ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Detekce síťových útoků typu Denial of Service
Student:	Otto Hollmann
Vedoucí:	Ing. Tomáš Čejka
Studijní program:	Informatika
Studijní obor:	Teoretická informatika
Katedra:	Katedra teoretické informatiky
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Prostudujte princip síťových útoků typu Denial of Service (DoS) a Distributed DoS (DDoS).
Navrhněte vhodnou datovou strukturu pro rychlé ukládání statistik o provozu zařízení na páteřních počítačových sítích.
Navrhněte efektivní algoritmus pro aktualizaci pozorovaných statistik ke každému aktivnímu zařízení v posledních N krátkých historických časových intervalech (kde N je volitelné přirozené číslo).
Pomocí existujících metod popsaných např. v [1] vytvořte funkční prototyp detekčního modulu pro systém [2].
Prototyp bude používat navržené datové struktury a algoritmy, při realizaci by měl být kladem důraz na paměťové a výkonnostní nároky.
Výsledné řešení otestujte pomocí dat, která dodá vedoucí práce.

Seznam odborné literatury

[1] T. Čejka, "Hardware Accelerated Anomaly Detection in Computer Networks", A Doctoral Study Report, Czech Technical University in Prague, 2014.
[2] T. Cejka, V. Bartoš, M. Svepes, Z. Rosa, and H. Kubatova, “NEMEA: A Framework for Network Traffic Analysis,” in *12th International Conference on Network and Service Management (CNSM 2016)*, Montreal, Canada, 2016.

L.S.

doc. Ing. Jan Janoušek, Ph.D.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 30. ledna 2017

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra teoretické informatiky

Bakalářská práce

Detekce síťových útoků typu Denial of Service

Otto Hollmann

Vedoucí práce: Ing. Tomáš Čejka

15. května 2017

Poděkování

Děkuji svému vedoucímu práce Ing. Tomáši Čejkovi za odborné vedení, cenné rady a připomínky při psaní této práce. Dále bych chtěl poděkovat sdružení CESNET z. s. p. o. za poskytnutí testovacích dat a pomoc při testování. V neposlední řadě bych rád poděkoval své rodině za podporu během mého studia a psaní této práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 15. května 2017 .....

Odkaz na tuto práci

Hollmann, Otto. *Detekce síťových útoků typu Denial of Service*. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Útoky typu Denial of Service (DoS) jsou v dnešní době stále častější a dostupné pro každého. Útoky omezují běžné uživatele a můžou působit finanční ztráty provozovatelům služeb i poskytovatelům připojení. Tato práce se zabývá detekcí volumetrických útoků na základě analýzy síťových toků v reálném čase. Zabývá se problematikou vzniku útoků a popisuje stávající řešení útoků. Dále popisuje návrh detekčního algoritmu využívajícího historických okének k detekci náhlého zvýšení velikosti provozu. V závěru práce je popsána implementace a testování výsledného detekčního programu. Detektor je implementován jako modul do systému NEMEA, který vyvíjí CESNET, zájmové sdružení právnických osob.

Klíčová slova Distributed Denial of Service, detekce DDoS útoků, analýza sítových toků, síťová bezpečnost, systém NEMEA

Abstract

Denial od Service attacks (DoS) have recently become more frequent and available for everyone. Attacks cause discomfort for common users and may also cause financial loss for service providers or internet service providers. This thesis deals with the detection of volumetric attacks based on real time network flow analysis. It also deals with the emergence of DoS attacks and describes existing solutions. Furthermore, it describes the design of a detection algorithm utilizing historical windows for detection of a sudden increase in traffic size. The detector is implemented as a module for NEMEA developed by CESNET, association of legal entities. Implementation details and testing of the resulting detection program are provided as well.

Keywords Distributed Denial of Service, DDoS detection, network flow analysis, network security, NEMEA system

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Internet v dnešní době bereme jako samozřejmost, většina z nás si bez něj život ani nedokáže představit. Používáme ho ke komunikaci, hledání informací, video hovorům nebo ke sdílení souborů, zkrátka ke všemu. Občas se ale stane, že služba nebo stránka, kterou právě hledáme, nefunguje. Příčin může být mnoho, od pravidelné údržby přes nefunkční internetové připojení až po útok typu Denial of Service (DoS) cílený na konkrétní službu. Na jedné straně stojí uživatelé, kteří se musí pouze smířit s tím, že jejich oblíbená služba právě teď nefunguje. Na druhé straně stojí firmy, které kvůli nefunkčním stránkám přicházejí o zákazníky a tedy i o zisk.

Efektivní obranu proti útokům DoS nelze provádět kdekoliv, odhalení útoků zneužívajících konkrétních zranitelností nebo vyčerpávání systémových prostředků vyžaduje analýzu paketů. Proto je vhodnější takové útoky řešit až v koncových sítích, kde není takový objem dat a je možná důkladnější analýza provozu. Naopak volumetrické útoky, které cílí na omezenou kapacitu linek, nelze řešit v koncových sítích. V případě zahlcení linky je filtrování provozu v koncové síti neúčinné, legitimní požadavky neprojdou, protože linka zůstane stále zahlcená. Navýšení kapacity linek by mohlo být řešením, ale tato varianta je nákladná. Linky o kapacitě, která by odolala i větším útokům, by byly drahé. Navíc dobře motivovaný útočník by mohl přijít s ještě větším útokem. Tento problém je nutné řešit u poskytovatelů připojení (ISP). Útoky je nutné detekovat a filtrovat už na páteřní síti, kde mají linky dostatečnou kapacitu, a do koncové sítě pouštět pouze provoz, který danou síť nezahltí a dokáže si s ním poradit.

Tato práce se zabývá detekcí volumetrických útoků typu DoS pomocí analýzy síťových toků. Jejím cílem je vytvořit opensource prototyp detekčního modulu do systému NEMEA. Současný stav řešení problému je nevyhovující, nelze ho automatizovat a komerční řešení jsou drahá. Nyní na CESNETu probíhá vývoj čističky síťového provozu, která s tímto tématem souvisí. Výsledný modul bude automaticky přesměrovávat závadný provoz do čističky.

Síťový tok je sekvence paketů se stejnou zdrojovou a cílovou adresou a po-

užitým protokolem. Toky se sbírají pomocí exportérů/sond, kde se jednotlivé pakety agregují do záznamů o toku. Ty se pak zasílají na kolektor, odkud se přeposílají do jednotlivých modulů.

Struktura práce

V kapitole 1 je po krátkém seznámení se základními pojmy vysvětlena problematika vzniku útoků, jejich rostoucí popularita a dostupnost. Dále kapitola představuje cíle této práce.

Kapitola 2 se zabývá typy útoků, rozborem stávajícího řešení obrany a je zde popsán systém NEMEA.

Kapitola 3 popisuje požadavky na detekční modul, návrh algoritmu a výběr vhodných datových struktur.

Kapitola 4 popisuje samotnou implementaci, výpočetní i paměťovou složitost výsledného detektoru.

V Kapitole 5 je analyzován kód a jsou popsány výsledky testování na datech poskytnutých vedoucím práce a na datech z reálného provozu na síti CESNET2.

Kapitola 1

Popis problému, cíl práce

Tato kapitola vysvětluje základní pojmy, vznik útoků a jejich rostoucí popularitu. V závěru kapitoly jsou specifikovány cíle práce.

1.1 DoS útoky, základní pojmy

Útoky typu DoS jsou vedeny proti webovým stránkám nebo internetovým službám, jejich cílem je znefunkčnit a znepřístupnit je ostatním uživatelům. Nejčastěji se setkáváme se zahlcením služby množstvím nevalidních požadavků tak, že nestíhá odpovídat ani na ty legitimní. Druhým nejčastějším způsobem je využití chyby, která bude spotřebovávat výpočetní výkon nebo způsobí pád serveru. Ani v jednom případě se ale nejedná o ovládnutí služby, pouze o její znepřístupnění. Pro běžného uživatele služba, na kterou je veden útok, bude vypadat jako nedostupná nebo bude zpomalená. Pokud ale některá služba vykazuje takové příznaky, neznamená to automaticky, že je proti ní veden útok.

- **Denial of Service (DoS)**: V českém překladu znamená odmítnutí služby. Tento útok jde pouze z jednoho zařízení. Často se ale zkratkou *DoS* souhrnně označují všechny útoky typu Denial of Service.
- **Distributed Denial of Service (DDoS)**: Jedná se o DoS, který je veden z více zařízení. Síť těchto zařízení se obvykle nazývá botnet. Útočník jim pošle příkazy a zařízení pak začnou útočit na jeden cíl. Častým jevem je podvrhování zdrojových IP adres, takže případná detekce zařízení účastnících se útoku je velmi složitá.
- **Distributed Reflection Denial of Service (DRDoS)**:
  Vychází z DDoS útoku, ale liší se tím, že zařízení, která útočí, nejsou napadená. Toho je docíleno tak, že útočník přímo nebo přes botnet pošle dotaz a jako zdrojovou IP adresou podvrhne IP adresu oběti. Tato zařízení na dotaz odpoví, ale odpověď jde díky podvržené adrese k oběti a

ne k útočníkovi. Výhodou takového dotazu je, že odpověď muže být výrazně větší než původní dotaz, takže dojde ke znásobení velikosti útoku a útočník i s malým počtem zařízení je schopen vytvořit masivní útok.

1.2 Jak vzniká botnet

Botnet je síť zařízení, která fungují autonomně. Toho je dosaženo pomocí malware [1], přes který útočník dané zařízení ovládá. Taková zařízení se nazývají *zombies*. Kromě plnění příkazů od útočníka je cílem napadených zařízení šířit malware dál a tím zvětšovat botnet. Článek [1] popisuje, nejčastější způsoby, jak k infikování zařízení dochází:

- Slabé zabezpečení: Útočník zjistí nebo uhodne heslo. V případě jednoduchých zařízení, například domácích routerů, je velká pravděpodobnost, že bude fungovat výchozí heslo, které se pro většinu modelů dá najít na internetu. Další možností je například použití slovníkového útoku [2].

- Chyba uživatele: Uživatel si může infikovat počítač při procházení webu nebo nakaženým mailem — například otevřením závadné přílohy nebo kliknutím na odkaz.

- Bezpečnostní chyba: Při tomto způsobu nakažení se využívá zranitelností v běžně používaných programech nebo zařízeních. Čím používanější zařízení nebo program, tím více na něj může existovat exploitů [3]. Využívá se již známých chyb, které byly opraveny, ale uživatel nemá nejnovější verzi nebo těch neznámých/neopravených — tzv. *zero-day* zranitelností. Dalším příkladem mohou být již zmíněné domácí routery, jejichž životnost vysoce převyšuje dobu, kdy na ně výrobce vydává nový firmware.

To, že je zařízení infikováno, nemusí jeho majitel vědět a v drtivé většině případů se to ani nedozví, protože zařízení na první pohled funguje pořád stejně.

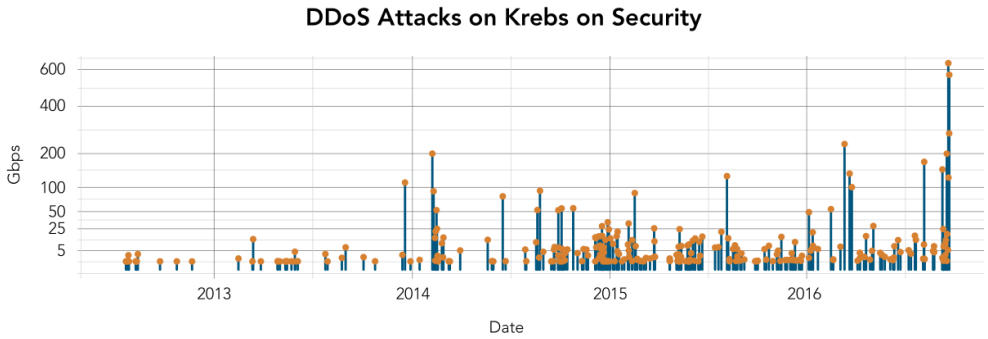
Pro ovládání botnetů se používá architektura klient-server nebo peer-to-peer spojení. Nakažené zařízení se přihlásí ke svému C&C serveru [4], přes který dostává od útočníka příkazy. Nemusí se jednat jen o útoky typu DDoS. Botnety se dají využít k inzerování reklam uživateli, který má na počítači malware, nebo k rozesílání nevyžádané pošty. Lze je využít i při sbírání citlivých informací o uživateli (osobní údaje, hesla, čísla platebních karet) a tato data následně prodat na černém trhu.

[1] Škodlivý software

[2] Při slovníkovém útoku zkouší útočník jako hesla běžná slova nebo často používaná hesla, například `123456`, `password`, atp.

[3] Program nebo sekvence příkazů, které zneužívají zranitelnosti v software.

[4] C&C command-and-control



Obrázek 1.1: Zvyšující se velikost útoků vedených proti stránkám `https://krebsonsecurity.com`. (*Zdroj: [3]*)

Přicházející internet věcí (IoT) s sebou přináší i mnoho slabě zabezpečených nebo nezabezpečených zařízení a tím i snazší vytváření velkých botnetů a snižování cen jejich pronájmů.

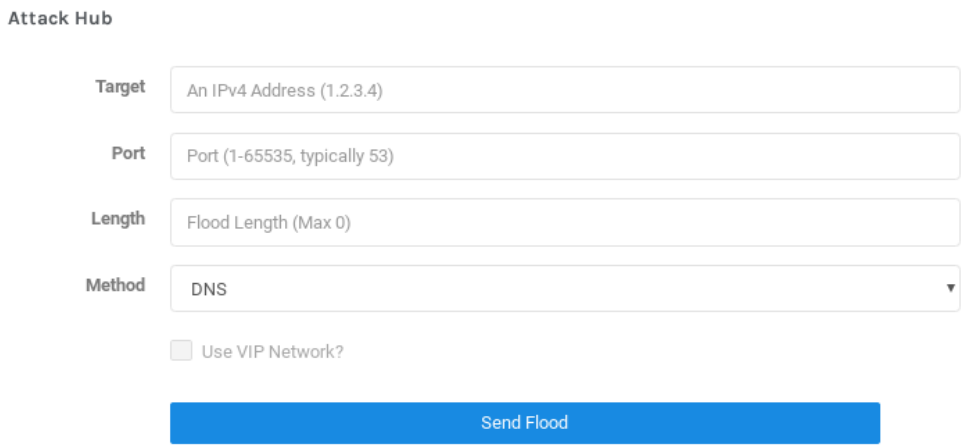
1.3 Rostoucí popularita útoků

Historie DoS útoků sahá až do roku 1974. Jeden z prvních velkých DDoS útoků se odehrál v roce 1999 proti univerzitní síti v Minnesotě a ochromil ji na víc než dva dny, jak uvádí článek [2]. Od té doby se počet i síla útoků zvětšují. Tento zvyšující se trend potvrzuje i autor v článku [3]. Na svých stránkách píše o kybernetické bezpečnosti, a právě proto se opakovaně stává cílem velkých DDoS útoků. Poslední dobou se počet útoků cílených na jeho stránky snižuje, ale za cenu větší síly útoků. Některé z nich přesahují hranici 100 Gbps. Doposud největší útok vedený na jeho stránky dosáhl síly 623 Gbps. Dle jeho analýzy ho vyvolalo přibližně 24 000 zařízení, převážně z internetu věcí. Vývoj počtu útoků proti těmto stránkám znázorňuje obrázek 1.1. Zároveň autor poukazuje na souvislost mezi shluky útoků a daty zveřejnění jeho článků.

Do povědomí veřejnosti se útoky dostaly zejména díky hnutí Anonymous kolem roku 2011. Jejich aktivity spojené s podporou WikiLeaks celý problém medializovaly (například v deníku The Guardian [4]) a ukázaly tak „sílu“ DDoS útoků široké veřejnosti.

1.4 Útok na objednávku

DDoS útoky se stávají čím dál častějšími. Totéž potvrzuje i autor článku [5]. Dříve byly botnety tvořeny lidmi pro vlastní potřebu nebo zábavu, velikost botnetu udávala míru prestiže v rámci komunity. Nyní se z toho stal byznys. S jejich rostoucí popularitou se začaly objevovat na diskuzních fórech, převážně ruských a čínských, nabídky pronájmu botnetů. Postupem času se



Obrázek 1.2: Ukázka „objednání“ útoku. (*Zdroj: [6]*)

Tabulka 1.1: Ceník DDoS útoků uvedený na stránkách [6]

Cena	Doba trvání	Síla
15 $	15 min	10+ Gbps
20 $	30 min	10+ Gbps
40 $	60 min	10+ Gbps
65 $	120 min	10+ Gbps
150 $	60 min	50+ Gbps
300 $	120 min	50+ Gbps
600 $	480 min	50+ Gbps

kromě pronájmu botnetů začal nabízet i samotný DDoS útok. Potenciálnímu zájemci pak stačí pouze zadat cíl útoku a poslat peníze. O vše ostatní se postará někdo jiný. Na obrázku 1.2 je vidět intuitivní grafické rozhraní pro „objednání“ útoku.

Dnes se takové služby dají na internetu najít pod označením *booter*, *stress test*, *ddos for hire* nebo *ddoser*. Cena pronájmu nebo útoku se odvíjí od velikosti botnetu, doby pronájmu, jeho umístění případně i podle využití. Útoky jsou levné, jejich ceny začínají už na jednotkách dolarů, jak uvádí tabulka 1.1. Na některých stránkách je útok nabízen dokonce zdarma.

1.5 Nástroje pro útok

Nejčastěji používaným nástrojem pro DoS útoku je LOIC (Low Orbit Ion Canon). Jedná se o opensource program určený k zátěžovým testům sítě. Směrem k cíli dokáže generovat velké množství TCP nebo UDP paketů a tím ho zaplavit. Dokáže se také připojit k IRC serveru a plnit zadané příkazy.

Dále [7] uvádí, že se lidé často pomocí tohoto nástroje dobrovolně připojují k „dobrovolným“ botnetů a propůjčují tak svůj počítač k větším útokům.

1.6 Uplatnění detektoru

Cílem práce je vytvořit prototyp detekčního modulu do systému NEMEA, který bude porovnáváním flow záznamů v historických intervalech detekovat volumetrické DDoS útoky v reálném čase. Svojí detekcí by měl přispět ke snížení účinnosti těchto útoků. Dalším přínosem práce je rozšíření opensource projektu NEMEA. Tento detektor, stejně jako systém NEMEA, bude volně k dispozici a každý ho bude moci použít k monitorování vlastní sítě a k předcházení bezpečnostním hrozbám.

Kapitola 2

Analýza

V této kapitole jsou popsány typy jednotlivých útoků a princip jejich fungování. Dále jsou zde rozepsaná podobná řešení zabývající se volumetrickými útoky. V této kapitole se seznámíme se systémem NEMEA a použitými knihovnami. V závěru je popsán současný stav řešení.

2.1 Typy útoků

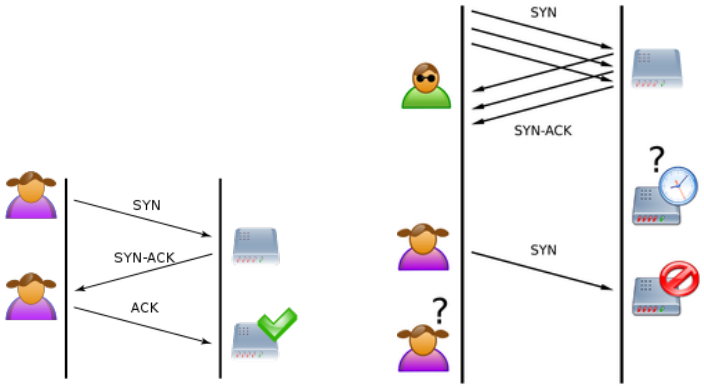
Existuje mnoho typů (D)DoS, tato práce se ale zaměřuje pouze na volumetrické útoky. Jejich charakteristickou vlastností je, že při nich jde směrem k cíli velké množství internetového provozu, který danou síť může i zahltit. Zde jsou jejich nejčastější typy:

- **TCP SYN flood** — Zranitelnost spočívá ve vlastnostech TCP protokolu, konkrétně v *three-way handshake* [5] při otevírání spojení. Útočník posílá pouze SYN pakety, na které cíl odpoví SYN-ACK. Cíl potom marně čeká na odpověď ACK a má „napůl“ otevřené spojení než vyprší časový limit. Obrázek 2.1 znázorňuje běžné navázání spojení a SYN flood.

- **HTTP flood** — Při tomto typu útoku je na cílový server posíláno velké množství zdánlivě legitimních požadavků typu HTTP POST nebo GET. Tyto požadavky jsou vytvářeny tak, aby spotřebovávaly výpočetní výkon a systémové zdroje cílového serveru. Server může být nedostupný, aniž by došlo k zahlcení linky.

- **ICMP flood** — Známý také jako „ping flood“ — spočívá v posílání velkého množství objemných paketů ICMP. Nástroj ping funguje tak, že

[5] Způsob navázání spojení v TCP — Klient pošle SYN, server odpoví SYN-ACK, klient odpoví ACK a spojení je otevřeno

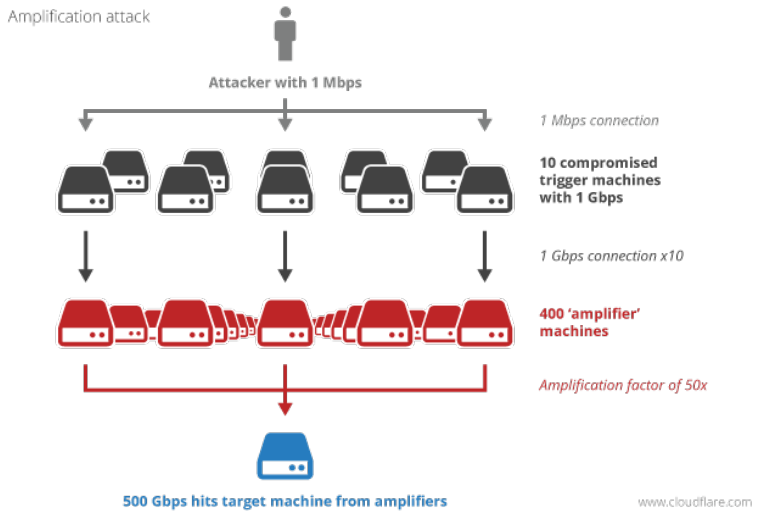


Obrázek 2.1: Vlevo normální navázání spojení, vpravo SYN flood útok. (*Zdroj: [8]*)

odešleme ICMP zprávu *Echo request* a čekáme odpověď *Echo reply*, velikost paketu se zachovává. Při tomto útoku pouze odesíláme nepřetržitě zprávy *Echo request*, aniž bychom kontrovali, zda přišla odpověď.

- **Reflected Attack** — Útočník pošle pakety s podvrženou zdrojovou adresou na velký počet zařízení. Tato zařízení je přijmou a odpoví. Odpověď ale místo k útočníkovi půjde na podvrženou adresu.
- **DNS flood** — Při tomto útoku je na cíl (DNS server) zasíláno mnoho dotazů z různých IP adres. Není možné rozlišit, které dotazy pochází od skutečných klientů a které ne. Může dojít k vyčerpání kapacity linky nebo zahlcení serveru a jeho následné nedostupnosti. Pokud se jedná o autoritativní server pro nějakou doménu, způsobí se tím znepřístupnění celé domény.
- **DNS amplification** — V tomto případě je DNS server pouze nástrojem útoku, na rozdíl od předchozího typu, kde byl DNS server cílem. Tento typ útoků patří mezi největší z hlediska přeneseného objemu dat. Zejména proto, že DNS odpověď může být výrazně větší než původní dotaz, takže i s linkou o malé kapacitě lze vyvolat velký útok. Obrázek 2.2 znázorňuje tento útok.
- **UDP flood** — Tento typ využívá protokolu UDP. Přes něj se odesílají velké pakety na náhodné porty cílového zařízení, které kontroluje, jestli na daném portu neposlouchá nějaká aplikace. Pokud ne, odešle odpověď *ICMP Destination Unreachable*. Opět lze zfalšovat zdrojovou adresu.

Kromě těchto útoků existují i takové, které se zaměřují na konkrétní zranitelnost nebo konkrétní aplikaci/platformu. Takové je při sledování velikosti síťového provozu obtížné detekovat, protože se přenášenou velikostí ani počtem paketů zásadním způsobem neliší od legitimního provozu.



Obrázek 2.2: Mnohonásobně zesílený DNS amplifikační útok. (*Zdroj: [9]*)

- **Slowloris** — Otevírá se mnoho síťových spojení a udržují se otevřená. Podle článku [10] útočník pošle nedokončený HTTP požadavek a těsně před vypršením časového limitu pošle další nedokončený požadavek.
- **Nuke** — Zástupce starších DoS útoků, který posílá oběti poškozené a fragmentované pakety. Na zařízeních se starší verzí operačního systému Microsoft Windows po přijetí takového paketu došlo k BSoD [6].
- **Ping of death** — Útočník pošle paket větší než je maximální velikost — 65535 bajtů. To je možné pouze díky fragmentaci paketu, po defragmentování dojde k přetečení zásobníku a pádu systému. Současné systémy jsou vůči tomuto útoku odolné.
- **Teardrop attacks** — Útok opět využívá chyby při fragmentaci paketů. Spočívá v překrývajících se fragmentech, takže při následném sestavení paketu došlo k chybě a k pádu systému. V současnosti je vůči útoku většina systémů imunních.
- **TCP Reset** — Útočník zavírá již otevřená spojení oběti pomocí paketů s příznakem TCP RST.
- **SSH Process Table** — Útok je podobný jako SYN flood. Vytváří se při něm stovky spojení s obětí pomocí protokolu SSH, aniž by se dokončil přihlašovací proces. Oběť pak nemá kapacitu na navázaní dalších (legitimních) SSH spojení.

[6] Blue Screen of Death — Modrá obrazovka smrti — chybové hlášení operačních systémů Microsoft Windows při kritické chybě

- **LAND Attack** — Local Area Network Denial — Útočník posílá TCP SYN pakety, které mají stejnou zdrojovou a cílovou adresu. Po přijetí takového paketu začne oběť odpovídat sama sobě.

- **ARP Poisoning** — Při tomto útoku je potřeba mít přístup do LAN sítě oběti. V článku [11] se píše, že protokol ARP se používá ke zjištění MAC adresy podle IP adresy. Útočník zahltí router zprávami *ARP reply* s podvrženou MAC adresou. Router si zaznamená MAC adresu útočníka do ARP tabulky a pakety pak posílá jemu. Provoz díky tomu jde přes útočníka, který ho může například měnit nebo blokovat.

- **Shrew attack** — Tento typ útočí na protokol TCP, konkrétně na RTO — čas opakování. Narušuje běžnou komunikaci pravidelným posíláním paketů, které ruší spojení na stejné lince.

2.2 Stávající řešení ochrany

2.2.1 Čistička síťového provozu

Čistička — *scrubber* nebo *scrubbing center* — je v článku [12] popsaná jako specializované zařízení, kde je provoz analyzován a ten závadný odstraněn. Tato zařízení jsou obvykle používána u velkých společností jako jsou poskytovatelé připojení nebo cloudových služeb. Do čističky je provoz přesměrován až při útoku pomocí DNS nebo směrovacích pravidel (BGP). V čističce se závadný provoz odstraní a zpět do sítě jde pouze legitimní provoz. Pokud neprobíhá žádný útok, provoz nejde přes čističku.

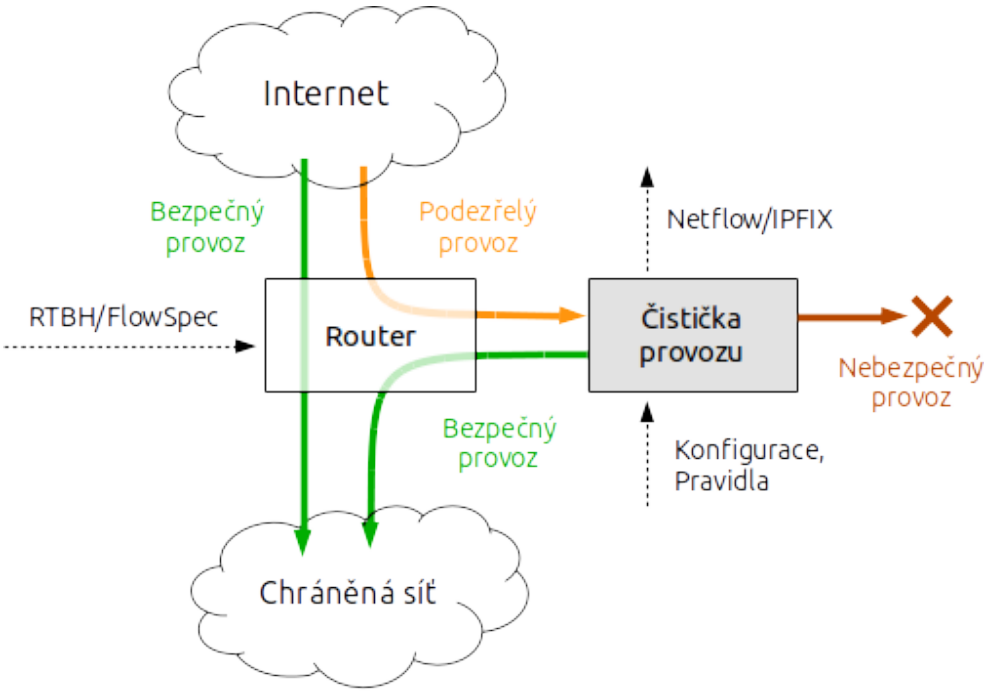
Projekt Liberouter

Sdružení CESNET vyvíjí vlastní čističku, pracuje na třetí (síťové) vrstvě ISO/OSI [7] modelu a podle článku [13] se skládá z běžného počítače a 100 GE síťové karty složené z FPGA obvodů. Počítač zde slouží pouze k řízení, o filtrování se stará sama karta. Do počítače se řídícímu software posílají pouze statistiky o zablokovaném provozu.

Míru filtrování lze nastavit jednoduchými pravidly pro každou síť zvlášť. Pokud nejsou překročeny prahové hodnoty, nic se nefiltruje a čistička všechen provoz propouští beze změny. Při překročení prahové hodnoty se začnou blokovat IP adresy, ze kterých přišel největší objem dat. Adresy k blokování se vybírají tak, aby výsledný provoz, který se vrací zpět do sítě, splňoval definované limity a nedošlo k zahlcení sítě. Schéma zapojení čističky do sítě popisuje obrázek 2.3.

Výhodou opensource řešení je, že díky znalosti a dostupnosti zdrojových kódů je možné pochopit princip jeho fungování a lépe jej konfigurovat. I když

[7] Referenční model standardizace počítačových sítí.



Obrázek 2.3: Schéma čističky. (*Zdroj: CESNET*)

se cena karty pohybuje okolo 15 000 €, v porovnání s komerčním řešením se jedná o relativně malou částku.

Stránky projektu [14].

2.2.2 Fastnetmon

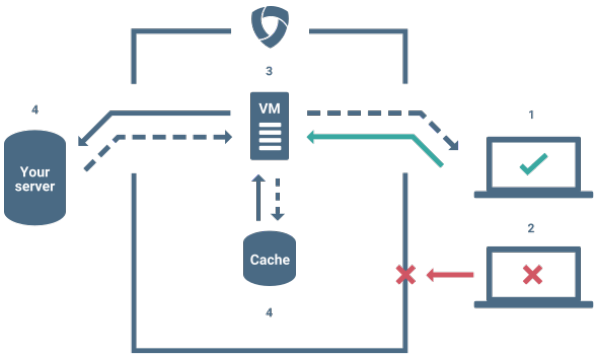
Dalším zástupcem v boji proti DDoS útokům je Fastnetmon dostupný z [15]. Jedná se o opensource projekt, který vyniká podporou mnoha formátů záznamů o síťových tocích (například sFlow v5, Netflow v5, v9, jFlow, IPFIX, BGP flow spec) a detekcí útoku do dvou vteřin. Opět nabízí široké možnosti nastavení limitů.

Záznamy o jednotlivých tocích se ukládají do mapy podle ASN. Následně se každou vteřinu přepočítávají a kontrolují zadaná pravidla. Při překročení pravidel je incident nahlášen a na základě nasbíraných dat se odhaduje typ útoku.

Podle [16] je program testován „pouze“ na 10 Gb lince s 12 Mpps [8]. Udávaná paměťová náročnost je lineární a lze ji spočítat podle vzorce:

$$total\_number\_of\_hosts * 208 * 3$$

[8] Millions of Packets Per Second



Obrázek 2.4: Google Project Shield. (*Zdroj: [17]*)

Při těchto parametrech byl server s procesorem Intel i7 3820 vytížen na 100 %. Proto je toto řešení vhodnější spíše pro ochranu koncových sítí než nasazení na páteřních sítích.

2.2.3 Google Project Shield

Poněkud odlišnější obranou je Project Shield od společnosti Google. Na jejich stránkách [17] je nejprve nutné poslat přihlášku. Po jejím schválení stačí změnit DNS záznamy tak, aby veškerý provoz na danou doménu směřoval přes servery Google. Podle nápovědy [18] je služba dostupná zdarma, ale pouze pro zpravodajské weby. Pro komerční nebo osobní potřeby ji nelze použít. Schéma fungování zobrazuje obrázek 2.4. V případě stránek zabezpečených pomocí SSL je potřeba nahrát certifikát včetně jeho privátního klíče na stránky projektu.

Služba funguje jako reverzní proxy, tedy přeposílá provoz na původní stránky. Před DDoS útoky chrání dvěma způsoby. Pomocí vlastních algoritmů identifikuje škodlivý provoz a zablokuje ho. Druhým způsobem ochrany je „caching“, při kterém se uloží aktuální verze stránek a tím se redukuje množství požadavků. Při případném útoku (mnoha požadavcích o jednu stránku), který by nebyl vyfiltrován prvním způsobem, projde pouze první požadavek. Další požadavky již zodpoví Project Shield z uložené odpovědi na první požadavek.

Mezi nevýhody tohoto řešení patří fakt, že provoz jde přes servery třetích stran, kde může být prohlížen nebo dokonce pozměněn, a to i při šifrované komunikaci. Další nevýhodou jsou omezené možnosti uplatnění. Naopak velkou výhodou jsou nulové náklady pro provozovatele webu.

2.2.4 Obrana v NIX.CZ

Otázkou DDoS útoků se v roce 2013 zabýval i peeringový uzel NIX.CZ. Řešení nebylo jednoduché, prosté odpojení sítě v případě útoku by nic nevyřešilo,

protože by útok přišel jinou cestou. Zároveň tu byla otázka, do jaké míry může NIX.CZ provoz sledovat a případně filtrovat. „Konkrétně bylo navrženo řešení pomocí Remotely Triggered BlackHole (RTBH), kdy by členové mohli centrálnímu route serveru poslat informaci o tom, že nechtějí některý provoz dostávat. Ten by se pak uvnitř NIXu ztrácel, aniž by se o tom zdrojová síť dozvěděla.“ Píše autor v článku [19].

Toto řešení je sice funkční, ale přináší s sebou vyšší nároky na správce sítí a zanáší do procesu lidský faktor znemožňující automatizaci.

Ve stejném roce vznikl i projekt FENIX, který provozuje sdružení NIX.CZ. Jeho cílem je vytvořit bezpečnou síť služeb, které budou dostupné i v případě DoS útoku.

2.2.5 IDS, IPS

Intrusion Detection System (IDS) a Intrusion Prevention System (IPS) jsou systémy, které monitorují aktivitu na síti a detekují škodlivý provoz.

Podle článku [20] je IDS pasivní a pouze monitoruje provoz na síti. V něm se snaží nalézt definované vzory a podle nich vyhodnocuje, zda dochází k útoku nebo ne. Pro správnou funkčnost je potřeba mít databázi signatur — vzorků útoků.

IPS se snaží útokům předcházet. Na rozdíl od IDS poskytuje vyšší ochranu, protože dokáže kontrolovat situaci na více vrstvách ISO/OSI modelu a vyhodnotit i míru rizika. Na základě toho buď upozorní správce, na firewallu začne blokovat dané pakety nebo odpojí server od sítě. Kromě detekce již známých útoků jsou některé tyto systémy schopny částečně detekovat i nové druhy útoků. Klíčová pro ně je opět databáze signatur.

2.2.6 Honeypot

Úkolem těchto systémů je na sebe lákat útočníky a následně analyzovat jejich činnost. Díky této analýze je pak možné vytvářet pravidla pro detekci. Aktuálnost takových pravidel je klíčová pro správnou funkčnost antivirů, IDS a IPS systémů.

2.3 Systém NEMEA

NEMEA — Network Measurements Analysis — je opensource projekt, který byl poprvé představen v roce 2013 a neustále se vyvíjí. Jedná se o systém složený z nezávislých modulů. Ty mají zpravidla alespoň jedno vstupní a jedno výstupní jednosměrné komunikační rozhraní, pomocí kterých jsou spojeny s ostatními moduly.

Data mezi nimi se přenášejí ve formě záznamů o IP tocích. Všechny záznamy v rámci jednoho rozhraní musí mít stejný formát, který se ale může lišit od formátu jiného komunikačního rozhraní. Použitý formát dat může zvolit

vývojář modulu, přičemž výchozí formát zpráv, které si moduly předávají, je UniRec, popsaný v sekci 2.3.1.

NEMEA podporuje několik typů komunikačních rozhraní. Pro lokální komunikaci modulů na stejném stroji se používá UNIX socket a pro komunikaci přes síť využívá TCP socket. Dalšími dvěma typy jsou file a blackhole pro ukládání do souboru respektive zahazování.

Každý modul běží jako samostatný proces, což přináší výhodu, že v případě pádu jednoho modulu nejsou ohroženy ostatní moduly. Rovněž je možné je zapínat a vypínat dynamicky, nezávisle na ostatních modulech. Další výhodou je možnost sledování paměťové a výpočetní náročnosti jednotlivých modulů. Každý modul má svůj vlastní úkol, například přehrávání dat ze souboru, detekci anomálií, detekce útoků, filtrování nebo logování.

Systém NEMEA zpracovává záznamy o tocích v reálném čase nebo téměř reálném bez nutnosti data ukládat. Zápis a čtení by se mohl stát úzkým hrdlem pro další moduly, a proto data zůstávají pouze v operační paměti, kromě modulů, které data ukládají záměrně. Díky tomu je možné v reálném čase zpracovávat data z velkých sítí na jednom serveru, uvádí autoři v [21].

2.3.1 UniRec

Unified Record (UniRec) je datový formát pro přenos a zpracování jednoduchých nestrukturovaných záznamů. Byl vyvinutý na CESNETu a používá se pro komunikaci mezi moduly. Šablona slouží na straně odesílatele i příjemce k přístupu na offsety a přes komunikační rozhraní se posílají samotná data.

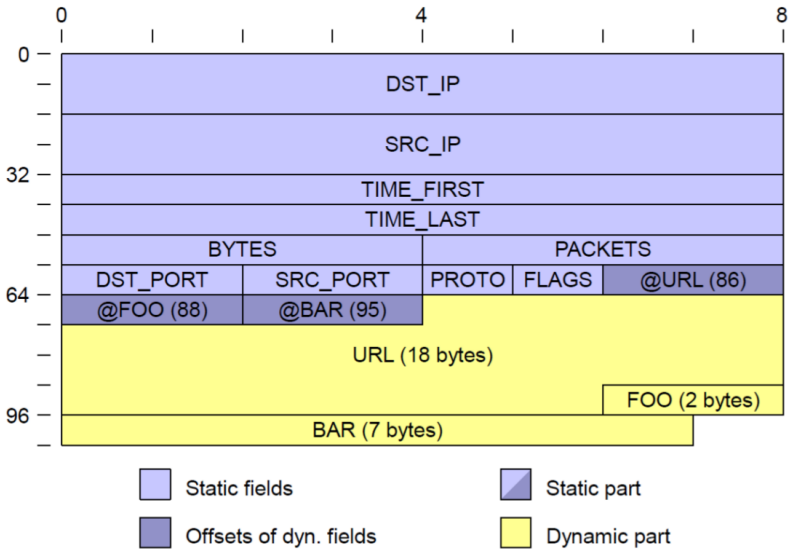
Knihovna vyniká velmi rychlým přístupem k polím v záznamu. Díky přímému přístupu je efektivita téměř srovnatelná se strukturou v jazyku C. Pro vytvoření UniRec záznamu je nutné nejprve vytvořit šablonu — specifikovat sadu pole a jejich typy. Pak už stačí pole naplnit hodnotami pomocí jednoduchých maker, uvádí dokumentace knihovny [22].

Z šablony UniRec na obrázku 2.5 jsou pro detektor důležitá tato pole: `DST_IP`, `SRC_IP`, `TIME_FIRST`, `TIME_LAST`, `BYTES`. Jejich význam je popsán v sekci 4.1.1.

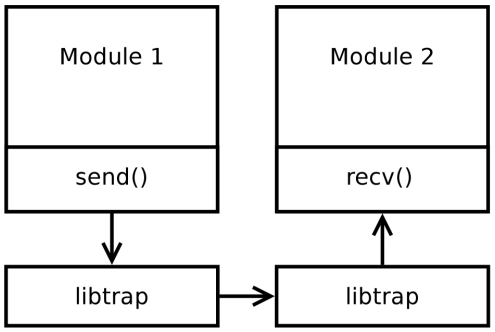
2.3.2 LibTrap

Knihovna TRAP abstrahuje modul od použitého typu komunikačního rozhraní a zjednodušuje psaní nových modulů. Vývojáři poté stačí zavolat funkce `trap_recv()` pro příjem nebo `trap_send()` pro odeslání dat [9]. Knihovna rovněž řeší chyby spojené s komunikačním rozhraním, například po přerušení spojení z důvodů restartování modulu se postará o automatické navázání spojení.

[9] V případě použití jazyku C



Obrázek 2.5: Formát UniRec. (*Zdroj: [21]*)



Obrázek 2.6: Schéma komunikace mezi moduly v NEMEA. (*Zdroj: [21]*)

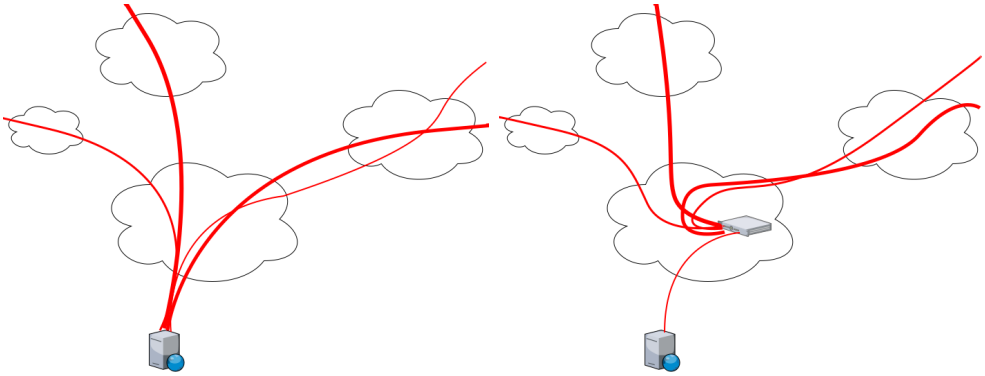
Komunikační rozhraní je reprezentováno sdíleným objektem libtrap, který je slinkován s každým NEMEA modulem, uvádí článek [21]. Obrázek 2.6 znázorňuje propojení modulů.

2.3.3 Common

Poslední částí NEMEA frameworku je knihovna `Common`. Poskytuje mnoho funkcí a datových struktur použitelných při psaní modulů, například rozptylovací funkce, prefixový strom nebo B^+ strom.

2.4 Současný stav

Nyní je na síti CESNET2 zapojena čistička síťového provozu (viz sekce 2.2.1) a filtruje provoz, který je do ní přesměrován. Přesměrování se dělá manuálně



Obrázek 2.7: Přesměrování provozu do čističky. Vlevo server, na který je vedený útok. Vpravo přesměrování celého provozu do čističky a následné vyčištění od nežádoucího provozu.

pomocí směrovacích pravidel. Obrázek 2.7 znázorňuje přesměrování provozu a jeho následné vyčištění.

Kapitola 3

Návrh

V této kapitole jsou popsány požadavky na detekční modul, návrh algoritmu a výběr optimální datové struktury.

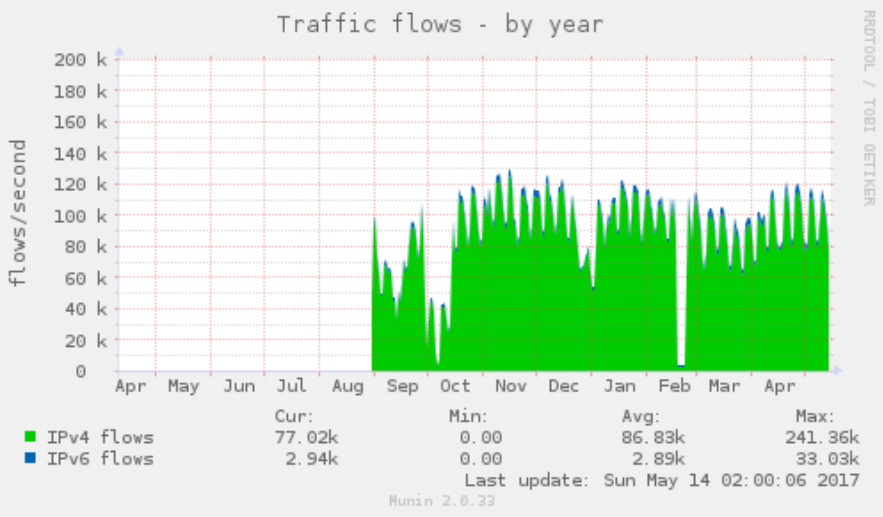
3.1 Požadavky na detekční modul

Modul by měl být schopný monitorovat páteřní sítě s 100 Gbps linkami v reálném čase. Dále bude kladen důraz na efektivní využití paměti a výpočetní složitost. Pro komunikaci s ostatními moduly budou použita komunikační rozhraní popsaná v sekci 2.3. Modul bude mít jedno vstupní rozhraní, přes které bude přijímat záznamy o síťových tocích, a jedno výstupní rozhraní, přes které bude odesílat hlášení o útocích. Vzhledem k tomu, že hlášení bude probíhat pomocí komunikačního rozhraní, není potřeba vytvářet žádné grafické rozhraní. Programovacím jazykem, ve kterém bude modul implementován, bude jazyk C. K tomuto závěru vedly výše zmíněné požadavky na modul a také podpora jazyku C systémem NEMEA.

Posledním požadavkem na vyvíjený modul bude jeho škálovatelnost. Pro správné fungování podobných detektorů na různě velkých sítích je žádoucí mít možnost upravit prahové hodnoty. Modul by tedy měl umět pomocí parametrů při spuštění tyto hodnoty nastavovat.

3.2 Návrh algoritmu

Modul bude využívat historické intervaly — okénka. Tím je z velké části předurčen i samotný algoritmus detekce. Pro každou cílovou IP adresu budou v historických okénkách uloženy statistiky o počtu bajtů posílaných na tuto adresu. Díky tomu bude možné porovnávat průměrný tok s aktuálním a sledovat jeho náhlé zvýšení. Aby se předešlo *false-positive* detekcím, bude muset toto zvýšení trvat několik okének a tok před tímto zvýšením musí být větší než nula.



Obrázek 3.1: Graf počtu IPv4 a IPv6 toků.

Tímto se zabrání například detekcím přenosů souborů nebo měřením rychlosti internetu, které s útoky DoS nemají nic společného.

Dalším potenciálním problémem jsou přenosy trvající delší dobu — například přenosy výsledků fyzikálních nebo meteorologických měření. Detekcím těchto událostí se zabrání, když se zároveň s počtem přenesených bajtů bude sledovat i počet různých zdrojů provozu na stejný cíl.

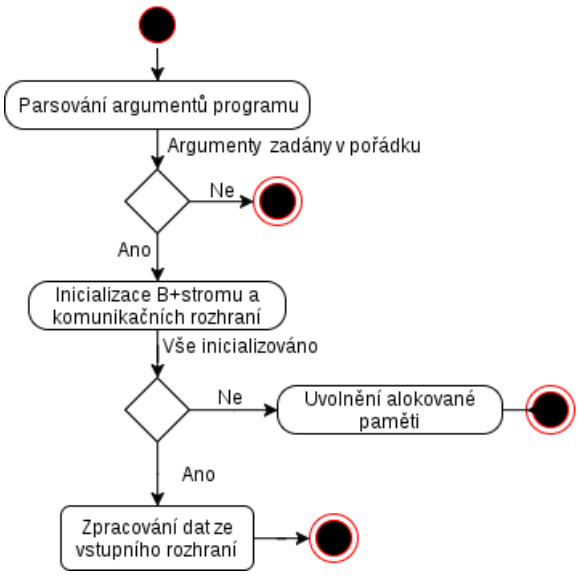
Detekce bude probíhat pouze na IPv4. Ačkoliv se adresy IPv6 používají stále častěji, většina útoků se odehrává v adresním prostoru IPv4. Graf počtu toků v síti CESNET2 na obrázku 3.1 ukazuje, že stále převládá provoz na IPv4. Pro podporu IPv6 bude potřeba vyřešit zejména problém obrovského adresního prostoru.

Podrobný diagram fungování detektoru je na obrázcích 3.2, 3.3 a 3.4.

3.2.1 Implementace algoritmu

Při každé aktualizaci záznamu detektor projde okénka od prvního do $n-2$. a provede následující body:

- Spočítá průměrný tok z předchozích okének a vynásobením průměru parametrem `threshold_flow_rate` spočítá mezní hodnotu.
- Spočítá průměrný počet zdrojů z předchozích okének a vynásobením průměru parametrem `threshold_ip_count_rate` spočítá mezní hodnotu.
- Zkontroluje, zda je tok v následujících dvou okénkách větší než vypočítaná mez.



Obrázek 3.2: Schéma inicializace detektoru.

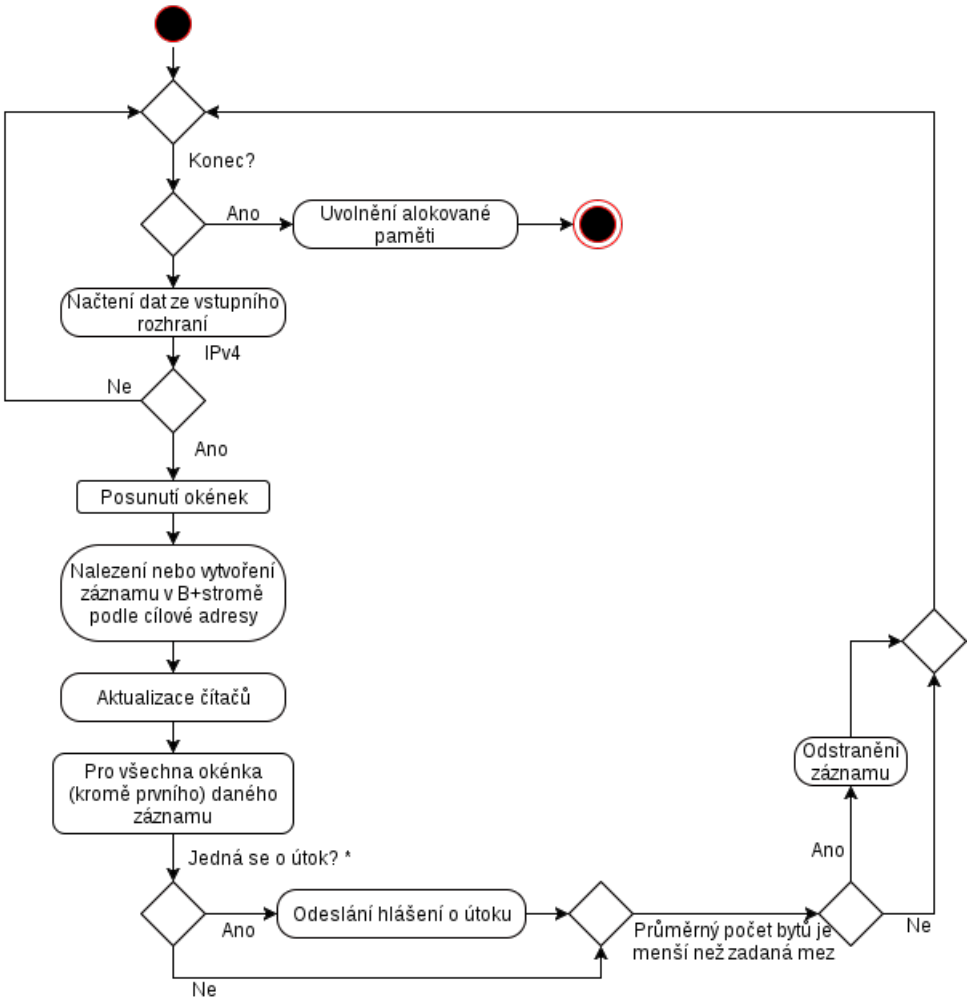
- Zkontroluje, zda je počet zdrojů provozu v následujících dvou okénkách větší než vypočítaná mez.
- Zkontroluje, zda je tok v následujících dvou okénkách větší než zadaná minimální velikost útoku.

Pokud jsou poslední tři podmínky splněny, je daný provoz vyhodnocen jako útok. Obrázek 3.5 znázorňuje algoritmus detekce a uložení provozu do historických okének.

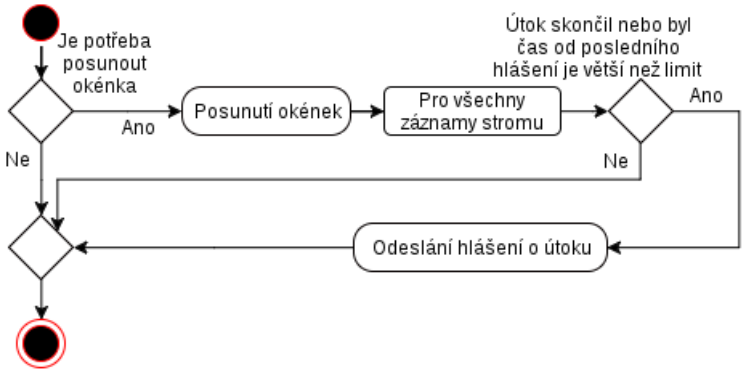
3.3 Výběr datových struktur

Datová struktura pro ukládání by měla mít efektivní operace vkládání a vyhledávání. Dále by měla mít nízkou paměťovou náročnost a umožňovat lineární procházení všech uložených záznamů. Posledním požadavkem na strukturu, který částečně souvisí s efektivním využitím paměti, je možnost uložit předem nespecifikované množství prvků. Počet IPv4 adres je sice konečný, ale lze předpokládat, že nebude potřeba mít uložené informace o všech adresách najednou. Tedy bylo by neefektivní předem alokovat místo pro všechny IPv4 adresy.

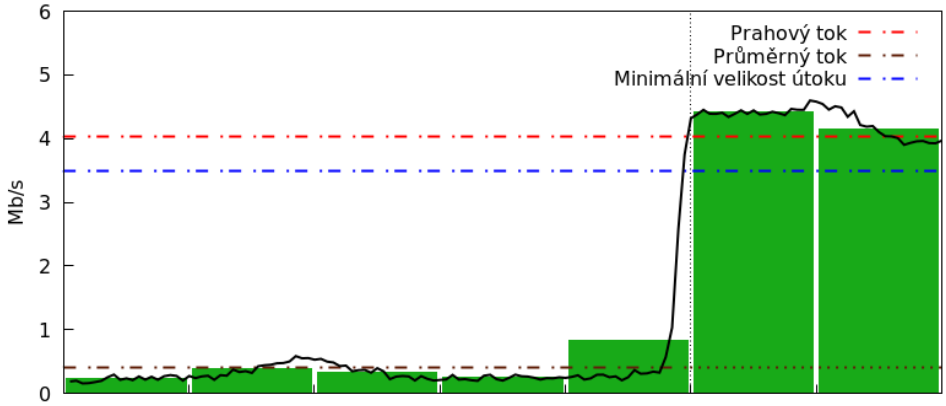
- **Rozptylovací tabulka** je datová struktura, která podle klíčů ukládá hodnoty do tabulky. Existují různé implementace, například jako asociativní pole nebo mapování pomocí rozptylovací funkce, ale všechny předem vyžadují počet prvků. Zvolením velkého počtu prvků zbytečně



Obrázek 3.3: Schéma běhu detektoru.
*) Algoritmus detekce útoku je rozebrán v sekci 3.2.1.



Obrázek 3.4: Schéma posouvání okének.



Obrázek 3.5: Algoritmus detekce.
Postupně od prvního do $n-2$. okénka se kontrolují zadané mezní hodnoty. Na obrázku je útok detekován až v posledních dvou okénkách. Aby byl útok detekován, musí být větší než zadaná minimální velikost. Podobně jako pro velikost bajtů se kontrolují podmínky pro počet zdrojových adres.

zvýšíme paměťovou náročnost. Naopak v případě malého počtu prvků vzroste časová složitost kvůli řešení kolizí. Výhodou rozptylovací tabulky je vyhledávání, vkládání i mazání v konstantním čase v případě, že nejsou kolize.

- **Červeno-černý strom** je podle [23] binární vyhledávací strom. Je vyvažovaný, jeho uzly jsou buď červené nebo černé a musí splňovat následující pravidla:
  - Každý uzel je buď červený nebo černý.
  - Kořen stromu je černý.
  - Každý list je černý.
  - Každý červený uzel má oba své syny černé.
  - Všechny listy mají stejnou tzv. černou hloubku, definovanou jako počet černých uzlů na cestě od nich do kořene.

  Operace vkládání, hledání a mazání se provádí v logaritmickém čase. Podmínky na vyvažování jsou volnější, a proto není potřeba tak často vyvažovat.

- **B strom** je vyvažovaný vyhledávací strom, který se hodí na ukládání velkých dat do pomalejších pamětí, typicky na disk. Toho je docíleno tím, že se hodnoty uzlů nacházejí v paměti vedle sebe, a proto dochází maximálně k několika výpadkům, uvádí [24]. Operace vkládání, vyhledávání i mazání mají logaritmickou výpočetní složitost.

Definice m-árního B stromu ze stejného zdroje:

- Všechny listy mají stejnou hloubku.
- Kořen musí obsahovat minimálně jeden prvek.
- Pokud kořen není listem, pak má minimálně dva syny.
- Vnitřní uzly různé od kořene musejí obsahovat minimálně $\lfloor \frac{m-1}{2} \rfloor$ prvků a tedy $\lfloor \frac{m-1}{2} \rfloor + 1 = \lfloor \frac{m+1}{2} \rfloor$ synů.
- Listy musejí obsahovat minimálně $\lfloor \frac{m-1}{2} \rfloor$ prvků.

- **B^+ strom** je podle [24] struktura, která vychází z B stromu a je rozšířená o následující podmínky:
  - Vnitřní uzly obsahují pouze klíče, nikoliv hodnoty.
  - Vlastní hodnoty jsou uloženy v listech.
  - Každý list obsahuje maximálně $m-1$ prvků a odkaz na následující list.
  - Klíč na pozici i obsahuje první klíč uložený v prvním listu podstromu na pozici $i+1$.

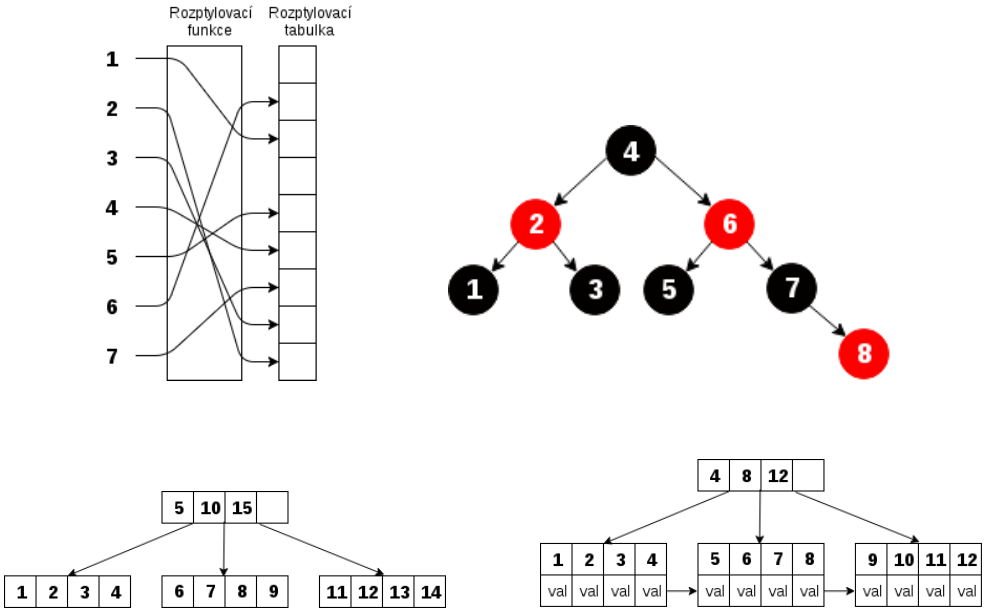
  Tyto podmínky zajišťují i možnost procházení celého stromu jako spojového seznamu.

Vzhledem k výše zmíněným důvodům byl pro uchovávání záznamů o IP adresách vybrán B^+ strom. Další výhodou této volby je, že je již implementovaný v knihovně `Common`.

3.3.1 Ukládání zdrojů provozu

Dále bylo potřeba vybrat vhodnou strukturu pro počítání unikátních zdrojů provozu. V adresním prostoru IPv4 je 2^{32} adres. Vzhledem k tomu, že se má počet zdrojů ukládat pro každý záznam cílové adresy, asociativní pole nepřipadá v úvahu. Ukládání do stromu také není v tomto případě efektivní. Potřebujeme pouze vědět, zda daná adresa komunikovala s cílem či nikoliv. Stromové struktury mají v porovnání s touto binární informací velkou paměťovou režii.

Jako nejoptimálnější možnost se jeví rozptylovací tabulka. Její velikost bude menší než celý adresní prostor, takže více zdrojových adres bude mapovaných na jedno místo v tabulce. To ale nevadí, protože v případě masivního DDoS útoku se rozdíl projeví i při tomto zkreslení.



Obrázek 3.6: Schémata diskutovaných datových struktur. Vlevo nahoře je rozptylovací tabulka, vpravo nahoře je červeno-černý strom, vlevo dole je B strom a vpravo dole B^+ strom.

Kapitola 4

Realizace

Tato kapitola popisuje úlohu implementovaných funkcí, vstupní a výstupní komunikační rozhraní, rozbor paměťové a výpočetní složitosti.

4.1 Popis implementace

Implementace se skládá z pěti funkcí.

- Funkce `compare_32b` slouží k porovnávání prvků uložených v B^+ stromu. Strom je implementován tak, aby mohl ukládat libovolné datové struktury. Aby bylo možné danou strukturu do stromu uložit, musí být možné jednotlivé instance seřadit. Tedy musí být definován operátor, který porovná libovolné dvě instance této struktury.

- Funkce `main` má na starosti řízení celého detektoru. Nejprve zpracuje zadané parametry detektoru, inicializuje B^+ strom a komunikační rozhraní. Poté už přijímá přes vstupní rozhraní záznamy o tocích, které ukládá do příslušných záznamu v B^+ stromu, a volá funkce pro posouvání okének. Při aktualizaci záznamu kontroluje, zda nedošlo k útoku. Pokud ano, alokuje se struktura `flood_t` a při nejbližším volání funkce `move_window` se spočítá průměrný tok a průměrný počet zdrojů před útokem. V případě přijetí signálu `SIGTERM`, `SIGINT` nebo chybě (například při nedostatku paměti) funkce zařídí korektní uvolnění paměti a ukončení detektoru.

- Funkce `move_window` posune všechny záznamy stromu o zadaný počet okének. Při posouvání zároveň kontroluje, zda se má odeslat průběžné hlášení o již existujícím útoku, aby byl dodržen stanovený interval.

- Funkce `report_flood` přes výstupní rozhraní odešle hlášení o útoku na základě informací uložených v příslušné struktuře `flood_info`.

- Funkce **delete_tree** uvolní prostředky použité v B^+ stromu. Knihovna Common má sice implementovanou funkci pro uvolnění paměti stromu, ale v tomto případě je před odstraněním nutné strom projít, až poté uvolnit strom pomocí knihovní funkce. V listech stromu může být ukazatel na dynamicky alokovanou strukturu se záznamem o útoku, který je před odstraněním stromu nutné uvolnit. Pokud by tato struktura nebyla uvolněna, docházelo by k únikům paměti — *memory leak* — a zvyšovalo by se množství alokované paměti.

4.1.1 Vstup a výstup

Detektor má dvě komunikační rozhraní ve formátu UniRec. Vstupní rozhraní přijímá záznamy o tocích a vyžaduje následující pole:

- SRC_IP — Zdrojová adresa toku.
- DST_IP — Cílová adresa toku.
- BYTES — Počet bajtů v toku.
- TIME_FIRST — Čas prvního paketu v toku.
- TIME_LAST — Čas posledního paketu v toku.

Díky tomu nezáleží na tom, jestli se jedná o data z aktuálního provozu nebo přehrávání dat ze souboru pro účely testování.

Přes výstupní rozhraní se posílají hlášení o detekovaných útocích a jsou odesílána v pravidelných intervalech po dobu trvání útoku. Význam polí v hlášení je následující:

- DST_IP — Cílová adresa cíle útoku.
- BYTES — Přibližný počet bajtů poslaných na cílovou adresu kvůli útoku v daném intervalu.
- TIME_FIRST — Začátek intervalu.
- TIME_LAST — Konec intervalu.
- EVENT_ID — Unikátní identifikátor útoku.
- EVENT_SEQ — Pořadové číslo hlášení útoku se stejným EVENT_ID.
- AVG_BYTES_CURRENT — Průměrný počet bajtů za vteřinu odeslaných na cílovou adresu v aktuálním intervalu.
- AVG_BYTES_ORIGINAL — Průměrný počet bajtů za vteřinu odeslaných na cílovou adresu před útokem.

- `AVG_IP_CNT_CURRENT` — Průměrný počet zdrojových IP adres komunikujících s cílovou adresou v daném intervalu.
- `AVG_IP_CNT_ORIGINAL` — Průměrný počet zdrojových IP adres komunikujících s cílovou adresou před útokem.

4.1.2 Použité datové struktury

Historická okénka byla implementovaná jako kruhové pole. Díky tomu má jejich posouvání konstantní asymptotickou složitost. Okénka jsou deklarovaná jako statické pole ve struktuře. Proto jejich velikost musí být známa už při překladu a nelze ji měnit při spouštění.

Pro uchovávání záznamů byla navržena struktura `dst_addr_record_t`. jednotlivé členské proměnné jsou vysvětleny pomocí komentářů přímo v kódu.

```
typedef struct dst_addr_record_s {
    /* Historicka okenka pro pocet bajtu */
    uint64_t bytes_per_int[N_INTERVALS];
    /* Historicka okenka pro pocet zdrojovych adres */
    uint16_t src_ip_per_int[N_INTERVALS];
    /* Rozptylovaci tabulka zdrojovych adres */
    char src_ip_table[N_TABLE_SIZE / 8];
    /* Pocet bajtu smerovanych na danou adresu */
    uint64_t total;
    /* Ukazatel na strukturu s~informacemi o utoku */
    flood_t *flood_info;
} dst_addr_record_t;
```

Velikost rozptylovací tabulky `src_ip_table` musí být také známa při překladu. Její velikost je zde dělena osmi, protože se pro ukládání adres používají přímo bity a ne bajty.

V případě, že bude detekován útok, alokuje se v příslušném záznamu `dst_addr_record_t` struktura `flood_t`, která uchovává podrobné informace o útoku. Informace v této struktuře jsou klíčové pro samotné hlášení útoku. Význam jednotlivých proměnných je opět vysvětlen pomocí komentářů.

```
typedef struct flood_s {
    /* Cilova adresa */
    uint32_t dst_ip;
    /* Cas posledniho hlaseni */
    uint32_t last_reported;
    /* Pocet bajtu */
    uint64_t total_bytes;
    /* Unikatni ID utoku */
    uint64_t uuid;
    /* Pocitadlo poctu hlaseni */
```

```
    uint8_t msg_cnt;
    /* Prumerny tok pred utokem */
    uint64_t avg_flow_original;
    /* Prumerny tok pri utokem */
    uint64_t avg_flow_current;
    /* Prumerny pocet IP pred utokem */
    uint32_t avg_ip_cnt_original;
    /* Prumerny pocet IP pri utokem */
    uint32_t avg_ip_cnt_current;
    /* Pocet okenek obsazenych v hlaseni */
    uint8_t window_cnt;
    /* Priznak, zda jsou prumery pred utokem platne */
    bool is_valid;
} flood_t;
```

Obrázek 4.1 znázorňuje implementování navržených datových struktur a závislosti mezi nimi.

4.1.3 Parametry detektoru

- `-m`, `--minimal_attack_size`
  Minimální velikost útoku v `kb/s`. Výchozí hodnota je 1 000 `kb/s`.

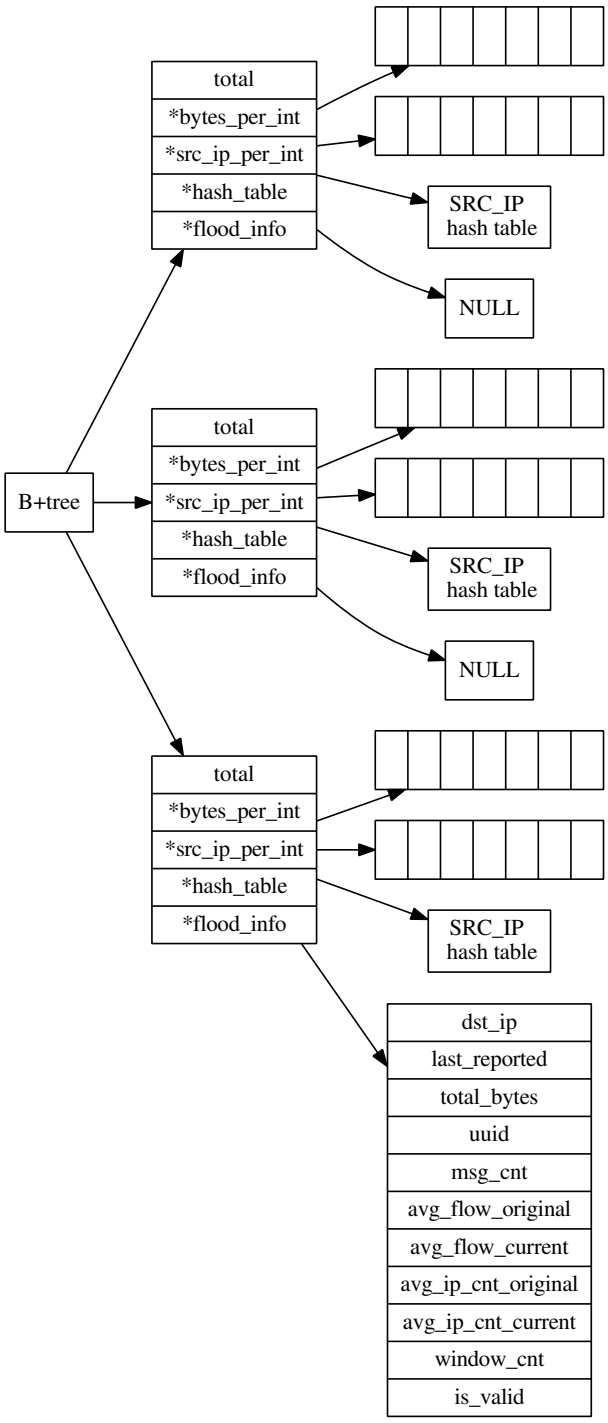
- `-t`, `--threshold_flow_rate`
  Poměr, kolikrát musí narůst průměrný tok v následujících dvou okénkách v porovnání s průměrem z předchozích okének.

- `-c`, `--threshold_ip_count_rate`
  Poměr, kolikrát musí narůst průměrný počet unikátních zdrojových IP adres (nebo prefixů) v následujících dvou okénkách v porovnání s průměrem z předchozích okének.

- `-s`, `--mask_source_addresses`
  Zpracovávat místo zdrojových adres pouze prefix odpovídající zadané masce. Výchozí hodnota je 24.

- `-d`, `--mask_destination_addresses`
  Zpracovávat místo cílových adres pouze prefix odpovídající zadané masce. Výchozí hodnota je 32.

- `-p`, `--min_threshold_before_pruning`
  Minimální tok na cílovou IP adresu (průměr ze všech okének) `kb/s`, aby záznam nebyl odstraněn. Záznamy obsahující méně než zadané minimum budou při posouvání okének odstraněny. Výchozí hodnota je 10 `kb/s`.



Obrázek 4.1: Implementace datových struktur pro ukládání statistik o síťovém provozu.

4.2 Složitost

4.2.1 Paměťová složitost

Tato složitost udává, kolik paměti bude detektor vyžadovat v závislosti na počtu cílových adres.

Detektor potřebuje mít uložený B^+ strom, jehož paměťová náročnost je $\mathcal{O}(n)$. V listech stromu je uložena struktura `dst_addr_record_t`, která může obsahovat ukazatel na strukturu `flood_t`. V nejhorším případě, kdy by byl veden útok na všechny adresy ve stromě, by bylo nutné alokovat

$$n * (\text{sizeof}(dst\_addr\_record\_t) + \text{sizeof}(flood\_t))$$

Po odstranění konstant vyjde $\mathcal{O}(n)$, tedy lineární složitost v závislosti na počtu cílových adres.

4.2.2 Výpočetní složitost

V následující sekci bude rozebrána složitost jednotlivých operací používaných v detektoru. Složitost je vztažena k počtu záznamů o cílových adresách ve stromu.

- Inicializace — Nejprve je nutné inicializovat B^+ strom. Jedná se o operaci s konstantní složitostí, která se provádí pouze jednou — $\mathcal{O}(1)$.
- Posouvání okének — Tato operace vyžaduje nejprve projít celý strom. Díky použití B^+ stromu je možné jej procházet lineárně jako spojový seznam a následně v každém záznamu o cílové adrese posunout okénka. Samotné posunutí o jedno okénko vpřed má konstantní složitost. Asymptotická složitost operace posouvání okének je $\mathcal{O}(n)$, kde n je počet unikátních cílových adres nebo prefixů.
- Odstranění celého stromu — Nejprve je nutné ho projít a zkontrolovat, zda neobsahuje ukazatel na alokovanou paměť — záznam o útoku. Pokud obsahuje, musí se odeslat hlášení o útoku s uloženými informacemi a paměť uvolnit. Následně je možné celý strom odstranit. Celková složitost odstranění B^+ stromu je $\mathcal{O}(n) * \mathcal{O}(1) + \mathcal{O}(n)$, tedy $\mathcal{O}(n)$. Tato operace se provádí pouze při ukončení modulu, tj. pouze jednou.
- Odstranění záznamu — Odstranění prvku z B^+ stromu má složitost $\mathcal{O}(\log n)$.
- Reinicializace stromu — Tato akce je potřeba v případě, že detektor přijme záznam o toku, který je příliš v budoucnosti, a bylo by nutné posunout okénka. Proto se místo toho celý strom odstraní se složitostí $\mathcal{O}(n)$ a následně znovu inicializuje se složitostí $\mathcal{O}(1)$. Celková složitost operace je $\mathcal{O}(n)$.

- Vkládání/aktualizace záznamu — Vyhledání nebo vložení nového záznamu do stromu je operace s logaritmickou složitostí. Následná aktualizace záznamu je provedena v konstantním čase, tedy celkem $\mathcal{O}(\log n)$.

Při běžném chodu detektoru se nejvíce využívají operace aktualizace záznamu se složitostí $\mathcal{O}(\log n)$ a posouvání okének se složitostí $\mathcal{O}(n)$. K reinicializaci stromu by při běžném provozu nemělo dojít. Případná reinicializace by se provedla místo posouvání okénka. Vzhledem k tomu, že její složitost je menší než složitost posouvání okénka, lze ji nahradit složitostí posouvání okénka a dále ji neuvažovat.

Celková složitost tedy je tedy součtem složitosti inicializace při spuštění detektoru, hledání záznamu, posouvání okénka a odstranění celého stromu — $\mathcal{O}(1) + \mathcal{O}(\log n) + \mathcal{O}(n) + \mathcal{O}(n)$. Celkem tedy $\mathcal{O}(n)$.

Kapitola 5

Testování

Tato kapitola se zabývá analýzou kódu, testováním na poskytnutých datech i nasazením v reálném provozu.

5.1 Statická analýza kódu

Pro statickou analýzu byl použit nástroj `coverity scan`, dostupný z [25]. Tento nástroj dokáže v kódu odhalit zranitelnosti a chyby popsané na stránkách [26] včetně jejich CWE označení. Při testování detekčního modulu nebyly objeveny žádné závažné chyby.

5.2 Dynamická analýza kódu

Modul byl otestován pomocí nástroje Valgrind [27]. Při běhu modulu ani po jeho skončení nebyly detekovány žádné chyby ani neuvolněné paměťové bloky.

5.3 Lokální testování

Při tomto testování byl modul spuštěn na počítači a monitoroval aktivitu na ethernetovém rozhraní. Po několika minutách „běžného“ provozu byl na testovací počítač zahájen útok z několika zařízení. Útok byl detekován až po 10 minutách, ale v hlášení byl uveden správný čas začátku útoku. Tato prodleva byla způsobena exportérem a také tím, že je potřeba počkat na posun okének, aby v hlášení mohly být vypočítány metriky útoku.

5.4 Testování na datech od vedoucího práce

Data poskytnutá vedoucím byly anonymizované záznamy komunikace ze sítě CESNET2 při detekovaném útoku. Tyto vzorky lze rozdělit od čtyř kategorií podle typu útoku:

- skenování portů: Ani jeden vzorek tohoto typu útoku nebyl detektorem vyhodnocen jako útok. Tento výsledek byl očekávaný a je žádoucí. Záznam komunikace obsahoval horizontální scanování různých hostů, ale bez jejich odpovědi. Od útočníka tedy šel velký provoz, ale na různé adresy.

- „Bruteforce“ útok hrubou silou: Podobné jako předchozí typ, ani tento nebyl detekován. Záznamy obsahovaly především útok na jednu adresu z jednoho zařízení. Opět se nejednalo o DDoS a chování detektoru bylo správné.

- DNS amplifikační útok: Tento typ již patří do DDoS útoků. Ze tří vzorků byl detekován pouze jeden. Zbylé dva vzorky nebyly detekovány, protože záznam byl příliš krátký a skončil dříve, než se stihly inicializovat potřebné proměnné.

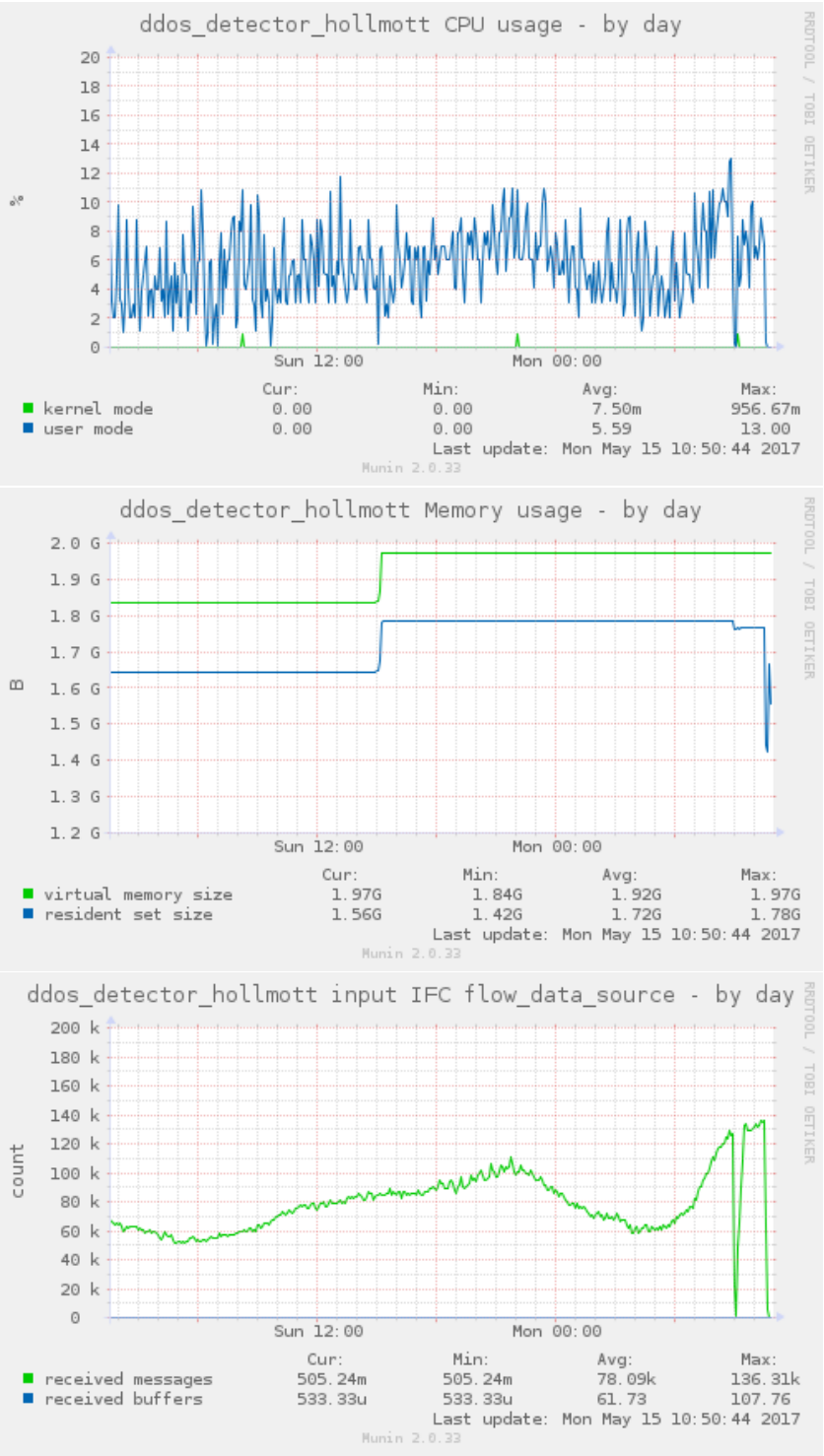
- DoS útok: Zde byl pouze jeden záznam, který při prvním testování nebyl detekován, protože nesplňoval podmínky na zvýšení počtu zdrojů. Při druhém testování byla podmínka pro nárůst počtu zdrojů upravena pomocí přepínače `-c 1`. To způsobilo, že se kontroloval pouze nárůst počtu bajtů směrovaných na danou adresu a útok byl detekován. V obou případech byl výstup detektoru správný.

5.5 Testování výkonu

Modul byl testován na počítači s procesorem Intel Core2 Duo P8600 a linuxovou distribucí Debian 8 (64b). Při průměrné zátěži procesoru 80 % dokázal zpracovávat kolem 146 500 záznamů za vteřinu.

5.6 Nasazení na síti CESNET2

Jednotlivé verze detektoru byly kromě testování na poskytnutých datech nasazeny také na síti CESNET2. Nejprve modul sledoval počet bajtů pro cílové adresy a po překročení mezní hodnoty i zdrojové adresy komunikující s danou cílovou adresou. Na testovacích datech fungoval správně, ale při nasazení na síti CESNET2 měl mnoho falešných detekcí, protože detekoval i přenosy velkých souborů. V další verzi zůstalo pouze sledování počtu bajtů pro cílové adresy a bylo přidáno počítání unikátních zdrojů provozu pro každou cílovou adresu. Zde bylo opět mnoho falešných detekcí, protože modul do průměrného provozu počítal i prázdná okénka. Docházelo k tomu, že velikost před útokem byla nula a jako útok byl detekován jakýkoliv nárůst splňující minimální velikost. V poslední verzi tedy byla doplněna podmínka nenulového provozu před útokem.



Obrázek 5.1: Grafy využití systémových prostředků detekčním modulem na síti CESNET2 a počet zpracovaných toků.

Grafy na obrázku 5.1 ukazují využití procesoru, paměti a počet zpracovaných toků detekčním modulem verze `1.0.2` nasazeným na síti CESNET2. Při zpracování kolem 78 000 toků za vteřinu detekční modul využívá průměrně 6 % výkonu procesoru [10] a potřebuje přibližně 2 GB operační paměti.

[10] Intel Xeon CPU E5-2670 0 @ 2.60GHz

Závěr

Cílem bakalářské práce bylo prostudovat problematiku útoků typu DoS a navrhnout a implementovat detekční modul do systému NEMEA. Prvním krokem bylo seznámení se s problémem, základními pojmy a definování cíle práce. Dále byly rozebrány principy jednotlivých útoků, stávající možnosti obrany proti nim a byl popsán aktuální stav řešení obrany. Na základě této analýzy byly vymezeny požadavky na detekční modul a vybrány vhodné datové struktury. V poslední části práce byla popsána implementace detekčního modulu, rozbor paměťové i výpočetní složitosti a výsledky jeho testování.

Implementovaný modul je určený pro nepřetržitý běh a umí detekovat volumetrické útoky. Při testování na datech od vedoucího úspěšně detekoval DNS amplifikační útok a při lokálním testování odhalil pokusné útoky. Pro správnou funkčnost modulu je nutné pomocí parametrů správně nastavit příslušné mezní hodnoty, které se pro každou síť můžou lišit.

Detekční modul je nyní k dispozici v repozitáři Nemea-Detectors `https://github.com/CESNET/Nemea-Detectors/tree/ddos/ddos_detector` ve větvi `ddos`. Modul je nyní nasazen ve verzi `1.0.2` na síti CESNET2, jeho hlášení o detekovaných útocích budou řešit bezpečnostní týmy. Tyto informace zároveň budou sloužit k dalšímu vývoji detektoru. Pokud se osvědčí, bude modul sloučen s hlavní větví repozitáře a bude distribuován společně se systémem NEMEA. Bude možné ho stáhnout přímo z repozitáře na githubu nebo jako balík `.rpm` pro linuxovou distribuci CentOS 7.

Modul detekuje útoky pouze na IPv4, ale do budoucna je v plánu rozšíření detekce i na IPv6. Dalším možným rozšířením je zohledňování kapacity linek do podsítí, sledování jejich saturace a typu provozu nebo whitelist adres, které budou považovány za bezpečné. Pro budoucí vývoj je také vhodné automatizovat testovací proces nových verzí modulu.

Literatura

[1] Nykodýmová, H.: Botnety: nová internetová hrozba. 2006, [Cited 2017-04-22]. Dostupné z: https://www.lupa.cz/clanky/botnety-internetova-hrozba/

[2] *History of DDoS Attacks*. 2017, [Cited 2017-04-22]. Dostupné z: https://security.radware.com/ddos-knowledge-center/ddos-chronicles/ddos-attacks-history/

[3] Akamai on the Record KrebsOnSecurity Attack. 2016, [Cited 2017-04-16]. Dostupné z: https://krebsonsecurity.com/2016/11/akamai-on-the-record-krebsonsecurity-attack/

[4] Halliday, J.; Arthur, C.: WikiLeaks: Who are the hackers behind Operation Payback? 2010, [Cited 2017-05-08]. Dostupné z: https://www.theguardian.com/media/2010/dec/08/anonymous-4chan-wikileaks-mastercard-paypal

[5] Krčmář, P.: Pronajmu botnet, ceník přiložen... aneb jak se dělá DDoS. 2013, [Cited 2017-03-27]. Dostupné z: https://www.root.cz/clanky/pronajmu-botnet-cenik-prilozen-aneb-jak-se-dela-ddos/

[6] *BetaBooter - The Luxury IP Stresser*. [Cited 2017-04-10]. Dostupné z: https://betabooter.com/

[7] Low Orbit Ion Cannon. [Cited 2017-05-08]. Dostupné z: https://en.wikipedia.org/wiki/Low_Orbit_Ion_Cannon

[8] *SYN flood*. [Cited 2017-04-10]. Dostupné z: https://en.wikipedia.org/wiki/SYN_flood

[9] *Cloudflare - The Web Performance, Security Company*. [Cited 2017-05-10]. Dostupné z: https://www.cloudflare.com/

[10] Jati, G.; Hartadi, B.; Putra, A. G.; aj.: Design DDoS attack detector using NTOPNG. In *2016 International Workshop on Big Data and Information Security (IWBIS)*, Oct 2016, s. 139–144, doi:10.1109/IWBIS.2016.7872903.

[11] ARP Poisoning. [Cited 2017-04-10]. Dostupné z: `http://hakipedia.com/index.php/ARP_Poisoning`

[12] What is a DDoS Scrubbing Center? [Cited 2017-05-11]. Dostupné z: `https://stratusly.com/what-is-a-ddos-scrubbing-center/`

[13] Krčmář, P.: CESNET vyvíjí vlastní pračku na DDoS útoky. 2016, [Cited 2017-03-25]. Dostupné z: `https://www.root.cz/clanky/cesnet-vyviji-vlastni-pracku-na-ddos-utoky/`

[14] *Liberouter*. [Cited 2017-04-10]. Dostupné z: `https://www.liberouter.org/`

[15] *FASTNETMON DDOS DETECTION TOOL*. [Cited 2017-05-10]. Dostupné z: `https://fastnetmon.com/`

[16] Odintsov, P.: FastNetMon. 2017, [Cited 2017-04-01]. Dostupné z: `https://github.com/pavel-odintsov/fastnetmon/blob/master/README.md`

[17] *Protecting news from digital attacks*. [Cited 2017-04-01]. Dostupné z: `https://projectshield.withgoogle.com/public/`

[18] *How to apply for Project Shield*. [Cited 2017-04-01]. Dostupné z: `https://support.google.com/projectshield/answer/6358116?hl=en&ref_topic=6358107`

[19] Krčmář, P.: NIX.CZ připravuje obranu proti útokům na české sítě. 2013, [Cited 2017-03-30]. Dostupné z: `https://www.root.cz/clanky/nix-cz-pripravuje-obranu-proti-utokum-na-ceske-site/`

[20] Lasek, P.: Intrusion prevention system. 2005, [Cited 2017-04-09]. Dostupné z: `http://www.systemonline.cz/clanky/intrusion-prevention-system.htm`

[21] Čejka, T.; Bartoš, V.; Švepeš, M.; aj.: NEMEA: A framework for network traffic analysis. In *2016 12th International Conference on Network and Service Management (CNSM)*, Oct 2016, s. 195–201, doi:10.1109/CNSM.2016.7818417.

[22] UniRec. Dostupné z: `https://github.com/CESNET/Nemea-Framework/tree/master/unirec`

[23] Tvrdík, P.: Vyvažované BVS – AVL a RB stromy. 2014, [Cited 2017-05-06]. Dostupné z: https://edux.fit.cvut.cz/courses/BI-EFA/_media/lectures/bi-efa2015prednaska9-vvs-anim.pdf

[24] Tvrdík, P.: B-stromy a jejich variace. 2014, [Cited 2017-04-25]. Dostupné z: https://edux.fit.cvut.cz/courses/BI-EFA/_media/lectures/bi-efa2014prednaska9-btrees-anim.pdf

[25] *COVERITY SCAN STATIC ANALYSIS*. [Cited 2017-05-12]. Dostupné z: https://scan.coverity.com/

[26] *Common Weakness Enumeration, A Community-Developed List of Software Weakness Types*. [Cited 2017-05-10]. Dostupné z: https://cwe.mitre.org/index.html

[27] *Valgrind*. Dostupné z: http://valgrind.org/

[28] Nemea. [Cited 2017-04-01]. Dostupné z: www.liberouter.org/technologies/nemea/

Příloha A

Seznam použitých zkratek

ARP Address Resolution Protocol

ASN Autonomous System Number

BGP Border Gateway Protocol

BSoD Blue Screen of Death

CWE Common Weakness Enumeration

DDoS Distributed Denial of Service

DNS Domain Name System

DoS Denial of Service

FPGA Field-programmable gate array

HTTP Hypertext Transfer Protocol

ICMP Internet Control Message Protocol

IPv4 Internet Protocol version 4

IPv6 Internet Protocol version 6

ISO/OSI International Standards Organization / Open Systems Interconnection model

ISP Internet Service Provider

LOIC Low Orbit Ion Canon

NEMEA Network Measurements Analysis

SSL Secure Sockets Layer

TCP Transmission Control Protocol

TRAP Traffic Analysis Platform

UDP User Datagram Protocol

UniRec Unified Record

Příloha B

Instalační manuál

Instalační manuál byl sepsán a otestován na Linux Debian 8. Podobnou instalaci je možné provést na některé jiné distribuci, na které je nainstalováno `gcc`, `make` a `dh-autoreconf`

Pro zkompilování a spuštění modulu je třeba doinstalovat knihovny Libtrap a UniRec dostupné z [28].

```
$ tar -xzf ddos_detector-1.0.2.tar.gz -C [cilove umisteni]
$ cd [cilove umisteni]/ddos_detector-1.0.2
$ autoreconf -i
$ ./configure
$ make
```

B.1 Spuštění modulu

Modul se spouští příkazem

```
$ ./ddos_detector [parametry_detektoru]
```

Pro vypsání nápovědy použijeme přepínač `-h`.

Příloha C

Obsah přiloženého CD

- `README.txt` ..... stručný popis obsahu CD
- `ddos_detector-1.0.2.tar.gz` ..... zdrojové kódy implementace
- `text` ..... text práce
  - `BP_Hollmann_Otto_2017.pdf` ..... text práce ve formátu PDF
  - `src` ..... zdrojová forma práce ve formátu LaTeX