu v původním datagramu, a položka More Fragments, udávající poslední fragment. Dle [26] musí být každý host schopen přijmout datagram dlouhý 576 bajtů. Ovšem větším datagramům nemusí dostupná paměť postačit a není tak možné vykonat defragmentaci.

Dalším problémem při defragmentaci fragmentovaných paketů může být například odlišné pořadí fragmentů při přijetí, než v jakém byly fragmenty odeslány. Jednotlivé fragmenty totiž mohou v závislosti na strategiích použitých v síťových uzlech putovat jinou cestou. Ve výsledku se tak mohou mísit fragmenty z odlišných datagramů, pro které nebude k dispozici volná paměť, která by se mohla uspořit, kdyby fragmenty ze společného

datagramu přicházely ve správném pořadí a nebyly promíchány s fragmenty odlišných datagramů.

Obtížnou rolí je také odhadnout interval, po který budou fragmenty uchovány. Jediným prvkem udávajícím dobu, od které si může příjemce odvodit daný iterval, je hodnota TTL příchozích fragmentů. Bezmyšlenkovitý výběr velikosti TTL povede na jednu stranu k příliš dlouhému čekání na zbylé fragmenty, čímž dochází k vyššímu zahlcení paměti, na druhou stranu nízká hodnota TTL zapříčiní příliš včasné zahazování fragmentů a s tím související retransmisi.

Další nevýhodou fragmentovaného provozu je, že potenciální útočníci mohou zneužít nevýhod fragmentace ve svůj prospěch. Problematice fragmentovaného provozu v útocích (zejména v DoS útocích) se věnuje kapitola 3.

2.2.6 Předcházení fragmentace provozu

Fragmentovaný provoz je kvůli svým nevýhodám, které spíše předčí své výhody považován za jakési nutné zlo. Je-li to možné měl by se uživatel v normální situaci fragmentaci vyhnout. Existují dva obecné způsoby, jež jsou využívány právě pro omezení fragmentace [25].

Path MTU Discovery

Prvním ze způsobů je technika Path MTU Discovery, zkráceně PMTUD [30]. V momentě, kdy potřebuje uživatel odeslat paket skrze rozhraní, rozhoduje právě hodnota MTU, kolik dat může obsahovat jeden paket. Přičemž ne vždy je hodnota MTU rovna pro všechny linky na cestě k požadovanému cíli.

PMTUD vyžaduje pro svou činnost dvě základní komponenty – nastavení Don't Fragment bitu v IP hlavičce a ICMP zprávu typu Destination Unreachable, Fragmentation Needed. Narazí-li příchozí paket na rozhraní s nižší hodnotou MTU, Don't Fragment bit nepovolí směrovači daný paket fragmentovat. Směrovač je tak nucen paket zahodit a vygenerovat novou ICMP zprávu, s požadavkem na fragmentaci. Naštěstí odeslaná ICMP zpráva obsahuje také MTU hodnotu, kvůli které byl zmiňovaný paket zahozen. S vědomostí MTU hodnoty přijaté v ICMP zprávě může zdrojová stanice vytvořit nový paket s adekvátní velikostí.

Proces PMTUD však nekončí, jelikož nadále odesílá pakety s nastaveným Don't Fragment bitem a zjišťuje, zda se v cestě k cíli nenachází další rozhraní s nižší hodnotou MTU, kterou je možné využít. PMTUD technika nebude samozřejmě fungovat v případě, kdy jsou blokovány ICMP zprávy firewally nebo filtračními zařízeními.

Maximum Segment Size

Druhou metodou je nastavení TCP Maximum Segment Size (MSS) hodnoty [35], která je definována jako „maximální množství dat, jež je host schopen přijmout v jediném TCP/IPv4 datagramu, přičemž daný datagram může být fragmentován“. Při TCP spojení si obě komunikující strany vymění svou MSS hodnotu skrze položku option v TCP hlavičce (pouze u TCP SYN segmentů). Přestože zní logicky, aby se obě komunikující zařízení předem domluvila na společné velikosti MSS, odesílatel pouze musí limitovat velikost dat tak, aby byla menší nebo rovna MSS hodnotě příjemce.

Z důvodu omezení fragmentace u koncových zařízení TCP spojení byla volba hodnoty MSS změněna na menší hodnotu z paměti přidělenou MSS a velikostí MTU výchozího rozhraní (nepočítaje IPv4 hlavičku a TCP hlavičku, tj. 40 bajtů). MSS tedy nyní funguje

tak, že odesílatel porovná velikost MTU svého výchozího rozhraní s velikostí své paměti a vybere menší z nich jako hodnotu MSS, kterou odešele příjemci. Příjemce poté opět porovná velikost příchozí MSS hodnoty s velikostí MTU svého rozhraní a vybere nižší z nich jako hodnotu MSS, kterou bude nadále odesílat.

Fragmentaci je předcházeno pouze u koncových bodů TCP spojení, jelikož jsou brány v potaz MTU hodnoty obou rozhraní, avšak po cestě mezi směrovači jednotlivých koncových zařízení může být stále potřeba provoz fragmentovat z důvodu nižší hodnoty MTU ostatních směrovačů.

Kapitola 3

Zneužití fragmentace k útokům

3.1 Útoky typu odepření služby

DDoS (angl. Distributed Denial of Service) jsou kybernetické útoky, které se v dnešní době staly jednou z největších hrozeb Internetu. Jedná se o rychle rostoucí problém z pohledu síťové bezpečnosti. Množství a rozmanitost jak útoků, tak obranných mechanismů je obrovská, jelikož se útočníci neustále snaží zdokonalovat techniky tak, aby pronikli stávajícím zabezpečením. Na oplátku se snaží vývojáři zdokonalovat již existující obranné strategie, ale také vytvářet strategie nové [18].

Hlavním cílem útočníka je odepření přístupu legitimním uživatelům k síťovému systému, síťové službě, webové stránce, aplikaci či jinému síťovému zdroji, respektive službě. Důsledkem DDoS útoku je typicky zpomalení reakce systému nebo jeho úplné odstavení. Klíčovým zdrojem informací uvedených v kapitole je [33].

Útok, který vychází pouze z jediného zařízení, se nazývá Denial-of-Service (DoS) útok. V dnešní době je mnohem bežnější distribuovaný typ DoS útoku zvaný Distributed Denial-of-Service (DDoS) útok, zprostředkovaný z vícero zdrojů, jež koordinuje jeden centrální uzel. Útoky jsou tedy složené z toků paketů z odlišných zdrojů. Veškeré toky jsou směrovány na zařízení oběti, kde se střetávají. Distribuovaná zařízení společnou kooperací generují toky, jež jsou těžko detekovatelné jako útok a mitigace provozu je velmi obtížná. Díky distribuovanému přístupu je napadená oběť vystavena útoku v neporovnatelně větším měřítku a s potenciálně znatelnějšími škodami.

Většina DDoS útoků je navržena tak, aby spotřebovala veškerou dostupnou šířku pásma nebo prostředky cílové sítě, systému, webové stránky. Útočník využívá jednu z mnoha metod a nástrojů k tomu, aby zahltil cíl záplavou škodlivých nebo obtěžujících požadavků, zneužil síťový protokol či využil jeho vlastní zranitelnosti takovým způsobem, že systém již není schopen na požadavky dále reagovat. DDoS útok by se dal ve své podstatě přirovnat k situaci, kdy se do fronty zákazníků se vstupenkami začnou náhle vměšovat zákazníci se vstupenkami padělanými. Legitimní držitelé vstupenky stojící v řadě by se tak nikdy nedostali na řadu.

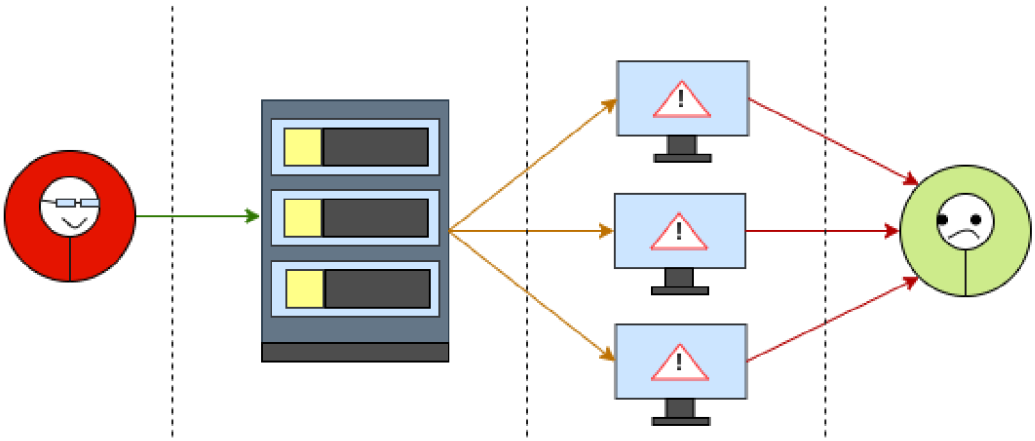
Ať už se jedná o DoS, nebo DDoS, výsledek je shodný – legitimní uživatelé nejsou schopni se připojit ke zdrojům a využívat jejich služeb. DDoS útok je jednou z nejefektivnějších metod, jakou může útočník porušit dostupnost, jeden ze tří základních bezpečnostních principů – důvěrnost, integrita, dostupnost, známých také jako CIA triad.

Role Botnetu v DDoS útocích

Pro útočníka s jediným zařízením by bylo příliš složité vygenerovat tolik provozu, aby dokázal způsobit kolaps celé sítě či webové stránky. K zahlcení celé šířky pásma nebo k dosažení potřebného výkonu utočníci využívají tzv. botnety, což jsou stovky až tisíce zařízení připojených k internetu (zombie, nebo boti), kteří jsou nositeli malwaru a jsou pod kontrolou útočníka (bot master, či bot herder). Ve většině případů si uživatelé infikovaných zařízení ani nejsou vědomi, že se stali součástí útoku. Z jednoho, či více zařízení, navržených jako command and control (C&C) server, útočník vzdáleně vyšle „launch“ instrukci botům. Tyto systémy kolektivně disponují dostatečným výkonem, aby uskutečnily masivní útok, mnohem intenzivnější než takový, který by vycházel pouze z jediného stroje. Využitím botnetu má také útočník možnost skrýt svou identitu, jelikož je útok proveden z mnoha odlišných zařízení, které se zdají být legitimními uživateli. Obrázek 3.1 vykresluje zmiňovaný útok, který využívá C&C server a botnet. Úplně nalevo obrázku je útočník ovládající C&C server. V pravé části obrázku jsou nakažená zařízení směrující útok na oběť.

Útočník na počátku buduje své vlastní botnety skenováním internetu, kde hledá zranitelná zařízení, která následně nakazí malwarem, díky němuž se k zařízení může vzdáleně připojit a ovládat tak boty. V současnosti si naneštěstí útočníci nepotřebují botnety budovat, mohou využít pronajmutí DDoS-for-hire botnetu od operátorů, kteří je nabízejí za velmi nízkou částku pro krátkodobé, avšak efektivní útoky [16].

Zatímco stále velkou část botnetů tvoří infikované počítače, začíná stoupat výskyt botnetů tvořených z Internet of Things (IoT) zařízení. Této skutečnosti nahrává fakt, že IoT zařízení existuje enormní množství v řádu několika miliard v podobě různých hraček, kamer apod. a jejich počet stále prudce stoupá. Vzhledem k tomu, že jsou také perfektními kandidáty stát se potenciálním zombie z důvodu slabé (i žádné) ochrany, počet zneužitých zařízení pro účely útoku rovněž stoupá [24].



Obrázek 3.1: Botnet v DDoS útoku.

3.2 Způsoby ochrany proti DDoS útokům

Kultivovanost DDoS útočníků, stejně jako útočných nástrojů se postupem času zvyšuje. Je proto logické, že detekce a ochrana proti DDoS útokům se stává mnohem komplexnějším a implementačně složitějším problémem. Hlavním cílem obranných mechanismů je zajištění, aby oběť útoku co nejlépe přečkala přicházející nápor. V ideálním případě dokáže úspěšná

ochrana obsloužit požadavky všech legitimních uživatelů a filtrovat veškerý provoz útočníka. Většina dnešních ochranných opatření se skládá ze dvou kategorií. První kategorie zajišťuje detekci a přítomnost útočného provozu. Útočné pakety je možno detekovat například technikami zaměřenými na anomálie příchozího provozu. Druhá ochranná kategorie se snaží zmírnit škody napáchané útokem. Snížení poškození útokem tkví v redukci intenzity útočného provozu, čehož lze dosáhnout blokováním útočných paketů nebo lokalizací zdroje útoku. V každém případě je vhodné si ochranu pečlivě promyslet a zvážit vícero obranných mechanismů. Více rozmanitých, dobře sladěných opatření je schopno efektivněji zabránit útoku. Doporučená opatření proti DDoS útokům se dle [18] dají rozdělit následovně:

Preventivní opatření

Preventivní opatření plní svůj účel i v případě, že momentálně k žádnému DDoS útoku nedochází. Ačkoliv prevence před útoky zahrnuje opravu všech zranitelností, jež může útočník zneužít ve svém útoku, vyplatí se slabým článkům věnovat pozornost. Pod preventivními prvky si můžeme představit například:

- **Zombie prevence:** předchází vytvoření armády nakažených počítačů, jež útočník využívá v útoku.
- **Zabezpečení protokolu:** řeší problém zneužití protokolu, jako například TCP SYN útok, podvržený formát paketů, fragmentované útoky apod.
- **Systémová bezpečnost:** ochrana zařízení před nevyžádaným přístupem.

Mitigační opatření

Zahrnuje opatření regulující pravděpodobnost útoku (například autentizace), monitorování provozu a techniky využívané v době útoku. Typickými zastupci mitigačních opatření jsou například:

- **Monitorování síťových toků:** poskytuje informace o uživatelích připojených k síti, aplikacích, špičkách v provozu apod. Díky monitorování je možno odhalit škodlivý provoz v síti.
- **Monitorování síťového provozu:** odhaluje náhlé výkyvy v provozu, které mohou indikovat DDoS útok.
- **Algoritmy proti DDoS útoku:** jednoduché algoritmy implementované v rámci operačního systému zmírňující dopady DDoS útoku.

Detekční strategie

Detekční strategie umožňují zviditelnění a odhalení útoků. Dvěma základními typy detekčních strategií jsou:

- **Detekce anomálií:** periodicky porovnává chování systému v normální situaci (kdy nedochází k žádnému útoku) s aktuálním modelem, čímž detekuje abnormality – potenciální útoky. Klíčovým prvkem při detekci je rovnováha mezi přesnou identifikací útoku a náchylností k označení legitimního chování jako anomálie.

- **Detekce vzorů:** vyhledává přítomnost vzorů, které jsou typické pro dobře známé útoky. Dochází zde k efektivní detekci známých útoků. Nové útoky nebo lehce pozměněné staré útoky však mohou detekci vzorů uniknout zcela bez povšimnutí.

Reaktivní mechanismus

Mechanismus je schopen se příslušně vypořádat s právě probíhajícím útokem. Svého výsledku mechanismy dosahují tak, že omezí dopad útoku, nebo přímo identifikují útočníka. Ke své činnosti vyžadují určitou míru volnosti, jelikož se musí vyrovnat s obrovským množstvím škodlivého provozu. Charakteristickými příklady reaktivních mechasnismů jsou:

- **Filtrace provozu:** je efektivní metodou k zastavení DDoS útoku. Funguje tak, že zahazuje veškerý nechtěný nebo škodlivý provoz. Filtrace škodlivého provozu není nijak jednoduchá. Útočníci mohou používat legitimních služeb k útoku a ty tak není možné filtrovat. V nejhorším případě může útočník zneužít vad filtrace provozu ve svůj prospěch a zapříčíní zahazování legitimního provozu. Metod, jakými lze filtrovat nebezpečný provoz, je velké množství a řadí se do vlastních podkategorií.
- **Omezení rychlosti (anglicky Rate limiting):** metoda vynucující limitaci podezřelých paketů, které byly vybrány jako podezřelé některou z detekčních strategií.

Analýza po útoku

Data uchovaná během DDoS útoku je potřeba analyzovat a náležitě se z nich poučit pro budoucí účely. Díky uloženým datům může oběť útoku upravit své obranné strategie a parametry. Způsoby vhodné analýzy po útoku jsou například:

- **Systémové záznamy (anglicky System logs):** uchování dat z průběhu celého útoku, která mohou být použita k forenzní analýze. Z uložených dat je také možné určit, o jaký typ útoku se jedná.
- **Rekonstrukce útoku:** určení entit, které provedly útok. Některé informace o útoku nemusí být zprvu zřejmé a vyplynou na povrch až po zpětném pohledu na předešlý útok.

3.3 Zneužití fragmentace v DDoS útocích

IP fragmentace může být útočníky zneužita různými způsoby a cílem útoku nemusí být pouze koncový systém, ale i bezpečnostní komponenty nacházející se mezi útočníkem a cílovou obětí. K úspěšné defragmentaci je zapotřebí příjem všech fragmentů, proto se fragmentované útoky využívající protokol UDP/ICMP obvykle skládají z falešných fragmentů, které nelze defragmentovat. Dočasné uchování falešných fragmentů v cílovém zařízení s sebou přináší spotřebu paměti určenou pro defragmentaci. V nejhorším případě útok vyčerpá veškerou přidělenou paměť [6]. Mezi nejznámější DDoS útoky zneužívající fragmentovaný provoz ke svým škodlivým účelům patří:

Fragmented ACK Flood

Tento typ útoku využívá pakety větších rozsahů (1500 bajtů a víc) k zahlcení co největší šířky pásma, zatímco takových paketů generuje poměrně nízký počet. Jelikož směrovače

neprovádí defragmentaci na IP vrstvě, mohou tyto vygenerované pakety bez zábran projít skrze jednotlivé směrovače, ACL, firewally a IDS/IPS, nebo vyčerpat značnou část prostředků přidělených k defragmentaci. Obsah paketů bývá typicky naplněn náhodnými, nesmyslnými daty. Hlavním cílem útočníka může být zaplnění veškeré šířky pásma sítě oběti nebo za pomocí fragmentace skrýt jiný DDoS útok zaměřený na aplikační vrstvu či jiné zákeřné útoky [10].

HTTP Fragmentation

V případě HTTP Fragmentation útoku naváže bot korektní HTTP spojení s webovým serverem. Bot následně rozdělí legitimní pakety do fragmentů o co nejmenší možné velikosti, které pak vysílá s maximálním zpožděním, jaké hodnota timeout webového serveru povolí. Pro webové servery jako například Apache, které nemají explicitně nastavenou vhodnou hodnotu pro timeout, může být základní hodnota příliš dlouhá. Nastane-li takový případ, je pro boty snadné otevřít několik různých spojení a společně způsobit výpadek webového serveru. Ačkoliv je dnes tento typ útoku vzácný, z důvodu řádné prevence ze strany softwarových vývojářů je stále zapotřebí být si vědom potencionální hrozby a s útokem počítat [11].

ICMP Fragmentation Flood

Jedná se o typ útoku, který využívá rozsáhlé ICMP fragmenty (opět 1500 bajtů a více), generovaných co největší rychlostí, přičemž jednotlivé fragmenty nelze defragmentovat. Útok vede ke zbytečnému alokování zdrojů potřebných k defragmentaci a také k zahlcení šířky pásma oběti. Často je následkem útoku přetížení serveru oběti [14].

UDP Fragmentation

Je variací DDoS útoku zvaného UDP Flood, ve kterém server oběti přijímá velké množství UDP paketů o velikostech podobných jako u ICMP Fragmentation Flood útoku (1500 bajtů a více). UDP nevyžaduje spojení v podobě handshake jako TCP, což činí tento typ útoku těžce detekovatelným a extrémě efektivním v zahlcení celé šířky pásma oběti. Napadený server je tak přetížen, protože se snaží vynaložit potřebné prostředky k defragmentaci nelegitimních paketů [14].

Teardrop

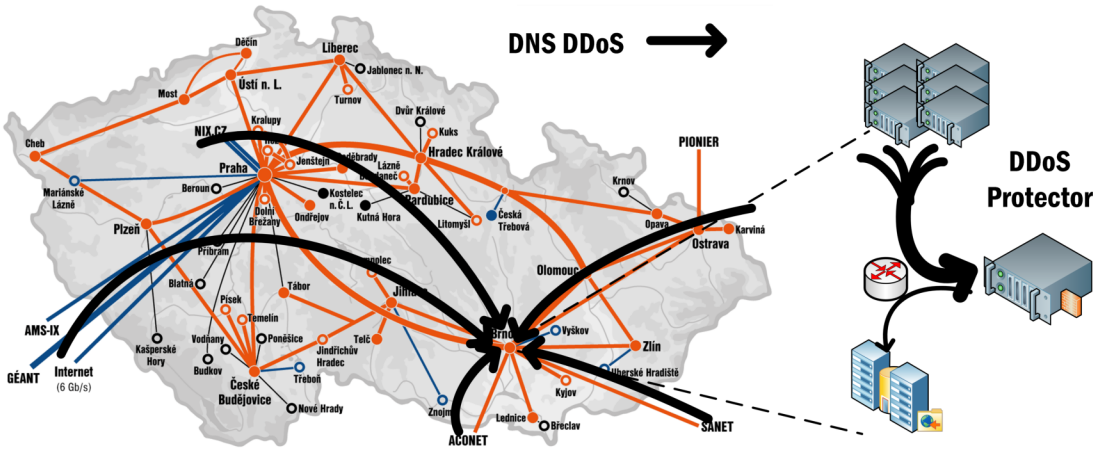
Teardrop je typ DDoS útoku, který napadá cílový TCP/IP mechanismus a znemožňuje znovu sestavení rozfragmentovaných paketů. Útok zapříčiňuje překrytí dat z důvodu nesprávného nastavení offsetů pro jednotlivé fragmenty a rychle přetěžuje server oběti, což končí jeho selháním. I přesto, že Teardrop útok využíval zranitelnosti předešlých operačních systémů Windows a dnes v nejnovější verzi operačního systém je chyba opravena, je potřeba nezapomenout na potencionální hrozbu teardrop útoku [15].

Kapitola 4

Zařízení DDoS Protector

Sdružení CESNET si vyvíjí svou vlastní DDoS ochranu zvanou DDoS Protector, taktéž známý jako Čistička. DDoS Protector je řádově levnější než obdobné projekty. Skládá se ze serveru a dedikované síťové karty vybavené FPGA čipem a vlastním firmwarem. FPGA zaručuje rychlý přenos a filtraci dat. Server vyhodnocuje vlastnosti síťového provozu a v případě útoku povolí FPGA filtraci [3].

Většina DDoS útoků spadá do kategorie tzv. Volumetrických útoků, pod které patří více typů útoků. Jejich záměrem je vyslání co největšího množství provozu a zahltit tak veškerou šířku pásma oběti. Pro menší sítě může množství takového provozu znamenat nezvladatelný problém. Útok totiž není možné zpracovat přímo v malé síti se zahlcenou přípojnou linkou, ale ihned na úrovni poskytovatele této sítě. Linky poskytovatele mají vysoké kapacity a útok je tak nijak zvlášť nepostihne. Sdružení CESNET je právě v roli jednoho z poskytovatelů a jeho cílem je chránit sítě připojených organizací [27]. CESNET připojuje organizace, jako jsou univerzity a jiné akademické instituce. Obrázek 4.1 znázorňuje situaci, ve které dochází k DNS DDoS útoku na některou z organizací, jejichž poskytovatelem je právě sdružení CESNET. Útok je nejprve směrován na DDoS Protector, ten zablokuje škodlivý provoz a legitimní provoz dále směruje na cílovou organizaci.



Obrázek 4.1: Směrování DNS útoku skrze DDoS Protector [19].

Cílem DDoS Protectoru je snížit objem provozu pro cílovou organizaci na úroveň, kterou je možno zpracovat. Výhodou DDoS Protectoru je funkcionalita na míru a několik implementovaných algoritmů pro účely mitigace. Principem Čističky je přesměrovat útočný

provoz směřovaný na oběť právě na Čističku. Dalším krokem v době útoku je kontrola provozu a následné zahození paketů na základě mitigační strategie. Za normálních okolností DDoS Protector nezahazuje žádný provoz směrovaný skrze něj. S omezováním provozu začne až v momentě, kdy zaznamená útok a snaží se procházející provoz filtrovat na vypočtenou optimální hodnotu. O zahazovaném provozu jsou vedeny statistiky v logovacích souborech exportovaných pomocí NetFlow/IPFIX protokolu. V logovacích souborech se typicky nachází informace jako počet zahozených bajtů, zahozených paketů apod. Detekci útoku a následnou filtraci provádí DDoS Protector na základě konfigurace obsahující pravidla. Jednotlivá pravidla jsou definována síťovým administrátorem. Pravidla a limity pravidel definuje administrátor dané sítě na základě svých zkušeností a monitoringu objemu provozu ve své síti. Mitigační pravidla se skládají ze tří základních částí:

1. Podmínky – třídí pakety, které splňují požadavek na filtraci (například cílová IP adresa, zrojové číslo portu apod.).

2. Limity – Limit určuje, kolik bajtů/paketů musí být přijato, aby se jednalo o DDoS útok a spustil se tak proces mitigace.

3. Optimální provoz – blíže specifikuje úroveň, na jakou se má provoz omezit po spuštění mitigace.

DDoS Protector má vhodné rozhraní pro vytvoření nových mitigačních strategií. Jednou z hlavních úloh této práce je právě vytvoření nových mitigačních strategií zaměřených na ochranu proti fragmentovaným DDoS útokům. Obrana proti fragmentovanému útoku vyžaduje speciální chování blíže popsané v kapitole 5.

4.1 Amplifikační modul

Jedním ze standardních modulů DDoS Protectoru je Amplifikační modul. Nově vytvořené mitigační strategie v této práci jsou rozšířením amplifikačního modulu. Cílem modulu je především detekce a mitigace DDoS útoku na základě amplifikačních pravidel. Pravidlo se skládá z chráněného prefixu, popisu provozu (protokol, rozsah portů apod.), speciální hodnoty threshold a limitu. Threshold slouží k určení hranice, po jejímž dosažení má dojít k mitigaci. Limit určuje míru provozu, na jakou se má současný provoz redukovat. Vlivem vytvoření nových strategií v Amplifikačním modulu se do pravidla přidala také možnost volby strategie [2]. V Amplifikačním modulu se po daném intervalu (nastaveném v konfiguračním souboru) opakuje tzv. mitigační cyklus. V mitigačním cyklu se odehrává téměř veškerá klíčová činnost celého modulu. Přibližně lze celý proces mitigačního cyklu popsat následovně (popis doplňuje obrázek 4.2):

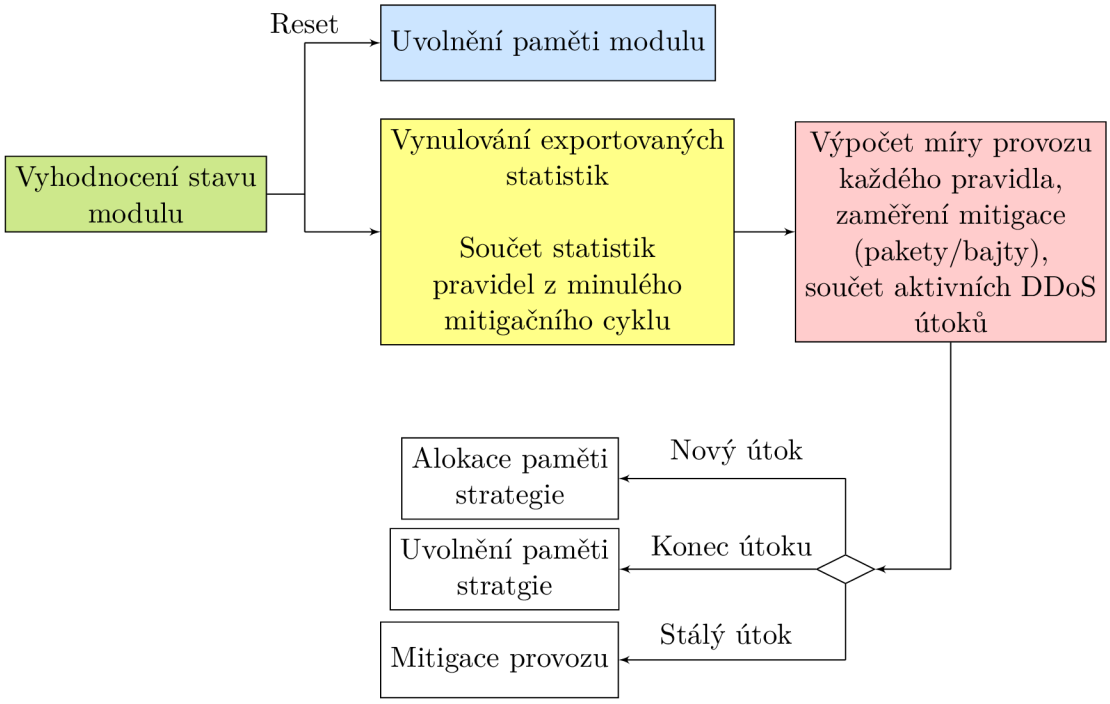
1. Vyhodnocení stavu amplifikačního modulu – deaktivace, restart (zelený blok). Dojde-li k některé ze zmíněných situací, musí se uvolnit veškerá dynamická paměť alokovaná v amplifikačním modulu, včetně paměti přidělené mitigačním strategiím (modrý blok).

2. Vynulování statistik exportovaných v předešlém mitigačním cyklu. Mezi exportované statistiky patří záznamy o nových, právě probíhajících a ukončených DDoS útocích v předešlém cyklu. Dále se sčítají statistiky pravidel z minulého mitigačního cyklu o počtu paketů a bajtů, které nesplnily podmínky žádného pravidla. Uchovává se

také provoz, který splňuje podmínky některého z pravidel. Poslední z uchovávaných statistik je seznam IP adres zablokovaných hardwarovou nebo softwarovou částí (žlutý blok).

3. Po zpracování statistik z minulého mitigačního cyklu přichází na řadu vyhodnocení aktivních DDoS útoků. Pro každé z pravidel se určí míra provozu vzhledem k počtu paketů a bajtů. Proběhne vyhodnocení, která ze dvou vypočtených hodnot je vyšší a má od této chvíle sloužit jako měřítko provozu mitigačním strategiím. Nakonec se určí celkový počet aktivních DDoS útoků (červený blok).

4. Poslední úlohou mitigačního cyklu je samotná mitigace a s tím spojené činnosti. Proces zde může procházet čtyřmi stavy. První stav nastane při novém DDoS útoku. Na základě vybrané mitigační strategie se alokuje dynamická paměť pro účel mitigace. Dalším možným stavem je uvolnění paměti mitigační strategie, ke kterému dojde po skončení útoku. Třetí stav odpovídá již aktivní mitigaci. Poslední stav nastane v momentě, není-li potřeba provedení žádné akce a tudíž není aktivní žádný DDoS útok.



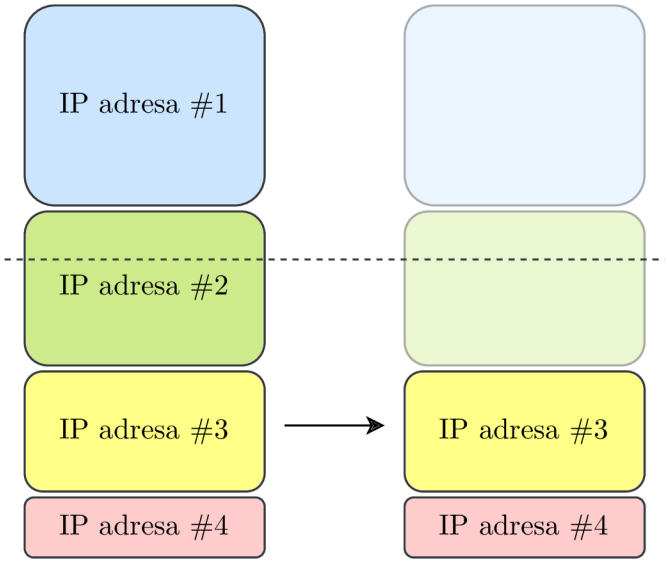
Obrázek 4.2: Mitigační cyklus Amplifikačního modulu.

TopN strategie

Jedinou mitigační strategií, kterou doposud Amplifikační modul používal, byla strategie TopN. V případě, že provoz u pravidla dosáhl prahové hodnoty, spustila se mitigace využívající aktivní strategii. Mitigace vyžaduje seřazení IP adres příchozího provozu podle jeho objemu sestupně. Uspořádané adresy strategie postupně blokuje od největšího po nejmenší, dokud nedosáhne optimálního provozu daného pravidla. Při blokování se zahodí veškerý provoz IP adresy. K mitigaci provozu dochází na konci mitigačního cyklu amplifikačního modulu. Kvůli celkovému zahazování provozu není strategie vždy nejideálnějším řešením.

Součástí bakalářské práce je implementace nové mitigační strategie popsané v 5.3, rozšiřující vlastnosti strategie TopN.

Na obrázku 4.3 je provoz skládající se ze čtyř IP adres seřazených sestupně. Po mitigaci provozu z obrázku nastane situace, kdy se kompletně zablokuje první adresa. Jelikož hodnota aktuálního provozu po zablokování adresy není nižší než optimální hodnota, je zapotřebí blokovat také druhou adresu. Omezením veškerého provozu druhé nejpočetnější adresy vzniká poměrně značné nevyužité místo. Efektivnější variantou by mohlo být částečné omezení blokovaných adres tak, aby se lépe využilo optima.



Obrázek 4.3: Mitigace použitím TopN strategie.

4.2 SYN flood modul

SYN flood modul je dalším standardním modulem Čističky. Jak samotný název napovídá, jedná se o modul detekující SYN flood útoky. Syn flood je typ DoS útoku, postihující hosty, kteří se snaží navázat spojení se serverem za pomoci TCP protokolu. Útok využívá stavu, kdy server po přijetí SYN zprávy odešle zpět SYN-ACK zprávu a čeká na opětovné potvrzení ACK zprávou. Server je tedy ve stavu, kdy vyčkává na potvrzení, které ovšem útočník nikdy neodešle. Základní myšlenkou útočníka je vyslat tolik SYN zpráv, že server nebude kvůli svému čekání na potvrzení schopen zpracovat požadavky legitimních uživatelů [13]. Pro detekci využívá SYN flood modul několik svých submodulů. Submoduly pro svou práci využívají různé detekční algoritmy. V současné době jsou v Čističce implementovány tři detekční algoritmy. Těmito algoritmy jsou [4]:

SYN drop

Modul SYN drop slouží pro mitigaci SYN flood útoku na základě rozhodovací tabulky. Vstupem z hlavního modulu jsou pakety s příznakem SYN nebo ACK. Modul si udržuje čítače viděných SYN a ACK příznaků pro jednotlivé zdrojové IP adresy, které komunikují s chráněnou sítí. Čítače jsou po určitých časových intervalech resetovány. Délka intervalu, po které se mají resetovat čítače, je specifikována v konfiguračním souboru. Modul rozhoduje pouze o zahození/přeposlání paketů s příznakem SYN, potřebných pro ustavení komunikace.

Prochází-li paket s příznakem ACK, je přeposlán vždy. Podle rozhodovací tabulky[1] je určena operace, která se má provést pro SYN paket. V tabulce se vyhledává podle aktuálních hodnot čítačů výsledná operace.

ACK spoofing

Acknowledge spoofing modul pracuje na úrovni firewallu či proxy. Chráněnému naslouchajícímu uzlu posílá uměle vytvořené ACK zprávy jako odpovědi na SYN-ACK zprávy. Pomocí této metody nejsou vyčerpány prostředky chráněného uzlu při výskytu velkého počtu nedokončených spojení. Po odeslání uměle vytvořené ACK zprávy čeká modul určitou dobu na ACK zprávu od legitimního uživatele. Pokud legitimní zpráva nedorazí, odešle modul chráněnému uzlu spoofnutou RST zprávu. Pro legitimní spojení pokračuje tok paketů normálním způsobem.

RST cookies

Myšlenkou RST cookies metody je, že legitimní host zašle TCP reset paket v momentě, kdy je TCP spojení v nesynchronizovaném stavu. Na základě této zprávy vytváří submodul seznam ověřených hostů (whitelist). Proces ověření hosta funguje tak, že submodul vždy zahodí první SYN paket od daného hosta a pošle mu zpět neplatný SYN+ACK paket. Legitimní host na tento paket odpoví RST paketem, čímž je dokončeno jeho ověření. Následně host standardně přepošle původní SYN paket a komunikace pokračuje obvyklým způsobem. Nevýhodou této metody je to, že první pokus každého klienta o ustavení TCP spojení vždy selže, to ale nepředstavuje zásadní problém.

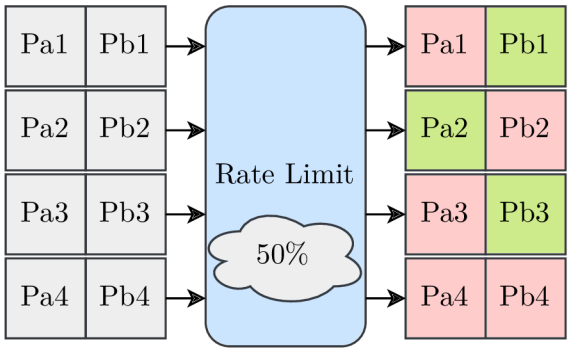
SYN flood modul řídí činnost jednotlivých submodulů. Modul při zpracování paketů očekává pouze pakety s nastaveným určitým příznakem. Takové pakety poté porovnává se všemi pravidly. Pokud nemá vstupní SYN paket nastaven ACK příznak a sedí cílová adresa paketu s adresou chráněné sítě daného pravidla, je paket přičten k počítadlu paketů odpovídajících tomuto pravidlu. Pokud množství příchozích SYN paketů překročí určitou mez, povolí se detekce potenciálního útoku. Za účelem případné mitigace započne pro dané pravidlo sledování provozu. Paket je předán do jednoho ze submodulů na základě strategie pravidla. Submodul vrací hlavnímu modulu rozhodnutí, zda má být paket zahozen, či propuštěn. Pravidla pro SYN flood modul jsou obdobně jako v Amplifikačním modulu zadána uživatelem prostřednictvím databáze.

[1] https://redmine.liberouter.org/projects/ddos-protector/wiki/SYN_flood_modul

Kapitola 5

Vlastní strategie mitigace

V rámci této práce byly do amplifikačního modulu implementovány dvě nové strategie. Hlavním důvodem pro vytvoření nových strategií byla schopnost vhodným způsobem limitovat fragmentovaný provoz, protože doposud žádná metoda v Čističce problémy fragmentovaného provozu neřeší. První nová strategie využívá k mitigaci takzvaného rate-limitingu. Obyčejný způsob rate-limitingu není vhodný pro fragmentovaný provoz, jelikož limituje jednotlivé pakety namísto celých toků. Představit si zmiňovaný problém můžeme na příkladu se síťovým směrovačem aplikujícím rate-limiting strategii. Dopomůže nám obrázek 5.1. Obrázek znázorňuje čtyři příchozí datagramy, které mají být směrovačem limitovány o 50 %. Každý datagram je složen ze dvou fragmentů. Proces limitace je aplikován na každý fragment. Jelikož není nijak řešeno, do jakého datagramu fragment patří, je běžné, že dochází k poškození velkého množství síťových toků a nedochází tak k redukci provozu, ale k jeho nárůstu. U poškozeného toku totiž musí dojít k retransmisi celého datagramu. Obrázek 5.1 similuje přesně takovou situaci, kdy důsledkem obyčejného rate-limitingu došlo k zahození vždy alespoň jednoho fragmentu z datagramu a tím se poškodily veškeré síťové toky.



Obrázek 5.1: Obyčejný rate-limiting

Bylo proto potřeba vytvořit novou strategii tak, aby náležitě limitovala fragmentovaný provoz. Nová Rate Limit strategie je popsána v sekci 5.1. Strategie využívá k definování toků klíč. Tento klíč byl v prvotní verzi složen kombinací pevně daných položek (zdrojová IP adresa, cílová IP adresa, Identification z IP hlavičky). Vylepšení nové strategie v podobě dynamického klíče a tedy definicí univerzálních toků se věnuje sekce 5.2. Druhá ze strategií kombinuje vlastnosti již existující strategie TopN a nové Rate Limit strategie 5.3. Kombinací obou zmíněných metod vznikla strategie, která neblokuje celkový provoz, jako je tomu u TopN, a zároveň je vhodná pro fragmentovaný provoz. Volba strategií se nachází na

úrovni definice amplifikačního pravidla v databázi. Tato volba byla přidána až s novými strategiemi, poněvadž dříve s jedinou strategií nebyla potřeba. Kromě samotných strategií bylo také zapotřebí náležitě upravit strukturu amplifikačního modulu, který počítal pouze s jedinou strategií.

Absence čísel portů fragmentovaných paketů zapříčiňující problémy při aplikaci filtračních pravidel motivovala k vytvoření nového modulu. Nový modul se nazývá Ports Inference modul a zabývá se jím sekce 5.5. Jeho cílem je chybějící čísla portů fragmentům přiřadit. Jelikož se jedná pouze o heuristiku, přiřazení portů není vždy zaručeno. Zejména v případě, kdy prostřední fragment neobsahující porty dorazí do Čističky ke zpracování dříve, než první fragment z datagramu. Proto bylo před samotným vývojem modulu potřeba zjistit, zda mají fragmenty tendenci přicházet ve správném pořadí, či nikoliv. Analýze reálného fragmentovaného provozu se věnuje sekce 5.4.

Hlavními úkoly z pohledu integrace nových strategií byly:

1. Rozšíření formátu pravidla o možnost volby mitigační strategie. Úprava struktury amplifikačního modulu odpovídajícím způsobem pro nové strategie.
2. Přidat rozdělení módu (TopN / Rate Limit) do zpracování paketů vůči pravidlům v amplifikačním modulu.
3. Implementace Rate Limit strategie a s tím související dynamický klíč.
4. Implementace Kombinované strategie.
5. Analýza fragmentovaného provozu.
6. Vytvoření Ports Inference modulu pro odvození čísel portů fragmentovaným paketům.

5.1 Rate Limit strategie

První novou strategií je Rate Limit strategie vhodná pro fragmentovaný provoz. Rate limiting je využíván pro řízení objemu síťového provozu. Cílem strategie je v době útoku limitovat fragmentovaný provoz na takovou úroveň, kterou lze zpracovat. Efektivní limitování fragmentovaného provozu vyžaduje zahození nebo propuštění celých toků. Jedním tokem jsou myšleny všechny fragmenty datagramu. Obyčejná rate limit strategie není vhodná pro obranu proti fragmentovanému útoku, pracuje totiž tzv. „per-packet“. Tento způsob činnosti vede k poškození více síťových toků. Nepřispívá tedy k redukci provozu, ale naopak k jeho nárůstu, jelikož při zahození jediného fragmentu musí dojít k retransmisi celého datagramu. Síťové směrovače typicky obsahují právě i funkcionalitu rate-limitingu, která je ale nevhodná pro fragmentovaný provoz. Proto je potřeba využít specializovaného zařízení jako je Čistička, ta nabízí implementovat vylepšené strategie vhodné pro fragmentovaný provoz. Výsledná operace strategie (zahození, propuštění) musí být shodná pro celý tok, tzv. „per-flow“ strategie.

Odlišení jednotlivých toků bylo docíleno vytvořením unikátního klíče. Klíč se v původní verzi strategie skládal z pevně daných položek – zdrojová IP adresa, cílová IP adresa a položka Identification z IP hlavičky. Pro limitaci provozu využívá strategie hashovací funkci, která převede vstupní klíč na hodnotu v určitém rozsahu. Amplifikační modul zpracovává všechny příchozí pakety následovně:

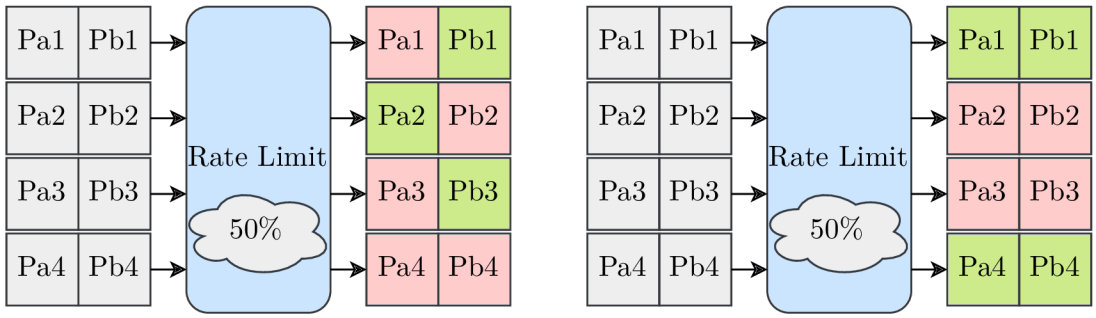
1. Pokud objem provozu odpovídající danému pravidlu (provoz s cílovou adresou chráněné sítě a další podmínky sestávající v první fázi pouze z fragmentovaných paketů) dosáhne prahové hodnoty, spustí se mitigace.

2. Pokud je zvolena Rate Limit strategie, spustí se.

3. Podle maximálního rozsahu hodnot hashovací funkce (N), množství aktuálního provozu odpovídajícího pravidlu (P) a hodnoty parametru pravidla limit (L) se určí prahová hodnota T = (LN)/P.

4. Z paketu se složí klíč (zdrojová IP, cílová IP, Identification) a zavolá se hashovací funkce. Výsledek hashovací funkce h(k) se porovná s prahovou hodnotou T. Pokud je větší (h(k) > T), paket je zahozen. Jinak je paket propuštěn.

Na obrázku 5.2 je porovnání obou přístupů rate-limitingu. V levé části je obyčejná strategie limitující pakety (například v síťových směrovačích). Na pravé straně je rate-limiting blokující celé toky (nová strategie v Čističce). Cílem obou strategií je blokovat 50 % příchozího provozu. Na vstup přichází 4 datagramy, každý složen ze dvou fragmentů. U obyčejné strategie je každý fragment zpracován nezávisle a výsledek propuštění se může lišit i pro jednotlivé fragmenty stejného datagramu. Může se tak stát, že je pokaždé z datagramu vybrán jeden fragment k zahození a tím dojde k poškození celého toku vyžadujícímu retransmisi datagramu. Na příkladu je situace, kdy dojde vždy k zahození alespoň jednoho fragmentu, tudíž dojde k poškození všech toků. Pravá strana obrázku vyobrazující přístup rate-limitingu celých toků přistupuje k limitaci jinak. Pro každý fragment z daného datagramu je operace propuštění shodná. O tom, zda budou fragmenty propuštěny, nebo zahozeny, rozhodují položky tvořící klíč. Vzhledem k tomu, že tok definující fragmentovaný provoz je u všech fragmentů z datagramu stejný, operace je rovněž totožná. Výsledkem je tedy propuštění, nebo zahození celého toku, čímž nedochází k tak razantnímu postupu jako u obyčejné strategie.



(a) Limitování paketů.

(b) Limitování toků.

Obrázek 5.2: Porovnání obyčejné Rate Limit strategie a nové Rate Limit strategie limitující celé toky.

Rate Limit strategii bylo možno využít obecněji. Rozšíření tkví ve vytvoření dynamického klíče. Dynamický klíč se neměl skládat pouze ze statických položek specifikujících fragmentovaný provoz, ale z klíče definovaného uživatelem. Princip dynamického klíče je vysvětlen v nadcházející sekci 5.2.

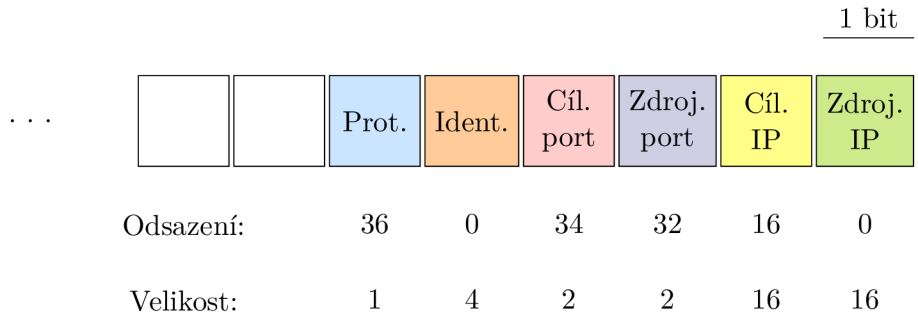
V každém mitigačním cyklu amplifikačního modulu bylo potřeba vypočítat pro každé pravidlo hodnotu, s níž se bude při zpracování paketu porovnávat výsledek hash funkce. Tato

hodnota je nazvána threshold. Hash funkce používaná u Rate Limit strategie má návratovou hodnotu `uint32_t`, tedy 32bitové nezáporné celé číslo. Z toho důvodu je maximální velikost threshold hodnoty rovněž 32bitové nezáporné číslo. Hash funkci se předává jako parametr klíč. Vytvoření klíče je řešeno zkopírováním hodnot, ze kterých se klíč skládá do paměti za sebe. To znamená v původní verzi, kdy se klíč skládal pouze ze zdrojové a cílové IP adresy společně s položkou Identification, se tyto jednotlivé hodnoty zkopírovaly do paměti za sebe a předaly se jako celek (klíč) hash funkci. Po vytvoření dynamického klíče se namísto pevně daných hodnot vyhodnocují různé položky definující tok.

5.2 Dynamický klíč

Smyslem dynamického klíče je odlišení toků definovaných uživatelem pro každé pravidlo. Dynamický klíč rozšiřuje možnosti jak Rate Limit strategie, tak Kombinované strategie. Uživatel si může namísto limitování fragmentovaného provozu sám určit, které položky mají definovat tok. Volba klíče probíhá přímo na úrovni definice pravidel v databázi. Uživatel má tedy možnost sám si vybrat veškerou komunikaci s hosty specifikovaným prefixem v pravidlu. Průběh limitace zůstává stejný. Z dynamického klíče se použitím hashovací funkce získá hash hodnota. Výsledná hodnota se porovná s výstupem mitigační strategie a nakonec se paket buď propustí, nebo zahodí. Přestože průběh mitigace se oproti statickému klíči nemění, bylo potřeba navrhnout a implementovat způsob, jak rozslišit, které položky klíče byly vybrány.

Dynamický klíč je v databázi uložen jako 32bitová hodnota. Nastavením jednotlivých bitů z této 32bitové hodnoty se vyberou položky dynamického klíče. Bity a jejich reprezentace jsou znázorněny obrázkem 5.3. Na obrázku je vidět 6 položek, jejichž kombinací může vzniknout dynamický klíč. Zbytek bitů je prázdný a v budoucnu může sloužit pro uložení více hodnot, ze kterých se klíč bude skládat.



Obrázek 5.3: Dynamický klíč a jeho položky.

Pro každé pravidlo se vytvoří šablona v podobě struktury jazyka C. Šablona obsahuje pole velikostí (bajty) a odsazení položek klíčů v rámci stuktury, ve které jsou v amplifikačním modulu uloženy. Šablona také uchovává počet elementů v jednotlivých polích. Identifikátor paketu se nachází v odlišné struktuře než zbylé hodnoty. Proto je v šabloně zapotřebí uchovávat pole velikostí a odsazení pro dvě odlišné struktury. Pro výpočet odsazení položky ve struktuře bylo použito makro jazyka C s názvem `Offsetof`. Makro vrací velikost odsazení položky od začátku struktury. Získání velikoti položky struktury je dosaženo makrem, které přetypuje null ukazatel na ukazatel na danou strukturu a operátorem sizeof dále získá přímo požadovanou velikost položky:

`sizeof(((type *)0)->member)`

Vytvoření šablony probíhá při inicializaci amplifikačního modulu. Zde se volá funkce `rl_create_template()`. Funkce má na svědomí vytvoření šablony danému pravidlu. Jako první parametr je funkci předávána 32bitová hodnota z databáze určující položky dynamického klíče. Základní strukturou využívanou při vytváření šablony je struktura zachycená tabulkou 5.1.

rl_hash_key_field_mapping_s	
position:	enum rl_hash_key_field_set_e
location:	enum rl_hash_key_field_location_e
offset:	size_t
size:	size_t

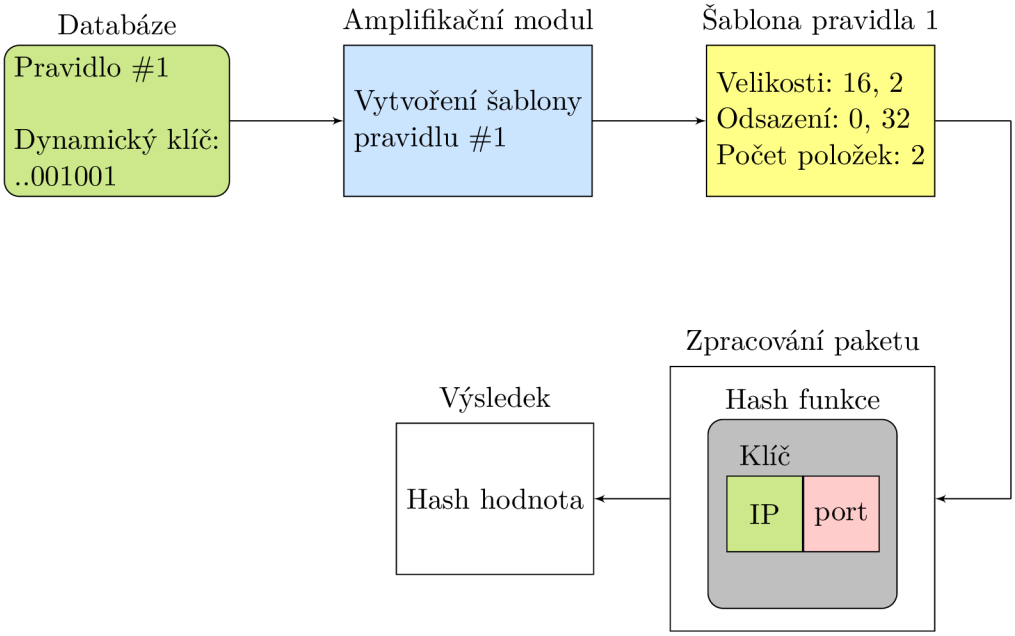
Tabulka 5.1: Struktura držící informace o jednotlivých položkách dynamického klíče.

Tato struktura obsahuje jako první člen pozici v databázi. Díky ní lze jednoznačně odlišit, o kterou položku dynamického klíče se jedná. Například zdrojová IP adresa je poslední z 32bitové hodnoty, proto by měla hodnotu pozice ve struktuře rovnu nule. Další informaci, kterou si struktura uchovává, je, jaké struktuře v rámci Čističky položka klíče náleží. Jako poslední se ve struktuře nachází odsazení a velikost položky. Obě hodnoty jsou získávány makry zmíněnými v předešlém odstavci. Jelikož je potřeba uchovávat popisované hodnoty pro každou položku klíče, bylo vytvořeno pole struktur. Výsledkem je tedy pole struktur, z nichž lze jednoduše vytvořit šablonu danému pravidlu. Formát pole pro bližší představu je následovný:

```
{
    {
        0, // zdrojová IP adresa
        1, // struktura uh_header
        0, // odsazení
        16, // velikost
    },
    {
        1, // cílová IP adresa
        1, // struktura uh_header
        16, // odsazení
        16, // velikost
    },
    ...
 }
```

Na obrázku 5.4 lze vidět vytvoření šablony, kdy je dynamický klíč pro pravidlo tvořen kombinací zdrojové IP adresy a cílového portu (zelený blok vlevo nahoře). Vytvoření šablon pro jednotlivá pravidla probíhá pouze jednou při inicializaci amplifikačního modulu (modrý blok). Pravidlu z obrázku se vytvořila šablona (žlutý blok), která obsahuje velikosti zdrojové IP adresy (16 bajtů) a velikost cílového portu (2 bajty). Dále šablona uchovává odsazení

položek ve struktuře. Zdrojová IP adresa je první položkou struktury a má odsazení nula. Cílový port se nachází až za oběma IP adresami, tudíž má odsazení o 32 bajtů. Poslední informací v šabloně je počet položek skládajících klíč. V tomto případě se jedná o zdrojovou IP adresu a cílový port, tedy 2 položky. Každý zpracovávaný paket v amplifikačním modulu při mitigování Rate Limit strategií nebo Kombinovanou strategií potřebuje získat svou hash hodnotu, dle které se bude mitigovat. Hash hodnotu paket získá načtením položek ze struktur daných šablonou. Položky se předají hashovací funkci (předposlední, šedý blok). Funkce navrátí paketu jeho hash hodnotu a proběhne porovnání, na základě kterého je paket propuštěn, či zahozen.



Obrázek 5.4: Průběh vytvoření šablony pro dynamický klíč.

5.3 Kombinovaná strategie

Kombinovaná strategie vychází ze spojení již existující TopN strategie a nové Rate Limit strategie. Hlavním účelem strategie bylo rozšířit TopN strategii, která zahazuje veškerý provoz blokované IP adresy. Kombinovaná strategie namísto zahození celkového provozu omezí IP adresu pouze částečně. Obecně platí, že Kombinovaná strategie je mírnější z pohledu zahazování adres, ale postihne jich více než obyčejná TopN strategie. Kombinovaná strategie tak lépe využije stanovený optimální limit – více se k limitu přiblíží. Přesnější přiblížení k optimálnímu limitu je zapříčiněno díky jemnějším blokováním, než k jakému dochází u TopN strategie.

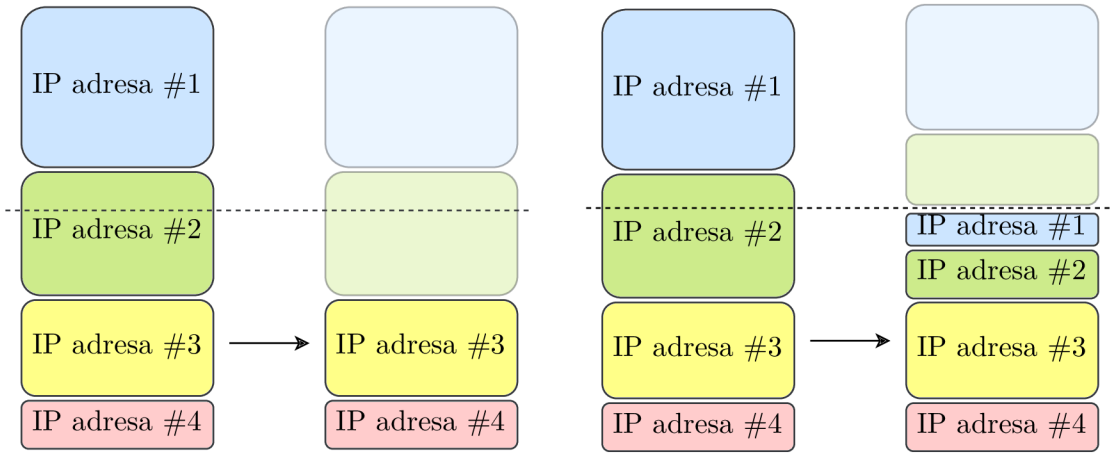
Nová strategie pracuje podobně jako předešlá TopN. Provoz jednotlivých pravidel se seřadí podle IP adres sestupně. Ovšem namísto blokování celé IP adresy se limitace provozu postupně zmírňuje s každou další adresou. Příklad blokování provozu Kombinovanou strategií obsahuje tabulka 5.2. Provoz v tabulce je tvořen šesti IP adresami seřazenými sestupně. Limitace první IP adresy začíná na 90 % provozu aktuálně blokované IP adresy. Celkový provoz je po zablokování 90 % první adresy snížen z 550 Mb/s na 370 Mb/s. Další adresa se limituje pouze o 0,9násobek z předešlého limitu, tedy 81 % z celkového provozu druhé IP

adresy. Provoz je po limitování druhé adresy opět snížen z 370 Mb/s na 248 Mb/s, což je stále více než optimální provoz, kterého chceme dosáhnout (200 Mb/s). Limit se s každou další adresou snižuje, dokud výsledný provoz po odečtení zablokovaného provozu není nižší nebo roven optimálnímu provozu. V našem případě je blokována ještě třetí nejobjemnější IP adresa. Po jejím limitování dosáhne celkový provoz hodnoty 175,6 Mb/s, čímž se splní požadavek na optimální provoz. Další adresy již nijak dále blokovány nejsou.

IP adresa	**Provoz**	**Limit**	**Původní celkový provoz**	**Nový celkový provoz**	**Optimální provoz**
147.220.101.3	200 Mb/s	90 %	550 Mb/s	370 Mb/s	200 Mb/s
78.103.22.18	150 Mb/s	81 %	370 Mb/s	248 Mb/s	
123.68.113.77	100 Mb/s	72,9 %	248 Mb/s	175,6 Mb/s	
154.60.140.127	50 Mb/s	0 %	175,6 Mb/s		
27.137.225.71	30 Mb/s	0 %			
135.40.236.45	20 Mb/s	0 %			

Tabulka 5.2: Blokování provozu Kombinovanou strategií.

Obrázek 5.5 vizuálně porovnává TopN strategii s novou Kombinovanou strategií. Na obrázku lze vidět, jakým způsobem se nová strategie dokáže více přiblížit optimálnímu provozu. U TopN strategie může dojít ke zbytečně velkému zahazování provozu i přesto, že je provoz daleko pod optimální hranicí. To bylo jedním z hlavních důvodů pro vytvoření nové strategie. Dalším důvodem byla samozřejmě mitigace celých toků, ať už se jedná o fragmentovaný provoz, nebo toky definované samotným uživatelem.



(a) Mitigace použitím TopN strategie.

(b) Mitigace použitím Kombinované strategie.

Obrázek 5.5: Porovnání strategií TopN a Kombinované strategie.

Kombinace obou strategií spočívá v tom, že se obdobně jako u Rate Limit strategie vytvoří šablona i amplifikačním pravidlům používajícím Kombinovanou strategii. Šablona obsahuje údaje o velikosti a odsazení položek klíče definovaných uživatelem v databázi. Strategie využívá stejné hash funkce jako Rate Limit strategie, vracející hodnotu uint32_t. Výsledná hodnota po vložení dynamického klíče do hash funkce je operací modulo (zbytek po celočíselném dělení) převedena na hodnoty v rozmezí 0-99 (u Rate Limit strategie tomu bylo 0 až maximální 32bitové nezaporné celé číslo). Tato hodnota je u zpracování paketů

porovnána s limitem jednotlivých IP adres (90 pro první adresu, 81 pro druhou atd.) a podle výsledku porovnání je adresa propuštěna, či zahozena. Strategie je opět vhodná jak pro fragmentovaný provoz, tak pro toky definované uživatelem.

Z hlediska implementace se Kombinovaná strategie příliš neliší od TopN strategie. Ve skutečnosti jsou téměř vždy využívány stejné funkce s tím, že se občas vyhodnocení liší kvůli odlišnému limitování. Kombinovaná strategie také využívá na rozdíl od TopN strategie šablony pro dynamický klíč. Tudíž bylo zapotřebí ošetřit odlišné chování při alokaci paměti v momentě nového útoku a stejně tak uvolnění paměti po skončení útoku. Nicméně základ je obdobný.

Při procesu blokování IP adres je využíváno struktury 5.3. Struktura uchovává pole indexů pravidel, pro které je IP adresa blokována. Druhou položkou je počet pravidel s blokovanou IP adresou. Momentálně je v Amplifikačním modulu nastaven maximální počet pravidel pro jednu blokovanou IP adresu na tři pravidla. Konstanta MAX_RULES tak odpovídá hodnotě tři. Poslední položka struktury byla přidána až s novou Kombinovanou strategií. Jedná se o samotný limit (threshold) pro danou IP adresu, tzn. do jaké míry se má daná IP adresa blokovat. Pro původní strategii TopN je threshold nastaven vždy na hodnotu 100. Tím je zajištěno, že při porovnávání thresholdu s výstupem hash funkce u zpracování paketu bude paket vždy zahozen tak, jak je u strategie žádoucí. Kombinovaná strategie ovšem threshold využívá pro určení míry, s jakou se má IP adresa zahazovat. Threshold první IP adresy je podle již popisované metody blokování 5.2 nastaven na hodnotu 90, druhé IP adrese odpovídá threshold 81 apod.

amp_block_rules_s	
rules_indexes[MAX_RULES]:	uint32_t
cnt:	size_t
threshold[MAX_RULES]:	size_t

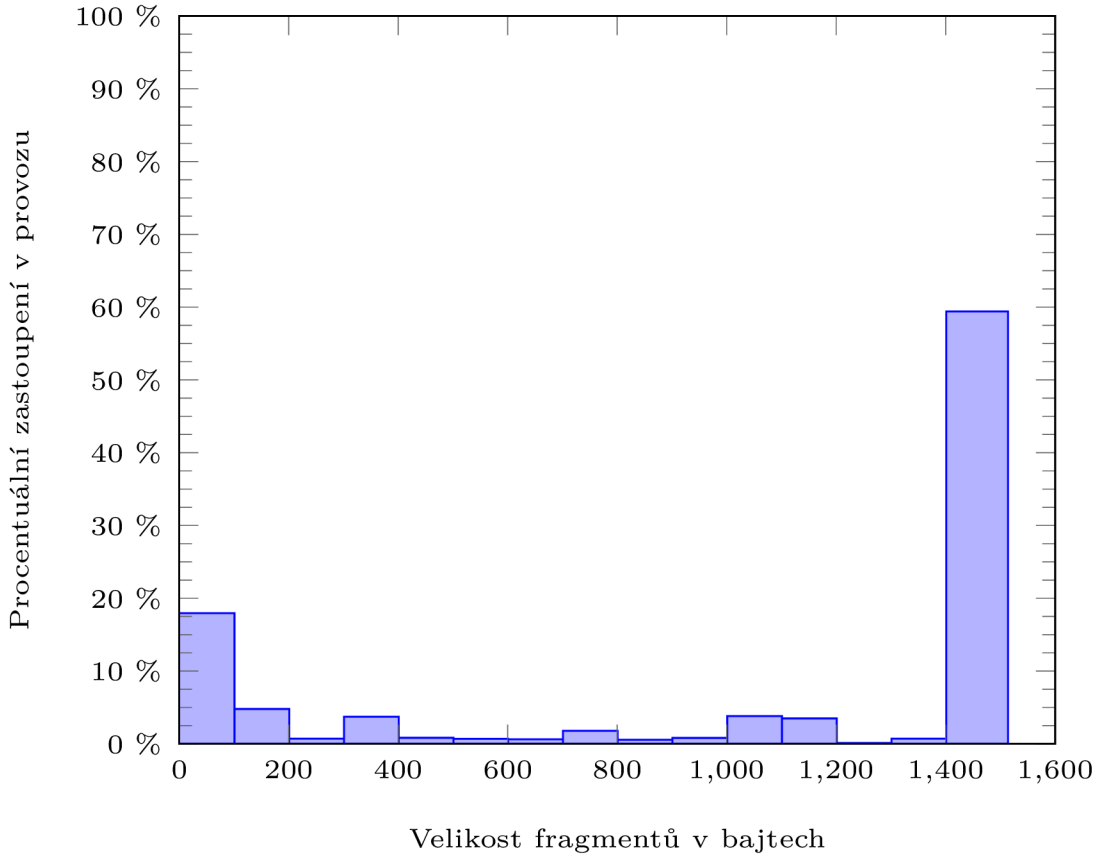
Tabulka 5.3: Struktura uchovávající informace o blokované IP adrese.

5.4 Analýza fragmentovaného provozu

Analýzu fragmentovaného provozu bylo zapotřebí provést proto, aby se ověřila myšlenka odvozování čísel portů z prvního fragmentu tak, jak činí Ports Inference modul. Jelikož modul nedokáže odvodit čísla dalším fragmentům bez příchodu prvního fragmentu, analýza provozu měla přiblížit, v jakém pořadí má fragmentovaný provoz tendenci přicházet a má-li modul vůbec smysl realizovat. Pro účely analýzy byl vytvořen skript v jazyce Python. Skript využívá programu Tcpdump[1] k filtraci fragmentů ze zachyceného síťového provozu uloženého do souboru. Dále skript používá knihovnu Dpkt[2]. Knihovna Dpkt slouží k rychlému parsování paketů. Po vyfiltrování čistě fragmentovaných paketů skript provádí analýzu. Hlavním cílem bylo zjistit, jak často se v reálném provozu stává, že první fragment nedorazí jako první z datagramu. A v případě, že k tomuto jevu dochází často, zvážit, zda Ports Inference modul vůbec pomůže v situaci filtrování fragmentovaného provozu. Skript si uchovává různé informace. Mezi tyto informace patří celkový počet datagramů, které jsou fragmentovány, počet jednotlivých fragmentů, skript také zjišťuje, z kolika fragmentů se datagramy skládají a jaké jsou jejich velikosti.

[1] https://www.tcpdump.org/
[2] https://dpkt.readthedocs.io/en/latest/

Velikosti fragmentů a jejich procentuální zastoupení z celkového počtu fragmentů znázorňuje obrázek 5.6. Graf je spjat se souborem zachyceným mezi sdružením CESNET a sdružením ACONET. O souboru lze najít bližší informace v tabulce 5.4. Z obrázku je zřejmé, že více než 50 % fragmentů dosahovalo velikosti větší než 1400 bajtů. Největší fragmenty se zároveň největším počtem výskytu měly 1514 bajtů. Celkově soubor takových fragmentů obsahoval 107 090. Dále si z obrázku můžeme povšimnout, že necelých 20 % fragmentovaného provozu zastupovaly fragmenty s velikostí do 100 bajtů, to znamená, že kdyby byla hodnota MTU pouze o něco málo větší, fragmentace by nebyla téměř potřeba. Velikosti od 100 do 1400 bajtů se vyskytovaly přibližně ve stejném množství, přičemž v rozmezí od 1200 do 1300 bajtů přišlo pouze 296 fragmentů.



Obrázek 5.6: Velikosti fragmentů a jejich procentuální zastoupení.

Počet fragmentů v datagramu nebyl nijak znázorněn, jelikož naprostá většina datagramů (přes 95 %) se skládala ze dvou až tří fragmentů. Datagramů, které obsahovaly více než deset fragmentů se objevilo pouhých 27.

Analýza probíhala na reálném provozu. Zdrojem dat byl převážně dataset obsahující anonymizovaný provoz z různých CAIDA vysokorychlostních monitorovacích nebo vysokorychlostních internetových linek [8]. Organizace CAIDA volně poskytuje své datasety pro výzkumné účely. Jelikož jsou data anonymizovaná, IP adresy jsou hashované a payload paketu kompletně chybí. Nicméně jako zdroj pro analýzu fragmentovaného provozu takový zdroj dat neztrácí hodnotu a lze jej plnohodnotně využít. Z důvodu absence payloadu fragmentů však nelze přímo určit velikosti jednotlivých fragmentů, a proto nejsou

velikosti vizualizovány v grafu. Výsledek analýzy znázorňuje tabulka 5.4 (první polovina). Můžeme si povšimnout, že se celkový provoz skládá z 33 milionů paketů. Tento údaj byl zjištěn za pomoci programu Capinfos[3], jenž vypisuje informace o souborech se zachyceným provozem, právě jako je počet paketů, průměrná velikost paketu apod. Celkově provoz obsahoval 91 174 fragmentů. Z tohoto počtu fragmentů bylo 44 077 prvních fragmentů, což představuje přibližně 48 % z fragmentovaného provozu. Tudíž platí, že většina datagramů je složena pouze ze dvou fragmentů. Z těchto necelých sta tisíc fragmentů se stalo, že 7447krát dorazil prostřední fragment dříve, než fragment první (přibližně 8 % ze všech fragmentů).

Analyzován byl také soubor s provozem zachyceným mezi sdružením CESNET a rakouským sdružením ACONET. Linka měla kapacitu 2x10 Gb/s [5]. Zachycené informace opět znázorňuje tabulka 5.4 (druhá polovina). Provoz obsahoval celkově 509 milionů paketů. Prvních fragmentů se vyskytlo 98 473 z celkových 250 434 fragmentů. První fragmenty tak zaujímají necelých 40 % ze všech fragmentů a opět platí, že naprostá většina datagramů se skládá ze dvou až tří fragmentů, přičemž se vyskytlo podstatně více datagramů obsahující tři fragmenty, než tomu bylo v předchozím případě. Prostředních fragmentů, které dorazily dříve než první fragment, se vyskytlo pouze 633.

	CAIDA		CESNET&ACONET	
Paketů celkem	33 M	100 %	509 M	100 %
Fragmentů	91 174	0,276 %	250 434	0,05 %
Prvních fragmentů	44 077	48,343 %	98 473	39,32 %
Prostředních před prvním	7 447	8,168 %	633	0,25 %

Tabulka 5.4: Analýza fragmentovaného provozu.

Z tabulky 5.4 je zřejmé, že z obou zachycených souborů tvoří fragmentovaný provoz pouze mizivé množství. Nicméně lze z dosažených hodnot určit účinnost (uvedena v tabulce 5.5) nového modulu a vyvodit tak závěr o jeho smysluplnosti. Úspěšností modulu se myslí v kolika procentech případů přidělí Ports Inference modul čísla portů správně. U souboru z datasetu organizace CAIDA dosáhne modul úspěšnosti přibližně 83 %. Zajímavé ovšem je, že druhý provoz, jenž obsahuje více než dvojnásobek fragmentovaného provozu, nedosahuje takového počtu prostředních fragmentů, které dorazily před prvním fragmentem. Zde dosahuje modul účinnosti 99,3 %. Je tedy patrné, že velmi záleží na vícero faktorech, jako jsou například množství nastavených Don't Fragment bitů, topologie sítí, přes které fragmenty prochází apod. Veškeré soubory z datasetu poskytovaného organizací CAIDA dosahují přibližně stejných výsledků jak bylo zjištěno výše (účinnost modulu okolo 80 %). I přesto, že účinnost modulu, které bylo dosaženo při odvozování portů datasetům poskytovaných organizací CAIDA byla nižší, než jakých dosáhl modul u provozu ze sdružení CESNET, účinnost činí více jak 80 %. Navržená heuristika proto dává smysl a rozhodl jsem se ji implementovat a následně integrovat do Čističky.

	CAIDA	CESNET&ACONET
Fragmentované datagramy	44 077	98 473
Prostřední před prvním	7 447	633
Účinnost modulu	83,1 %	99,3 %

Tabulka 5.5: Účinnost Ports Inference modulu.

[3] https://linux.die.net/man/1/capinfos

5.5 Ports Inference modul

Ports Inference je zcela novým modulem integrovaným do Čističky. Jedná se o heuristiku, jejímž hlavním účelem je odvození čísel portů fragmentovaným paketům. Modul je zapojen ještě před vyhodnocením pravidel probíhajících například v amplifikačním modulu tak, aby odvozené porty mohly být součástí vyhodnocení. První fragment z datagramu obsahuje čísla portů vždy. Modul má za úkol přiřadit čísla portů i dalším příchozím fragmentům z téhož datagramu. Přiřazení čísel portů lze využít především při filtraci provozu. Obě z nově implementovaných strategií využívají dynamický klíč, jehož součástí mohou být právě čísla portů.

Modul využívá hashovací tabulku, která je komponentou už dříve používanou v Čističce. Po příchodu prvního fragmentu z datagramu vytvoří modul klíč do hashovací tabulky spojením zdrojové IP adresy, cílové IP adresy a položky Identification z IP hlavičky. Modul klíčem přistupuje do hashovací tabulky a ukládá do ní data z paketu. Ukládaná data obsahují zdrojový port, cílový port a časovou známku paketu. Uložené porty budou sloužit pro odvození portů následujícím fragmentům z datagramu. Časová známka paketu slouží pro ověření platnosti dat v tabulce. U následujícího fragmentu se modul pokusí pomocí klíče složeného ze stejných položek přistoupit k hashovací tabulce. Jsou-li v tabulce pod daným klíčem uložena data prvního fragmentu, přiřadí modul fragmentu stejná čísla portů. Kromě přiřazení portů modul upraví časovou známku v tabulce na hodnotu právě zpracovávaného fragmentu, čímž prodlouží platnost dat v tabulce. Jelikož se jedná pouze o heuristiku, nefunguje přiřazení portů v případě, kdy je modulem zpracováván následující fragment před prvním. V takové situaci se modul pokusí načíst data z hashovací tabulky, ale pod stejným klíčem zatím nejsou v tabulce uložena žádná data. Analýze fragmentovaného provozu a s tím souvisejícímu pořadí fragmentů v reálném provozu se věnuje předchozí sekce 5.4. Může nastat situace, kdy se modul pokusí načíst data a zjistí, že časová známka získaných dat není platná. V takovém případě odstraní z hashovací tabulky veškerá data pod tímto klíčem.

I přesto, že se ke zpracování modulem dostanou pouze fragmentované pakety, bylo zapotřebí, aby modul příliš nezpomalil celý proces zpracování paketu. Proto bylo nutné přistupovat k hash tabulce co nejméně. Modul využívá funkcí `fht_get_data_insert_empty()` a `fht_get_data()` již integrovaných v Čističce pro zápis a čtení do hash tabulky. Odvozování čísel portů modulem by se dalo zjednodušeně přiblížit pseudokódem v algoritmu 3.

Algoritmus nejprve porovnává, zda se jedná o první fragment a je potřeba ukládat jeho čísla portů do tabulky (řádek 1). V momentě, kdy je fragment první z datagramu, je volána funkce `fht_get_data_insert_empty()`. Samotná funkce nezajišťuje zkopírování dat do tabulky (pouze klíče), ale navrátí ukazatel na paměť, kam je možné data zkopírovat. Jako klíč jsou použity zdrojová a cílová IP adresa společně s položkou Identification (řádek 2). Následně jsou do paměti v tabulce postupně kopírovány hodnoty z právě zpracovávaného paketu, včetně jeho časové známky (řádky 3, 4, 5).

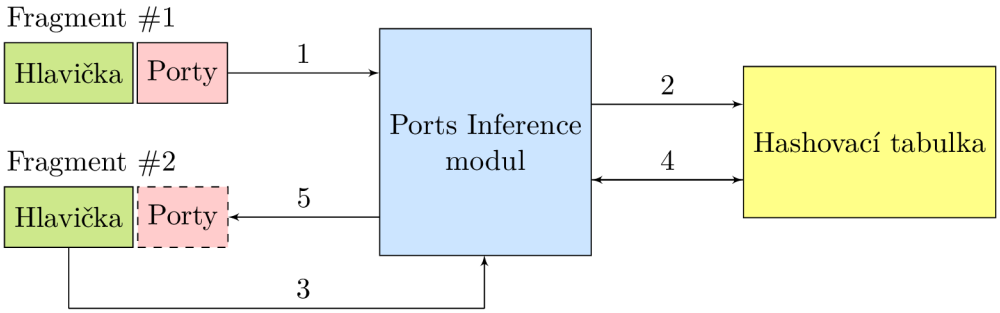
Jestliže počáteční podmínka nebyla splněna a momentálně zpracovávaný fragment není první, volá se funkce `fht_get_data()`. Funkce navrací ukazatel na nalezená data pod daným klíčem (řádek 7). Samozřejmě existuje možnost, kdy žádná data v tabulce nebyla, protože první fragment datagramu zatím do Čističky ke zpracování nedorazil. Tento stav je ošetřen podmínkou a modul ukončuje svou činnost pro tento fragment, jemuž nelze přidělit čísla portů (řádek 8 a 9). Po ověření přítomnosti dat v tabulce je ještě nutno zjistit, zda se časová známka fragmentu a časová známka dat z tabulky neliší o hodnotu větší, než je nastaven interval pro vypršení platnosti dat (řádek 10). Splňuje-li časová známka interval,

dochází k přiřazení portů fragmentu (řádek 11 a 12). Kromě přiřazení portů se také upraví časová známka právě čtených dat v tabulce (řádek 13). Poslední možnost, kdy modul nedosáhne zdárného konce, nastane pokud časový interval dat v tabulce vypršel. V tu chvíli modul odstraní data pod požadovaným klíčem (řádek 15).

Algorithm 3: Odvozování portů v PI modulu.

```
1  if zpracovavany_paket je první fragment then
2      fht_get_data_insert_empty(hash_tabulka, klic, (void **)&data);
3      data.zdrojovy_port ← zpracovavany_paket.zrdojovy_port;
4      data.cilovy_port ← zpracovavany_paket.cilovy_port;
5      data.casova_znamka ← zpracovavany_paket.casova_znamka;
6  else
7      data ← fht_get_data(hash_tabulka, klic);
8      if !data then
           // data nebyla v tabulce nalezena
9          Návrat z modulu;
10     if časová známka v hash_tabulka je validní then
11         zpracovavany_paket.zdrojovy_port ← data.zdrojovy_port;
12         zpracovavany_paket.cilovy_port ← data.cilovy_port;
13         data.casova_znamka ← zpracovavany_paket.casova_znamka;
14     else
15         Odstraň data z hash_tabulka;
```

Na obrázku 5.7 je zachycena situace, kdy se zpracovává počáteční fragment (krok 1). S využitím PI modulu uloží za pomocí klíče čísla portů společně s časovou známkou do hashovací tabulky (krok 2). Následující fragment ze stejného datagramu se při zpracování pokusí získat chybějící čísla portů (krok 3). PI modul se dotáže hashovací tabulky, zda jsou v ní pod konkrétním klíčem data. Jestliže ano, navrátí data (čísla portů) zpět PI modulu (krok 4) a upraví se časová známka v tabulce. Posledním krokem je přiřazení chybějících portů právě zpracovávanému fragmentu (krok 5).



Obrázek 5.7: Návrh Ports Inference modulu.

Modul je inicializován společně s amplifikačním modulem. V rámci inicializace je potřeba alokovat paměť pro strukturu obsahující konfiguraci modulu a hashovací tabulku. Struktura s konfigurací obsahuje informaci o stavu modulu, to znamená zda je modul aktivní, nebo vůbec není spuštěn. Dále se ve struktuře nachází velikost hashovací tabulky. Posledním elementem struktury je časový interval, po který mají být položky v hashovací tabulce validní a data lze považovat za platná. Jak velikost tabulky, tak časový interval jsou konfigurovány uživatelem při spuštění Čističky.

5.6 Shrnutí

Kapitola 5 se zabývala návrhem a implementací nově vytvořených strategií a nového Ports Inference modulu. Veškerá nová funkčnost je zaměřena na fragmentovaný provoz, respektive na jeho efektivnější mitigaci a zlepšení práce s fragmentovaných provozem v podobě odvození čísel portů a možné následné filtraci. V rámci bakalářské práce byly vytvořeny dvě nové strategie – Rate Limit a Kombinovaná strategie. Rate Limit strategie využívá k limitování pouze míru provozu. Kombinovaná strategie kombinuje novou Rate Limit strategii a existující TopN strategii. Obě strategie jsou navrženy tak, aby byly schopny efektivně blokovat fragmentovaný provoz. Limitace fragmentovaného provozu tkví ve vytvoření klíče definujícího tok. Vzhledem k omezení klíče pouze na fragmentovaný provoz byl později navržen dynamický klíč. Dynamický klíč se skládá z kombinace vybraných položek, které si sám určí uživatel a definuje tak svůj tok. Díky dynamickému klíči má uživatel možnost limitovat i jiné toky než pouze fragmentovaný provoz, dle své potřeby.

Kapitola se také věnuje analýze fragmentovaného provozu. Výsledky analýzy byly vyhodnoceny, a neboť neukázaly závažný problém v podobě příliš častého či nesprávného pořadí fragmentovaných paketů, umožnily vytvoření nového Ports Inference modulu. Ports Inference modul slouží k odvození čísel portů prostředním a posledním fragmentům, jež čísla portů postrádají. Jak obě nové strategie, tak Ports Inference modul byly navrženy, implementovány a integrovány do Čističky v rámci této práce.

Původní Amplifikační modul v Čističce podporuje hardwarovou akceleraci, to znamená využití speciálního hardwarového zařízení k efektivnějšímu výpočtu, než jakého by modul mohl dosáhnout pouze za pomocí softwarové činnosti. Amplifikační modul uplatňuje hardwarovou akceleraci například při blokování IP adres. Modul je tedy schopen filtrovat IP adresy přímo v hardwaru. Užitečné informace je možné do softwarové části předávat například v podobě UH hlavičky. Tato hlavička je v Čističce reprezentována strukturou a uchovává důležitá data o paketu, jako jsou například IP adresy paketu, čísla portů, protokol apod. Bohužel zmíněná efektivní řešení musejí být v momentě použití nových strategií v konfiguračním souboru vypnuta, jelikož funkčnost by vyžadovala jisté hardwarové změny. Na příklad UH hlavička v Čističce vůbec neobsahuje položku Identification, která je u strategií hojně využívána. Hodnota Identification je často součástí klíče definujícího tok, a to i v případě filtrování fragmentovaného provozu. Jednou z klíčových činností pro správnou funkčnost hardwarové akcelerace u nových strategií by tedy bylo přidání položky Identification do UH hlavičky. Tento krok by využil také Ports Inference modul, který používá položku Identification jako klíč do hashovací tabulky. Tudíž Ports Inference modul taktéž vyžaduje softwarové filtrování. Další z možných úprav, jimiž lze dosáhnnout hardwarové akcelerace v mitigaci, je možnost úpravy limitace jednotlivých IP adres. Poněvadž stávající strategie TopN limituje veškerý provoz blokované IP adresy, není v hardwaru řešeno pouze částečné limitování provozu tak, jak provádí Kombinovaná strategie. Zrychlení Kombinované strategie by mohlo být docíleno přídáním možnosti vypnutí zahazování veškerého provozu, ale pouze jeho částečnou limitaci. Implementace Kombinované strategie do hardwaru, přesněji řečeno implementace pouze částečné limitace provozu v hardwaru je poměrně jednoduše realizovatelná. Postačí spočítat hash hodnotu a následně provést porovnání s thresholdem, který byl nastaven v softwaru.

Následující kapitola 6 se věnuje vyhodnocení výkonnosti a změření implementace.

Kapitola 6

Vyhodnocení výsledků

Kapitola se zabývá vyhodnocením výkonnosti nově přidaných funkcí, aby ověřila jejich kvalitu a přínos do Čističky. K měření nám poslouží speciální software zvaný Spirent TestCenter[1]. Spirent slouží ke generování libovolných paketů a umožňuje nastavení různých vlastností. Díky Spirentu jsme schopni měřit latenci a propustnost, klíčové prvky, kterými ověříme kvalitu dosažených výsledků. Stroj, na kterém byla spuštěna Čistička, měl parametry znázorněné v tabulce 6.1.

CPU	Intel(R) Xeon(R) Silver 4114 CPU @ 2.20 GHz
OS	Scientific Linux 7.8 (Nitrogen)
RAM	12 * 8192MB (2400MT/s)

Tabulka 6.1: Parametry zařízení.

Konfigurace byla rozděleno na pět variant, jednotlivé varianty (reprezentující sloupce) a s nimi i jejich položky (řádky) jsou k vidění v tabulce 6.2. První položka hw-filter-enable povoluje filtraci za pomocí hardware. Druhá hodnota sw-forwarding znamená směrování všech paketů skrze software, hardwarové filtrování není využito. Full packets vyžaduje přijetí pouze celých paketů ze systémové části firmwaru. Nastavení hodnoty PI modul povoluje Ports Inference modul. Poslední dvě položky určují, dle které strategie se ve variantě mitiguje. Jak obě strategie, tak Ports Inference modul vyžadují softwarovou filtraci z důvodů zmiňovaných v sekci 5.6. Například varianta PI modul z tabulky znamená filtrování za pomocí softwaru s povolením Ports Inference modulu. Varianta SW zpracování naopak znamená filtrování pouze za pomoci softwaru, ale s vypnutým Ports Inference modulem.

	HW zprac.	SW zprac.	PI mod.	RL strategie	Komb. strategie
hw-filter-enable	✓	✗	✗	✗	✗
sw-forwarding	✗	✓	✓	✓	✓
full-packets	✗	✓	✓	✓	✓
PI modul	✗	✗	✓	✗	✗
Rate limiting	✗	✗	✗	✓	✗
Komb. strategie	✗	✗	✗	✗	✓

Tabulka 6.2: Varianty nastavení konfiguračního souboru při testování.

[1] https://www.spirent.com/products/testcenter

Propustnost byla měřena všem zmiňovaným variantám na fragmentovaném provozu. Generován byl provoz obsahující dva datagramy. Oba datagramy byly složeny ze tří fragmentů, každý s fixní délkou 512 bajtů. Čistička využívala osm jader procesoru. Výsledky viz tabulka 6.3. U první varianty, ve které dochází k filtraci v hardwaru byla naměřena stoprocentní propustnost, i při maximálním vstupním provozu 100 Gb/s, paketová propustnost zde dosáhla 23,5 milionů paketů za sekundu. Další variantou filtrující provoz v softwarové části, bez jakékoliv činnosti hardwarové filtrace, byla naměřena propustnost 22,779 Gb/s a 5,615 milionů paketů za sekundu. Jelikož zbylé varianty vyžadují aktivní softwarovou filtraci, poslouží nám tato varianta jako výchozí, se kterou budeme dále porovnávat výsledky propustnosti nových heuristik, ale i výsledky latence.

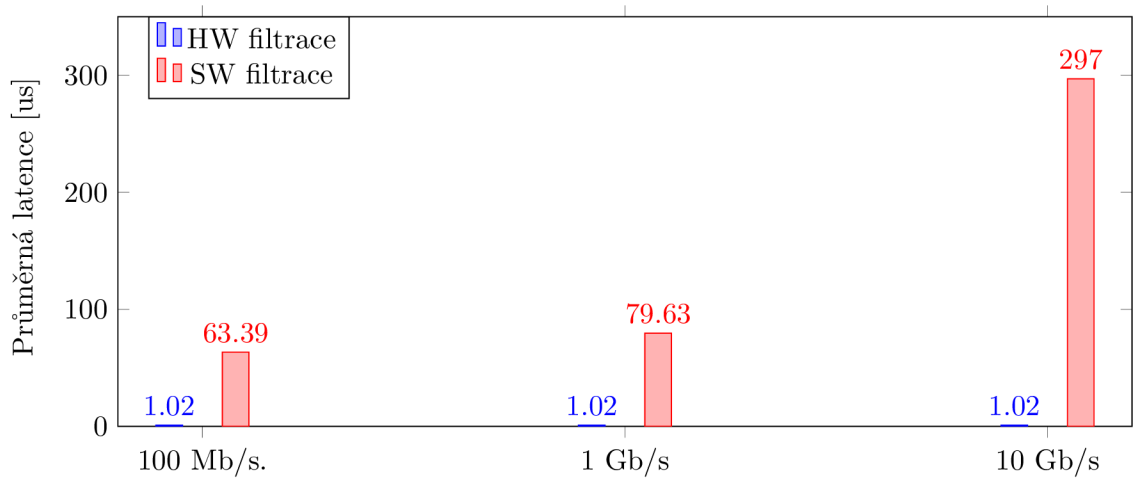
Třetí varianta, využívající modul pro odvození čísel portů dosahuje nejnižší propustnosti ze všech naměřených výsledků. Propustnost zde dosahuje 13,755 Gb/s, s celkovým počtem paketů 3,336 milionů za sekundu. Ačkoliv byl při vývoji modulu kladen důraz na co nejmenší počet přístupů do hashovací tabulky, je nutné přistoupit do tabulky vždy alespoň jednou u každého fragmentu, což zjevně zpomalí modul natolik, že k dosažení vyšší propustnosti bude vyžadována hardwarová akcelerace PI modulu.

Nová strategie využívající rate-limitingu nabývá propustnosti 22,079 Gb/s (5,167 milionů paketů za sekundu). Strategie tak docílila nejvyšší propustnosti z nově intergrovaných heuristik. Narozdíl od Ports Inference modulu se liší pouze o přibližně 0,7 Gb/s od softwarové filtrace. Poslední Kombinovaná strategie dosahovala při měření propustnosti výsledku 19,105 Gb/s (4,488 milionů paketů za sekundu).

	Hw zprac.	SW zprac.	PI modul	Rate Limit	Komb. strat.
Bitová propustnost	>100 Gb/s	24,779 Gb/s	13,755 Gb/s	22,079 Gb/s	19,105 Gb/s
Paketová propustnost	>23,5 M/s	5,615 M/s	3,336 M/s	5,167 M/s	4,488 M/s

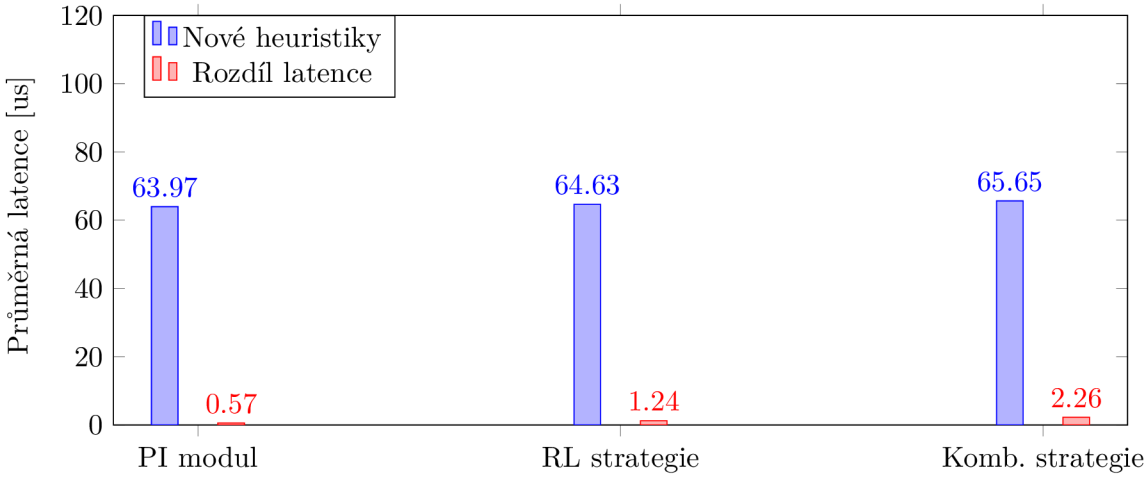
Tabulka 6.3: Propustnost všech měřených variant při velikosti fragmentu 512 bajtů.

Měření latence probíhalo nad generováním provozu s různými rychlostmi, opět pro každou variantu zvlášť. Cílem testování bylo odeslat šest milionů paketů. Veškeré pakety byly fragmentované. Rychlost generování provozu dosahovala hodnot 100 Mb/s, 1 Gb/s a 10 Gb/s. První graf 6.1 porovnává latenci při filtrování v hardwaru s filtrací v softwaru. Můžeme si povšimnout velmi nízké latence při filtraci v hardwaru, která dosahuje 1,02 us. Její latence se zvyšuje pouze v řádech tisíciny mikrosekund, tudíž není změna v grafu viditelná. Naopak softwarová filtrace se z poměrně nízké latence 63,39 us postupně navýšila na hodnotu 297 us.



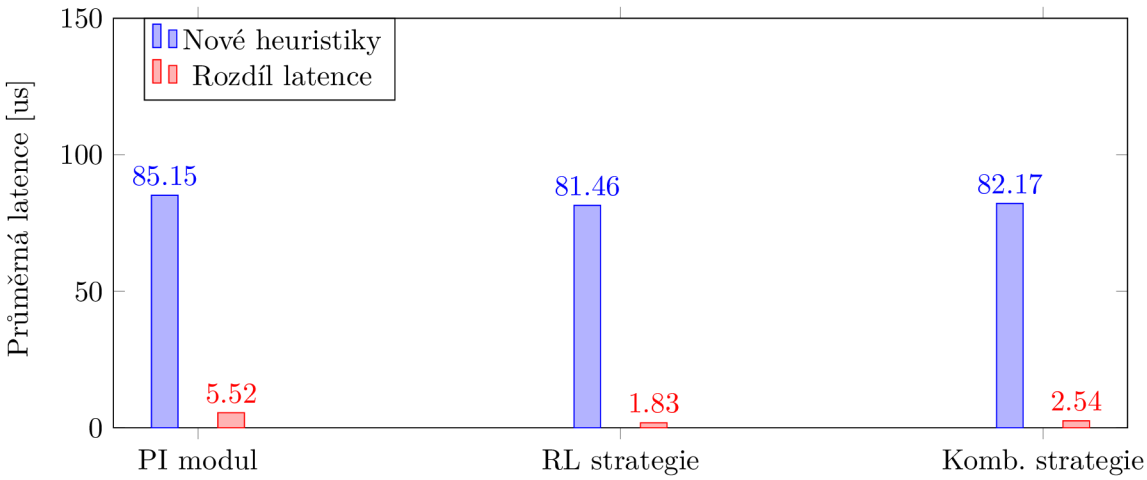
Obrázek 6.1: Průměrná latence HW a SW filtrace.

U prvního měření nových heuristik byla rychlost generování provozu nastavena na 100 Mb/s. Výsledky měření lze vidět v grafu 6.2. V grafu jsou modrou barvou znázorněny latence nových strategií a modulu pro odvození portů. U každé z nich je poté červený sloupec, který značí rozdíl latencí aktuální heuristiky oproti výchozí softwarové filtraci. Můžeme si tedy všimnout, že latence PI modulu se navýšila oproti softwarové filtraci o pouhých 0,57 us, RL strategie o 1,24 us a Kombinovaná strategie o 2.26 us. Z pohledu latence tedy zatím nedošlo k žádnému závažnému navýšení ani u jedné z heuristik.



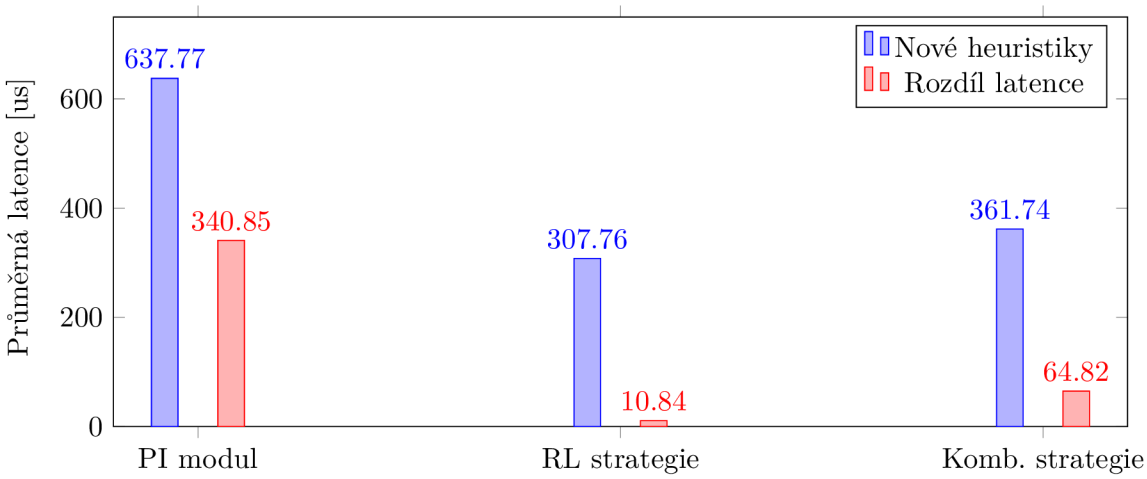
Obrázek 6.2: Průměrná latence při rychlosti 100 Mb/s.

Druhá z rychlostí při měření latence dosahovala až 1 Gb/s. Z grafu 6.3 si můžeme všimnout drobného navýšení latence u Ports Inference modulu oproti zbylým variantám. Ports Inference modul tedy svou latencí převýšil obě strategie oproti přechozímu měření, kde dosahoval nejnižší hodnoty. Latence se oproti softwarovému filtrování pro PI modul navýšila na 5,52 us, zatímco u rate-limitingu a Kombinované strategie je rozdíl 1,83 us a 2,54 us.



Obrázek 6.3: Průměrná latence při rychlosti 1 Gb/s.

Ve třetím měření s rychlostí až 10 Gb/s došlo oproti minulým výsledkům k poměrně výrazným změnám. Jak lze vidět v grafu 6.4, latence SW filtrování, i všech heuristik narostly až do řádů stovek mikrosekund. Obě strategie jsou mírně pomalejší oproti klasickému softwarovému filtrování, ale nejedná se o žádný prudký nárůst a kopírují latenci SW filtrace. PI modulu se ovšem navýšila latence na zhruba dvakrát vyšší hodnotu než je tomu u SW filtrace. Latence PI modulu převýšila SW filtraci o 340,85 us, rate-limiting o pouhých 10,84 us a Kombinovaná strategie o 64,82 us.



Obrázek 6.4: Průměrná latence při rychlosti 10 Gb/s.

Z naměřených výsledků popisovaných v kapitole je jednoznačně patrná výhoda hardwarové akcelerace. U žádné z testovaných rychlostí nedošlo u hardwarové filtrace k vizuálnímu nárůstu latence ani ke ztrátě v propustonosti. U softwarové filtrace narůstá latence s každým novým měřením. Proto nebyly nové heuristiky porovnávány s HW filtrací, ale s filrací softwarovou, kterou využívají jak obě strategie, tak PI moudul. Porovnáme-li strategie se SW filtrací zjistímě, že se příliš neliší. Samozřejmě lze na latenci sledovat nárůst zapříčiněný především počítáním hash hodnoty pro každý paket a kopírováním položek dynamického klíče do paměti. Ports Inference modul však trpí vyšší latencí podstatně více. Modul do-

sahoval při rychlostech vyšších než 10 Gb/s až dvojnásobné latence než jaké docílila SW filtrace. Vysoká latence je patrně způsobena častou prací s hashovací tabulkou nutnou pro každý paket. Propustnost byla nejhorší u PI modulu, dosahuje hodnoty 13,755 Gb/s než začalo docházet k nekontrolovatelnému zahazování paketů. Propustnost Rate Limit strategie činí 22,079 Gb/s a Kombinované strategie 19,105 Gb/s. Propustnost samotné SW filtrace dosahuje hodnoty 23,779 Gb/s. Velikost generovaných fragmentů měla fixní délku 512 bajtů.

Se zmíněnými problémy s latencí a propustností by pomohlo využití hardwarové akcelerace, jak je popsáno v sekci 5.6. Zejména přidání položky Identification do UH hlavičky a paralelizace výpočtu hash funkcí by pomohlo ke snížení latence jak novým strategiím, tak i Ports Inference modulu. Využití paralelizace výpočtu se nabízí i v použití vektorových instrukcí procesoru a s tím související optimalizace hash funkce. Případná vektorizace totiž probíhá plně na straně kompilátoru a není jisté zda jsou tyto instrukce využity.

Kapitola 7

Závěr

Smyslem této bakalářské práce bylo vytvoření nových strategií potlačujících DDoS útoky, jež zneužívají fragmentaci paketů, a následně integrovat heuristickou metodu pro odvození čísel portů fragmentovaným paketům. Splnění úkolů vyžadovalo seznámení se s problematikou útoků typu odepření služby a také se zařízením vyvíjeným sdružením CESNET, které se zabývá ochranou proti těmto útokům. K úspěšnému dokončení bylo také potřeba nastudovat princip samotné fragmentace provozu. Úsek věnující se fragmentaci obsahuje mimo jiné příklad algoritmu fragmentace a defragmentace, popisuje problémy vznikající s fragmentací provozu a následně možnosti, jimiž lze fragmentaci předcházet. Podstatná část práce se zabývá návrhem a implementací nových mitigačních strategií, analýzou charakteristických vlastností fragmentovaného provozu a novým modulem pro odvození čísel portů.

Potlačení fragmentovaných útoků bylo dosaženo především díky limitování celých toků definujících fragmentovaný provoz. Tímto se odlišují nově vytvořené strategie od obyčejných metod, kde dochází k odlišnému vyhodnocení pro jednotlivé pakety v rámci celého datagramu. Omezením či propuštěním celých toků se předchází porušení datagramu, což u fragmentovaného provozu znamená nutnost retransmise celého datagramu. V rámci bakalářské práce vznikly dvě nové strategie. První strategie Rate Limit, limitující čistě na základě míry provozu. Druhá implementovaná strategie vznikla kombinací nové strategie Rate Limit s již existující strategií TopN. Kombinací dochází k mírnějšímu, avšak efektivnějšímu omezení IP adres s největší mírou provozu, a tím pádem k lepšímu využití optimálního provozu. Pro účely vylepšení obou nových strategií byl také navržen dynamický klíč, jehož přínos spočívá v definování toků na základě různých položek uživatelem. Uživatel tak není omezen pouze na limitování fragmentovaného provozu, ale sám si podle dostupných položek určuje toky, dle kterých chce provoz limitovat.

Za účelem filtrace fragmentovaného provozu byl implementován nový modul pro odvození čísel portů fragmentovaným paketům, jež tato čísla postrádají. Jelikož se jedná pouze o heuristiku, neumí si modul poradit se situací, dorazí-li prostřední fragment dříve než první. Takovým fragmentům nelze čísla portů přidělit. Proto byla před samotnou implementací modulu provedena analýza reálného provozu. Na základě analýzy bylo potvrzeno, že má smysl vyvíjet nový heuristický modul, neboť dosažené výsledky obavy nenaplňují. Jak nové strategie, tak heuristický modul již byly v rámci bakalářské práce integrovány do zařízení zvané DDoS Protector vyvíjené sdružením CESNET.

Poslední kapitola se věnuje měření nově vytvořených funkcionalit. Měření testovalo latenci a propustnost s různou rychlostí generování provozu. U obou strategií byla latence pouze mírně vyšší než u samotné softwarové filtrace. Největším problémem byla naměřená latence Ports Inference modulu, která dosahovala při posledním měření až dvojnásobné

latence oproti měření bez využití modulu. Ports Inference dosahoval z nově vytvořených heuristik nejhorších výsledků také v propustnosti. Nové strategie dosahovaly o poznání lepších výsledků.

Kritickou operací je především výpočet velkého množství hashovacích funkcí. Jedním z možných pokračování práce tak může být hardwarová akcelerace navržených strategií a modulu pro odvozování čísel portů. Hardwarovou implementací je možné výpočet hashovacích funkcí vhodně paralelizovat, podstatně snížit latenci a dosáhnout tak vyšší propustnosti při zpracování síťového provozu. Zařízení DDoS Protector využívá síťovou kartu s FPGA kromě distribuce síťového toku na jádra CPU také k akceleraci již existující TopN strategie pro limitaci vybraných zdrojů nežádoucího provozu. Nové strategie vytvořené v rámci bakalářské práce je možné v budoucnu díky vhodnému návrhu obdobně implementovat v hardwaru s využitím technologií FPGA.

Literatura

[1] *Network Access Layer* [online]. Dec 2014. Dostupné z: https://darkness19935.wordpress.com/2014/12/01/network-access-layer-2/.

[2] *Amplifikační modul*. Naposledy navštíveno 7. 5. 2020, (online). Dostupné z: https://redmine.liberouter.org/projects/ddos-protector/wiki/Amplifika%C4%8Dn%C3%AD_modul.

[3] *DDoS Protector*. Naposledy navštíveno 7. 5. 2020, (online). Dostupné z: https://www.liberouter.org/technologies/ddos-protector/.

[4] *Synflood modul*. Naposledy navštíveno 7. 5. 2020, (online). Dostupné z: https://redmine.liberouter.org/projects/ddos-protector/wiki/SYN_flood_modul.

[5] *Topologie sítě CESNET2*. Naposledy navštíveno 9. 5. 2020, (online). Dostupné z: https://www.cesnet.cz/sluzby/pripojeni/topologie/.

[6] Adam, O. *IP fragmentation attacks – how do they work?* [online]. Feb 2018. Dostupné z: https://www.link11.com/en/blog/threat-landscape/ip-fragmentation-attacks-how-do-they-work/.

[7] Alienor. *The Network Layers Explained [with examples]* [online]. Nov 2018. Dostupné z: https://www.plixer.com/blog/network-layers-explained/.

[8] CAIDA. *The CAIDA Anonymized Internet Traces Dataset (April 2008 - January 2019)*. Naposledy upraveno: 2. 12. 2019. Dostupné z: https://www.caida.org/data/passive/passive_dataset.xml.

[9] Chavan, K. *What is Payload in Computer Network?* [online]. Březen 2019. Dostupné z: https://www.tutorialspoint.com/what-is-payload-in-computer-network.

[10] Corero. *DDoS Attack Types: Glossary of Terms*. Naposledy navštíveno 14. 4. 2020, (online). Dostupné z: https://www.corero.com/blog/glossary/#Fragmented%20ACK%20Flood.

[11] DDOS Guard. *HTTP Fragmentation (Fragmented HTTP Flood, Nuke)*. Naposledy navštíveno 20. 4. 2020, (online). Dostupné z: https://ddos-guard.net/en/terminology/attack_type/http-fragmentation-fragmented-http-flood-nuke.

[12] Deering, S. a Hinden, R. *Internet Protocol, Version 6 (IPv6) Specification* [IETF]. Prosinec 1998. Dostupné z: https://tools.ietf.org/html/rfc2460.

[13] Eddy, W. *TCP SYN Flooding Attacks and Common Mitigations*. Srpen 2007. Dostupné z: https://tools.ietf.org/html/rfc4987.

[14] GigeNET. *The different types of DDoS attacks and their definitions*. Naposledy navštíveno 14. 4. 2020, (online). Dostupné z: https://www.gigenet.com/ddos-attack/denial-of-service-definitions/.

[15] Imperva. *IP Fragmentation Attack*. Naposledy navštíveno 20. 4. 2020, (online). Dostupné z: https://www.imperva.com/learn/application-security/ip-fragmentation-attack-teardrop/.

[16] Imperva. *Booters, Stressers and DDoSers*. Naposledy navštíveno 7. 5. 2020, (online). Dostupné z: https://www.imperva.com/learn/application-security/booters-stressers-ddosers/.

[17] Kent, C. A. a Mogul, J. C. Fragmentation Considered Harmful. *SIGCOMM Comput. Commun. Rev.* New York, NY, USA: Association for Computing Machinery. leden 1995, roč. 25, č. 1, s. 75–87. Dostupné z: https://doi.org/10.1145/205447.205456. ISSN 0146-4833.

[18] Keshariya, A. a Foukia, N. DDoS Defense Mechanisms: A New Taxonomy. In: Garcia Alfaro, J., Navarro Arribas, G., Cuppens Boulahia, N. a Roudier, Y., ed. *Data Privacy Management and Autonomous Spontaneous Security*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, s. 222–236.

[19] Kuka, M., Vojanec, K., Kučera, J. a Benáček, P. Accelerated DDoS Attacks Mitigation using Programmable Data Plane. In: *2019 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS)*. 2019, s. 1–3.

[20] Kurose, J. F. a Ross, K. W. *Computer Networking: A Top-Down Approach, Seventh Edition*. 7. vyd. Pearson Education, 2017. ISBN 978-0-13-359414-0.

[21] Le, J. *The 4-Layer Internet Model Network Engineers Need to Know* [online]. Dec 2017. Dostupné z: https://medium.com/@james_aka_yale/the-4-layer-internet-model-network-engineers-need-to-know-e78432614a4f.

[22] NFWARE. *Limitations of IPv4 and IPv4 exhaustion*. Listopad 2018. Dostupné z: https://nfware.com/blog-ipv4-limitations.

[23] Oki, E. a Roberto Rojas-Cessa, C. V. *Advanced Internet Protocols, Services, and Applications*. Wiley, 2012. ISBN 9781118180822.

[24] Olshansky, S. a Wilton, R. *Internet of Things Devices as a DDoS Vector* [online]. April 2019. Dostupné z: https://www.internetsociety.org/blog/2019/04/internet-of-things-devices-as-a-ddos-vector/.

[25] Olson, D. *Best Ways to Avoid IP Fragmentation* [online]. Jan 2019. Dostupné z: https://www.summitir.com/2019/01/18/best-ways-to-avoid-ip-fragmentation/.

[26] Postel, J. *INTERNET PROTOCOL* [DARPA Internet Program Protocol Specification]. Září 1981. Dostupné z: https://tools.ietf.org/html/rfc791.

[27] Ramanathan, S., Mirkovic, J., Yu, M. a Zhang, Y. SENSS Against Volumetric DDoS Attacks. In: *Proceedings of the 34th Annual Computer Security Applications Conference*. New York, NY, USA: Association for Computing Machinery, 2018, s. 266–277. ACSAC '18. Dostupné z: https://doi.org/10.1145/3274694.3274717. ISBN 9781450365697.

[28] Rouse, M. *Jumbo frames*. Sep 2013. Dostupné z: https://searchnetworking.techtarget.com/definition/jumbo-frames.

[29] Socolofsky, T. a Kale, C. *A TCP/IP Tutorial* [IETF]. Leden 1991. Dostupné z: https://tools.ietf.org/html/rfc1180.

[30] Stretch. *Path MTU Discovery* [online]. Aug 2008. Dostupné z: https://packetlife.net/blog/2008/aug/18/path-mtu-discovery/.

[31] Touch, J. *Updated Specification of the IPv4 ID Field* [IETF]. únor 2013. Dostupné z: https://tools.ietf.org/html/rfc6864.

[32] Vasseur, J.-P. a Dunkels, A. Chapter 6 - Transport Protocols. In: Vasseur, J.-P. a Dunkels, A., ed. *Interconnecting Smart Objects with IP*. Boston: Morgan Kaufmann, 2010, s. 63 – 74. Dostupné z: http://www.sciencedirect.com/science/article/pii/B9780123751652000065. ISBN 978-0-12-375165-2.

[33] Walkowski, D. *What is a Distributed Denial-of-Service Attack?* [online]. June 2019. Dostupné z: https://www.f5.com/labs/articles/education/what-is-a-distributed-denial-of-service-attack-.

[34] Wireshark. *Common MTU Values* [online]. Červen 2008 [cit. 2019-12-16]. Dostupné z: https://wiki.wireshark.org/MTU.

[35] Wu, C. a Vyncke, E. *Resolve IPv4 Fragmentation, MTU, MSS, and PMTUD Issues with GRE and IPsec* [online]. Jan 2019. Dostupné z: https://www.cisco.com/c/en/us/support/docs/ip/generic-routing-encapsulation-gre/25885-pmtud-ipfrag.html.

Příloha A

Obsah přiloženého paměťového média

Struktura kořenového adresáře přiloženého paměťového média:

```
Kořenový adresář
├── source/        Zdrojové kódy s návodem
├── doc/           Zdrojové texty bakalářské práce
└── thesis.pdf     Text bakalářské práce
```