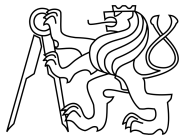
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Automatizovaný záchyt síťových dat k detekovaným událostem
Student:	Bc. Zdeněk Rosa
Vedoucí:	Ing. Tomáš Čejka
Studijní program:	Informatika
Studijní obor:	Systémové programování
Katedra:	Katedra teoretické informatiky
Platnost zadání:	Do konce zimního semestru 2017/18

Pokyny pro vypracování

Nastudujte principy monitorování počítačových sítí založené na sledování paketů a síťových toků.
Navrhněte architekturu monitorovací infrastruktury se systémem pro detekci, která bude zahrnovat využití programovatelných hardwarově akcelerovaných monitorovacích sond [1] pro zachytávání vybraného provozu.
Realizujte napojení výsledků detekčního systému [2,3] na monitorovací sondy [4] pro automatické spouštění zachytávání důkazního materiálu.
Rozšiřte programovatelnou monitorovací sondu o průběžné ukládání krátkodobé historie provozu s možností výběru dat relevantních k detekovaným událostem.
Vytvořený systém otestujte s použitím různých detekčních (existujících) modulů.

Seznam odborné literatury

[1] https://www.liberouter.org/technologies/cards/
[2] https://github.com/CESNET/NEMEA
[3] Bartoš, Václav, Martin Žádník, and Tomáš Čejka. *Nemea: Framework for Stream-Wise Analysis of Network Traffic*. Technical Report, CESNET, a.l.e., 2013.
[4] https://www.flowmon.com/en/products/flowmon/probe

L.S.

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

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 20. září 2016

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

Diplomová práce

Automatizovaný záchyt síťových dat k detekovaným událostem

Bc. Zdeněk Rosa

Vedoucí práce: Ing. Tomáš Čejka

6. ledna 2017

Poděkování

Za odborné a metodické vedení, cenné rady a připomínky při tvorbě diplomové práce upřímně děkuji Ing. Tomáši Čejkovi. Za pomoc při nasazení výsledného systému děkuji především Ing. Janu Kučerovi a RNDr. Petru Velanovi. Za poskytnutí potřebných nástrojů a dat pro tvorbu této práce děkuji organizaci CESNET, z. s. p. o., Nakonec bych rád poděkoval své rodině a přítelkyni za jejich psychickou podporu během mého studia a během psaní diplomové práce.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) 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 6. ledna 2017 .....

Odkaz na tuto práci

Rosa, Zdeněk. *Automatizovaný záchyt síťových dat k detekovaným událostem.* Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Vysokorychlostní sítě pro analýzu provozu a detekci anomálií často používají nástroje založené na síťových tocích. Po nahlášení bezpečnostní události pomocí těchto nástrojů nám chybí důkazní materiál a informace pro verifikaci události. Tato práce rozšiřuje klasickou detekci založenou na síťových tocích o detekci pomocí paketů. Navržený systém umožňuje získat provoz podezřelých adres, podílejících se na události a následně provést analýzu tohoto provozu.

Klíčová slova záchyt paketů, vysokorychlostní sítě, monitorovací sondy, detekční systémy

Abstract

High-speed networks usually use flow based approaches for network measurement and anomaly detection. When an event is reported, there is no evidence and no information for event verification. This thesis extends flow based detection approach by packet based detection. The proposed system enables to retrieve traffic of suspicious address involved in the reported event and automatically analyze it by packet based monitoring tools.

Keywords packet capture, high-speed networks, monitoring probes, detection systems

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Sítová bezpečnost je v dnešní době velice důležitým aspektem, na který bychom neměli zapomínat. Většina z nás bere internet jako nedílnou součást života. Využíváme jej na mobilních telefonech, počítačích, televizích a mnoha dalších zařízeních. Při komunikaci na internetu bychom neměli opomíjet i jeho negativní stránku a měli bychom se snažit vyvarovat nástrahám, které nám hrozí.

Útočníci hledají slabiny v zabezpečení počítačových sítí pro vytváření různých typů útoků. Často se jedná o zneužití principů počítačové sítě pro vlastní prospěch anebo za účelem omezení či znepřístupnění síťových služeb.

Analýza síťového provozu a následné oznámení či zamezení škodlivého provozu je důležitou součástí počítačové bezpečnosti, která napomáhá předcházet bezpečnostním incidentům nebo alespoň omezit škody způsobené bezpečnostními incidenty. Existuje mnoho nástrojů, které mohou být využívány na koncových uzlech komunikace, uvnitř lokální sítě, na hranicích lokální sítě nebo na specifických bodech páteřní sítě.

Analýza provozu páteřní sítě s propustností několik desítek Gb/s v reálném čase vyžaduje mnoho výpočetních prostředků a zároveň omezuje výpočetní složitost algoritmů analýzy. Proto je často využita redukce informací, která vede ke snížení objemu analyzovaného provozu. Redukce informací může být zprostředkována například pomocí síťových toků, které shrnují informace o komunikaci dvou zařízení. Pro jejich vytváření jsou na vysokorychlostních sítích využívány speciální hardwarově akcelerované monitorovací sondy, které dokáží zpracovat provoz sítě s propustností až 100 Gb/s.

Nástroje pro analýzu síťových toků, po nalezení anomálie, nahlásí podezřelé IP adresy a informace spojené s detekcí. Tyto informace často nestačí k posouzení, zda se jedná o falešné hlášení, protože neobsahují data dokazující přenos škodlivého provozu. Navíc v době nahlášení události může být podezřelá komunikace již ukončena, tudíž následný záchyt dat nemusí být dostačující.

Cílem práce je navrhnout a implementovat systém pro automatizované zís-

kání dat nahlášených událostí. Data událostí budou sloužit k automatizované paketové analýze, manuální forenzní analýze, posouzení správnosti detekce, shromáždění důkazního materiálu a k ladění či dalšímu vývoji detekčních algoritmů.

Přínosy této práce pokrývají několik oblastí.

- Návrh a implementace **kruhového bufferu** pro ukládání a vyhledávání paketů vysokorychlostní sítě.
- Rozšíření monitorovací sondy o automatizovaný **živý záchyt** a kruhový buffer umožňující **získání historických paketů** z doby před spuštěním živého záchytu.
- Návrh **komunikačního protokolu** pro ovládání rozšířených monitorovacích sond.
- Návrh a implementace **správce událostí** pro distribuovanou správu infrastruktury monitorovacích sond.
- **Propojení** správce událostí s detekčním systémem.

Systém byl vytvořen ve spolupráci s organizací CESNET, z. s. p. o.

Práce je rozdělena do sedmi částí:

Kapitola 1 popisuje existující systémy pro monitorování a analýzu sítového provozu. Kapitola 2 popisuje analýzu provozu sítě CESNET2, na které bude nasazen a testován výsledný systém pro automatický záchyt dat nahlášených událostí. Kapitola 3 popisuje existující řešení automatizovaného záchytu dat událostí a porovnává jejich využití na síti CESNET2. Kapitola 4 popisuje návrh rozšíření monitorovací sondy o automatizovaný záchyt dat nahlášených událostí. Kapitola 5 popisuje návrh manažera událostí pro distribuované ovládání monitorovacích sond. Kapitola 6 popisuje implementaci rozšíření monitorovací sondy. Kapitola 7 popisuje implementaci prvků pro ovládání rozšířených monitorovacích sond. Mezi tyto prvky patří manažer událostí, klientská aplikace a modul detekčního systému pro komunikaci se správcem událostí. Kapitola 8 popisuje výsledky testování rozšířené monitorovací sondy na prostředcích a datech sítě CESNET2. Kapitola 9 analyzuje zachycená data nahlášených událostí pomocí rozšířených monitorovacích sond a správcem událostí.

Kapitola 1

Analýza monitorování počítačových sítí

Monitorování počítačových sítí je založeno na průběžném zpracování a analýze dat na síti. K monitorování dochází na místech sítě, které představují největší riziko škodlivého provozu. Většinou se jedná o hraniční body sítě. Monitorováním můžeme odhalovat bezpečnostní rizika a incidenty, které se v síti nacházejí. Včasným nahlášením bezpečnostních incidentů můžeme předcházet následkům útoků, mezi které může patřit znepřístupnění, zahlcení, podvržení identity a mnoho dalších. V sítích s provozem o maximální rychlosti kolem 1 Gb/s není problém analyzovat všechny pakety provozu. Na vysokorychlostních sítích již není možné detailně analyzovat všechny pakety, proto dochází ke vzorkování provozu nebo k agregaci informací pomocí síťových toků.

1.1 Síťové toky

Síťové toky [1] obsahují agregované informace o komunikaci dvou uzlů v síti, ohraničené počátečním a koncovým časem. Pakety jsou agregovány do toků podle zdrojové IP adresy, cílové IP adresy, zdrojového portu, cílového portu a protokolu. Ukončení agregace paketů a následný export toku je proveden po ukončení komunikace nebo po uplynutí fixního intervalu. Konkrétní obsah a zpracování síťového toku závisí na použitém protokolu a jeho verzi, například NetFlow verze 9 [2] nebo IPFIX [3]. Agregace informací z paketů do síťových toků umožňuje monitorovat a analyzovat chování sítě s menšími výpočetními nároky. Základní informace poskytované síťovými toky jsou: zdrojová/cílová IP adresa, zdrojový/cílový port, protokol, čas začátku toku, čas konce toku, počet paketů, počet bajtů.

Pro detekci určitých typů útoků nejsou informace poskytované základními síťovými toky dostačující. Proto se využívají rozšířené síťové toky, které obsahují další informace z protokolů vyšších vrstev. Množina údajů exportovaných

z paketů do síťových toků se liší na základě protokolu. Například u DNS se může jednat o doménové jméno, typ požadavku a další. Díky těmto informacím lze zpřesnit detekci či vytvořit nové detekční algoritmy a konkurovat tak nástrojům založeným na analýze paketů.

1.2 Extrakce dat ze sítě

Výběr způsobu a prostředků pro export dat ze síťové linky do softwarového prostředí závisí především na propustnosti měřené linky. Dle [4] je standardní monitorování založeno na síťové kartě, která přenáší všechna data do softwaru, kde dochází k samotnému zpracování paketů nebo počítání síťových toků. Maximální propustnost zpracování dat je určena propustností síťové karty, použité sběrnice, výpočetním výkonem procesoru a použitým softwarem. Pro zpracování všech dat síťového provozu, musí být propustnost systému vyšší než maximální rychlost měřené linky.

1.2.1 Vzorkování

V případě, že hardware nestíhá přenášet všechna data do softwaru, může být použito vzorkování provozu. Několik různých způsobů vzorkování je popsáno v článku [5]. Za nejhorší variantu vzorkování, pro monitorování sítě, je považováno naivní rovnoměrné vzorkování provozu. To exportuje především pakety dlouhých toků. Doporučuje se zvolit metodu založenou na určité heuristice, například vzorkování podle velikosti paketu, adresy a další.

1.2.2 HANIC

Systém HANIC neboli Hardware Accelerated Network Interface Card popsaný v [4] umožňuje přenášet všechny pakety do softwarového prostředí a to až při propustnosti 100 Gb/s. Systém je založen na síťové kartě s FPGA (Field Programmable Gate Array), operačním systémem Linux, ovladači karty a nástroji pro vysokorychlostní zpracování paketů. Karta je připojena k základní desce pomocí rozhraní PCIe x16, které umožňuje do softwaru přenést takto velký objem dat. Firmware karty umožňuje:

- Filtrování paketů na základě IP adres, protokolu nebo portů. Vhodné, pokud chceme analyzovat pouze určité komunikace.
- Vzorkování s poměry 1:1, 1:2, 1:4, 1:8 nebo 1:16. Vhodné, pokud software nestíhá zpracovávat příchozí pakety.
- Redukci (zastřižení) velikosti paketů. Vhodné například pro výpočet toků, kde nás zajímá jen hlavička paketu.
- Parsování paketů. Karta umožňuje do softwaru odesílat již vyparsovaná políčka z hlavičky paketu.

Firmware distribuuje pakety do osmi DMA kanálů, pro maximální využití více jádrových procesorů. Distribuce je založená na hašovací funkci, která rozděluje pakety jednotlivých toků do nezávislých kanálů (pakety jednoho toku jsou ve stejném kanálu). Každý kanál zapisuje do vlastního kruhového bufferu alokovaného v paměti RAM. Pakety z bufferu jsou čteny a zpracovávány příslušným vláknem procesoru. Tento přístup umožňuje nezávislost komunikace mezi kartou a softwarem. Pokud software nestíhá zpracovávat pakety z bufferu, dojde k zaplnění bufferu a nově příchozí pakety jsou zahazovány.

V softwaru je možné s pakety libovolně pracovat, například počítat síťové toky, ukládat, filtrovat atd. Hlavním omezením je rychlost zpracování dat v softwaru, která nesmí dlouhodobě klesnout pod rychlost přijímání nových dat. To může být při propustnosti linky 100 Gb/s zásadní problémem, který vede k zahazování nově příchozích paketů.

1.2.3 Software Defined Monitoring

Hlavní myšlenka Software Defined Monitoring (SDM) je v rozložení výpočetní zátěže mezi softwarem a hardwarem [6], [7]. Hardware na základě dynamicky nastavených pravidel vybírá pakety, které agreguje na síťové toky bez nutnosti jejich přenášení do softwaru. Zbytek paketů je přeposlán a libovolně zpracován v softwaru. Díky tomuto principu je možné detailně analyzovat určité množství paketů a zároveň počítat síťové toky na lince s maximální propustností do 100 Gb/s.

Pakety nových síťových toků jsou přenášeny do softwaru, kde je zahájena jejich agregace do síťových toků. Kromě agregace je možné s těmito pakety dále pracovat. Software má po zpracování potřebných paketů toku možnost přidělit zbytek výpočtu síťového toku hardwaru. Nadále tak nebude přijímat žádné pakety tohoto toku. Přidělení výpočtu probíhá zasláním speciálního pravidla obsahující informace o toku. Síťová karta dopočítá zbytek toku v hardwaru a do softwaru odešle jen výsledný tok. Software poté zkombinuje vypočtené informace ze začátku toku (vypočítáno v softwaru) a konce toku (vypočítáno v hardwaru) do jednoho síťového toku.

Systém SDM se skládá ze tří vrstev: hardware, firmware a software. Obrázek 1.1 znázorňuje propojení jednotlivých vrstev.

První vrstva, hardware, obsahuje síťovou kartu s FPGA čipem. Používají se karty s jedním či dvěma porty (80–100 Gb/s), čipem Virtex-7 FPGA a platformou NetCOPE [8]. Přehled všech podporovaných karet s Virtex-7 FPGA čipem je dostupný na [9]. Hlavním úkolem této vrstvy je příjem dat z fyzického média a komunikace se softwarem.

Druhá vrstva, FPGA firmware, obsahuje konfiguraci FPGA čipu pro zpracování paketů v hardwaru. Z hlavičky každého přijatého paketu jsou extrahována potřebná metadata a na základě pravidel je rozhodnuto o dalším zpracování paketu. Podporované operace jsou:

1. Přeposlání paketu. Paket je beze změn přeposlán do softwaru. (Základní a nejvíce vytěžující operace pro datovou sběrnici a procesor počítače).

2. Redukce velikosti paketu. Paket může být oříznut na stanovenou délku a odeslán do softwaru. Vhodné především pro velké pakety, jejichž obsah není softwarem využit.

3. Parsování paketu. Informace z hlavičky paketu jsou extrahovány a přeneseny do softwaru. Výhodou je, že do softwaru nemusí být přenesen celý paket, ale jen potřebné informace. Díky tomu se sníží využití prostředků pro komunikaci a zpracování v softwaru.

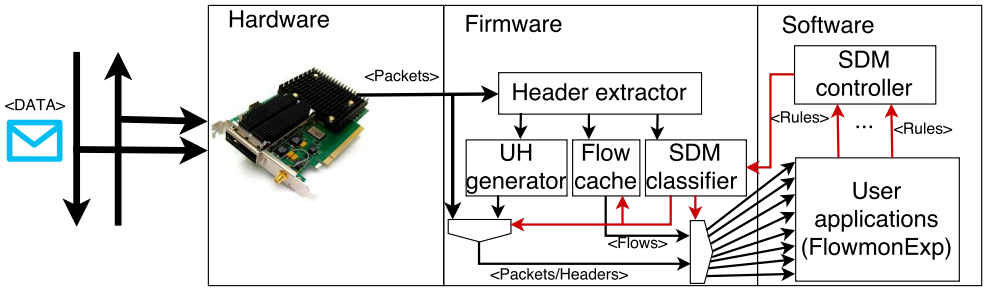
4. Výpočet toku v hardwaru. Informace o paketu jsou připočteny, k již vytvořenému toku nebo je v mezipaměti síťových toků vytvořen nový záznam o toku. Po ukončení výpočtu toku je tok odeslán do softwaru.

5. Zahození paketu. Paket je bez jakéhokoliv zpracování zahozen. Neobjeví se tedy ve výpočtu toků, ani není zaslán do softwaru.

Třetí vrstva, software, se skládá z nástrojů pro základní nastavení firmware, zasílání agregovaných pravidel (pro manipulaci s pakety) a příjem dat (paketů, toků či hlaviček).

Nástroj pro zasílání SDM pravidel do firmware přijímá od ostatních aplikací požadavky na manipulaci s pakety specifického toku. Pokud nástroj příjme pro jeden tok více požadavků s rozdílnými operacemi, provede agregaci požadavků do jednoho pravidla. Agregace se vždy snaží o maximalizaci informací, tudíž je odesláno pravidlo s nejméně destruktivní operací. Ve výše popsaném seznamu jsou pravidla seřazena podle priority (pravidla s nižším číslem mají vyšší prioritu).

Pro příjem a zpracování paketů v softwaru může být použita například aplikace Flowmonexp popsaná v sekci 2.1.2.



Obrázek 1.1: SDM design

1.3 Detekční systémy

Detekční systémy analyzují provoz sítě přímo z paketů nebo z již agregovaných síťových toků. Analýza pomocí paketů může být detailnější, ale zato výpočetně náročnější. V následujícím textu jsou popsány tři detekční systémy, první dva jsou založeny na monitorování paketů, třetí na monitorování síťových toků.

1.3.1 Snort

Detekční systém Snort popsaný v [10] je založen na knihovně libpcap [11], která zprostředkovává aplikační rozhraní pro práci s pakety nezávislé na platformě. Hlavními funkcemi Snortu jsou selektivní záchyt paketů a detekce anomálií. Detekce využívá knihovnu pravidel, podle které probíhá filtrování jednotlivých paketů. Událost je nahlášena po splnění všech podmínek pravidla. Knihovna je jednoduše editovatelná, uživatel tak může libovolně deklarovat nová pravidla nebo upravovat stávající pravidla. Pro deklaraci pravidel je použit jednoduchý jazyk, který se skládá z filtrů a akcí. Snort umožňuje události nahlašovat do syslogu, zasílat zprávou ve formátu Server Message Block (SMB) nebo je ukládat do souboru.

Hlavní slabinou Snortu je propustnost, která je podle [12] maximálně 500 Mb/s při použití knihovny libpcap a komoditního hardwaru. Při použití speciální karty s hardwarovou akcelerací pro Snort je možné zvýšit propustnost až na 10 Gb/s. Doba zpracování paketu je ovlivněna především hloubkou analýzy vyšších aplikačních vrstev a počtem aplikovaných pravidel.

1.3.2 Bro

Detekční systém Bro popsaný v [13] podporuje komoditní síťové karty a platformy Linux, FreeBSD a MacOS. Obsahuje knihovnu detekčních skriptů, které jsou interpretrem zpracovány a aplikovány na jednotlivé pakety. Pomocí skriptů lze implementovat sofistikovanější detekční mechanismy, které nelze vyjádřit jednoduchým pravidlem. Díky tomu se jedná o silnější, ale výpočetně náročnější detekční nástroj, než je Snort.

Bro nedisponuje možností vícevláknového běhu, zato jeho architektura umožňuje distribuovat detekci na více výpočetních uzlů. Distribuovaná detekce je použita na vysokorychlostních sítích, které vyžadují větší propustnost. Bro implementuje pouze distribuci a správu procesů na výpočetních uzlech, nikoliv zařízení pro rozdělení dat do jednotlivých procesů (frontend). Dle [12], Bro zvládá propustnost do 1 Gb/s pro jednu instanci. Navýšení propustnosti lze docílit distribuováním na více výpočetních uzlů. Konkrétní propustnost závisí na použitých skriptech pro detekci.

1.3.3 NEMEA

Detekční systém NEMEA (Network Measurement Analysis) popsaný v [14], je heterogenní modulární systém založený na analyzování síťových toků v reálném čase. Vstupem systému mohou být soubory (ve formátech csv [15] nebo nfdump [16]), síťové rozhraní nebo síťové toky (ve formátu UniRec [17]).

Moduly jsou spuštěny jako nezávislé procesy, které mohou být mezi sebou jednosměrně propojeny pomocí unifikovaného rozhraní. Unifikace je zajištěna použitím knihoven NEMEA framework [17]. Knihovna TRAP (Traffic Analysis Platform) poskytuje vysokoúrovňové komunikační rozhraní a Knihovna UniRec (Unified Record) poskytuje efektivní datový formát pro zasílání zpráv. O správu, běh a monitorování jednotlivých modulů se stará modul supervisor, popsaný v [18].

Systém NEMEA obsahuje sadu detekčních modulů, které umožňují detekovat různé anomálie na síti. Některé z nich jsou popsané v sekci 2.1.4. Po nalezení podezřelého provozu je tato událost odeslána na výstupní rozhraní modulu, kde může být zpracována dalším modulem. Pro hlášení událostí jsou určeny exportní moduly, které ukládají události do databáze, souboru, zasílají emailem atd. NEMEA dále obsahuje manipulační moduly určené pro filtrování toků, ukládání toků v různých formátech, replikování souborů s toky atd.

Propustnost systému NEMEA je závislá na použitých modulech a zařízení pro export síťových toků ze sítě. Moduly lze spouštět na různých výpočetních uzlech, tak aby jako celek zvládaly potřebnou propustnost.

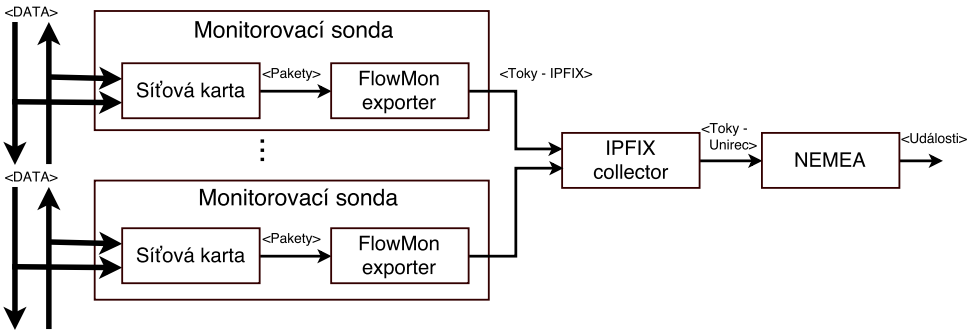
Kapitola 2

Analýza provozu sítě CESNET2

Práce je cílená k využití na vysokorychlostních sítích, které pro analýzu a monitorování provozu používají monitorovací sondy exportující síťové toky. Tyto požadavky splňuje síť CESNET2 [19], jejíž stávající monitorovací infrastruktura a prostředky jsou základem pro rozšíření o automatizovaný záchyt vybraného provozu. Na síti se používají jak klasické monitorovací sondy ke zpracování provozu linky v softwaru, tak monitorovací sondy se systémem SDM. V této kapitole jsou popsány jednotlivé prvky infrastruktury a je provedena analýza provozu, která stanovuje základní požadavky navrhovaného systému.

2.1 Monitorovací infrastruktura sítě CESNET2

Základ Monitorovací infrastruktury se skládá ze tří částí (Monitorovací sondy, IPFIX collector a NEMEA) znázorněných na obrázku 2.1. Všechny tyto části jsou základem pro navrhovaný systém a jsou detailně popsány v této sekci.

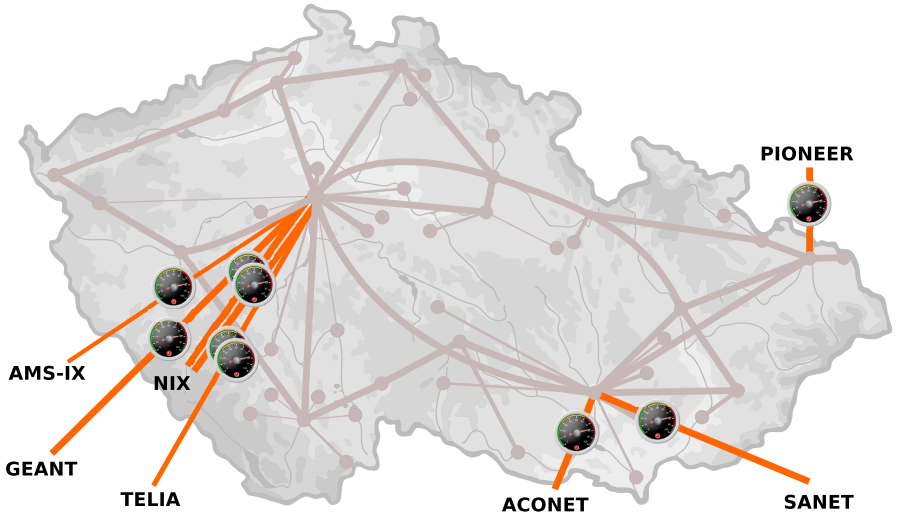


Obrázek 2.1: Monitorovací infrastruktura sítě CESNET2

2.1.1 Monitorovací sondy

Sondy se nacházejí na hraničních bodech sítě. Obrázek 2.2 znázorňuje topologii sítě a umístění jednotlivých sond.

Monitorovací sonda se skládá se dvou částí: síťové karty a softwaru Flowmonexp [20]. Hlavním úkolem monitorovací sondy je přijímat data ze sítě, rozdělit je na pakety a následnou agregací paketů je převádět na síťové toky. Sonda umožňuje vytvářet jak klasické síťové toky, tak rozšířené síťové toky s informacemi z aplikační vrstvy. Výstupem jsou záznamy toků ve formátu IPFIX nebo NetFlow.



Obrázek 2.2: Topologie sítě CESNET2

Sondy jsou vybavené síťovými kartami se systémem SDM (viz sekce 1.2.3), který umožňuje softwarově zpracovávat vybrané komunikace, nebo systémem HANIC (viz sekce 1.2.2), který umožňuje softwarově zpracovávat linky s propustností až 100 Gb/s. V softwaru jsou data přijímány aplikací Flowmonexp.

2.1.2 Flowmonexp

Flowmonexp (Flowmon Exporter) je software vyvíjený společností Flowmon Networks a.s. [20]. Jeho základní funkcí je příjem dat ze síťové karty a následný výpočet síťových toků (standardních i rozšířených).

Flowmonexp je navržen jako vícevláknová aplikace, která díky využití více jader procesoru umožňuje zpracovávat vysoké objemy dat. Každé vlákno se stará o načítání a zpracování dat z jednoho DMA zásobníku. Distribuci dat do jednotlivých zásobníků zajišťuje přímo síťová karta. Výstupem aplikace

Flowmonexp jsou síťové toky ve formátech NetFlow nebo IPFIX. V popisované infrastruktuře je použit formát IPFIX.

Aplikace je rozšířitelná o zásuvné moduly, které umožňují libovolnou manipulaci s daty. Zásuvné moduly se dělí do tří kategorií: vstupní, procesní a výstupní.

Při použití síťové karty s SDM je možné pomocí zásuvných modulů Flowmonexp ovlivňovat data, která se dostanou do softwaru. Pro softwarové zpracování n paketů ze začátku toku je použit jednoduchý vstupní zásuvný modul. Ten po přijetí n paketů toku zašle SDM pravidlo pro zpracování zbytku toku v hardwaru. Pokud není současně odeslána jiná žádost pro tento tok, zbytek toku je vypočítán v hardwaru a Flowmonexp tyto pakety již nepřijme.

2.1.3 IPFIXcol

IPFIXcol (IPFIX collector) [21] vyvíjený výzkumnou skupinou Liberouter, je aplikace sloužící pro příjem síťových toků, jejich zpracování, modifikaci a export. Aplikace umožňuje připojení mnoha zásuvných modulů, které umožňují rozšíření funkcionality.

IPFIXcol je v popsané monitorovací infrastruktuře použit především pro evidenci síťových toků a jejich konverzi z formátu IPFIX do formátu UniRec, proprietárního pro systém NEMEA. IPFIXcol aktuálně exportuje základní síťové toky všech protokolů a rozšířené síťové toky protokolů DNS, SIP, HTTP a SMTP.

2.1.4 NEMEA

NEMEA systém, popsaný v sekci 1.3.3, je v infrastruktuře použit pro detekci anomálií na základě síťových toků. Navržený systém záchytu dat bude testován na událostech detekovaných různými moduly systému NEMEA. Z tohoto důvodu jsou zde stručně popsány použité detektory a jimi rozpoznávané útoky.

VoIP fraud detector

Voice over IP (VoIP) [22] je protokol určený pro zprostředkování hlasového či video hovoru po IP sítích. Pro zahájení, správu a ukončení hovoru se používá protokol SIP [23]. Spojení s klasickými PSTN sítěmi je realizováno pomocí PSTN brán, které se většinou nacházejí uvnitř organizací. Nezabezpečené PSTN brány umožňují přepojit hovor na základě zadání tajného prefixu před číslo volajícího.

VoIP fraud je útok, kdy se útočník snaží uhádnout tajný prefix PSTN brány. Zkouší proto vytáčet různé kombinace prefixů, dokud není hovor spojen. Následně útočník využívá znalosti prefixu voláním na placené telefonní linky a způsobuje tak organizaci finanční ztrátu. V případě sofistikovanějšího útoku, se útočník se snaží skrýt svou aktivitu a testuje jednotlivé prefixy s delším časovým odstupem. Takovéto útoky mohou trvat až několik dní. Čím delší je

časový odstup, tím je složitější útok detekovat. Detekční algoritmus modulu je popsaný v [24].

Vstupem detektoru jsou rozšířené síťové toky o záznamy SIP protokolu. Poskytnuté informace jsou dostačující i k pozdějšímu ověření správnosti detekce. Záchyt provozu detekované adresy však může odhalit další podezřelé aktivity.

DNS tunnel detector

DNS (Domain Name System) [25] je hierarchický systém doménových jmen používající stejnojmenný protokol pro výměnu informací. Hlavním účelem DNS je překlad mezi IP adresami a doménovými jmény.

Protokol DNS je častým cílem různých útoků, jedním z nich je tunelování protokolu. Útočník nacházející se v síti s omezeným provozem kóduje data do DNS žádostí a zasílá je na svůj tunelovací server umístěný volně v internetu. Firewall neblokuje tento požadavek, protože se pro něj jedná o legitimní DNS provoz. Požadavek je doručen až k tunelovacímu serveru, který data dekóduje, vykoná jimi požadovanou akci a zpět posílá zakódovanou odpověď. Detekční algoritmus modulu je popsaný v [26]. Tento detektor navíc umožňuje rozpoznat DoS útok přes DNS, kdy je ze zdrojové adresy odesíláno velké množství dotazů na překlad stejného doménového jména.

Vstupem detektoru jsou rozšířené síťové toky o záznamy DNS protokolu. Podobně jako u VoIP fraud detektoru jsou poskytnuté informace dostačující k pozdějšímu ověření správnosti detekce. Záchyt provozu detekované adresy může být použit pro odhalení další podezřelé aktivity či bližší zkoumání kódování přenášených dat.

DNS amplification detector

DNS protokol popsaný u předchozího detektoru, umožňuje odesláním vhodného dotazu na volně dostupný DNS server vygenerovat odpověď řádově větší, než je samotný dotaz. Tato vlastnost může být využita pro generování DDoS (Distributed Denial of Service) útoku.

Detekce probíhá na základě krátkodobého sledování komunikace dvou klientů. Porovnáním zaslaných a přijatých dat každé komunikace je určeno, zda se jedná o útok. Vstupem detektoru jsou pouze základní síťové toky. Záchytem podezřelých dat tak lze verifikovat správnost detekce.

Hoststats

V obrovském množství dat přenášených na páteřní síti lze jednoduše zamaskovat různé typy útoků. Zaměřením se na komunikaci každé IP adresy získáme detailní pohled, který umožní detekovat a nahlásit nestandardní chování jednotlivých adres.

Detektor vytváří statistiky komunikace každého individuálního hosta (IP adresy) na síti. Tyto statistiky periodicky sleduje a porovnává je se sadou statických pravidel. V případě porušení pravidla dochází k nahlášení hosta. Detektor obsahuje pravidla pro útoky: DoS (Denial od Service), SSH brute-force a DNS amplifikace.

Vstupem detektoru jsou pouze základní síťové toky. Záchyt podezřelých dat tak může vést k získání cenných údajů pro forenzní analýzu a další vývoj detekčních algoritmů.

IP blacklist filter

Detektor využívá veřejně dostupné seznamy podezřelých IP adres. Tyto seznamy pravidelně aktualizuje a v případě nalezení některé z adres v síťových tocích, které přijímá na vstupu, nahlásí tuto událost.

Detektor nezkoumá chování adresy, záchyt jejího provozu tak může sloužit ke zjištění účelu komunikace a odhalení různých typů útoků.

Vertical port scan detector

Vertikální skenování (skenování portů určité IP adresy) umožňuje vyhledávat otevřené porty a nabízené služby na určité adrese. Samotné prohledání portů není považováno za útok, ale je většinou využito při hledání slabých míst systému pro následný útok. Pro skenování síťových portů existuje mnoho volně dostupných a jednoduše ovladatelných nástrojů. Detekční algoritmus modulu je popsaný v [27]. Vstupem detektoru jsou pouze základní síťové toky. Záchyt provozu detekované adresy tak může odhalit další podezřelé aktivity.

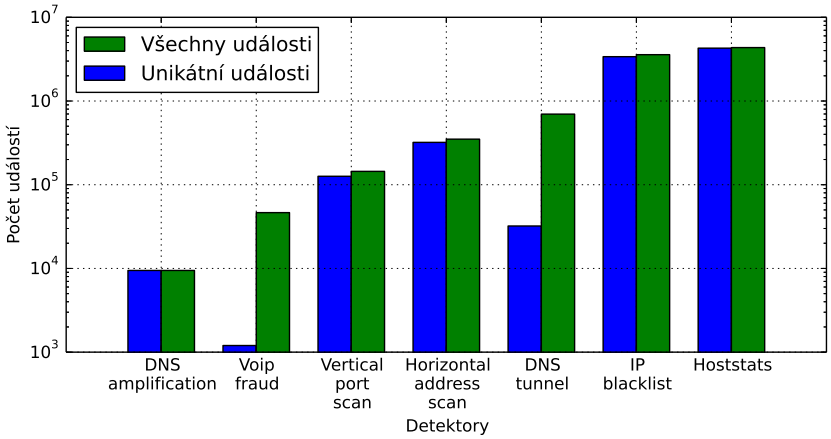
Horizontal address scan detector

Horizontální skenování sítě je principem podobné vertikálnímu skenování. Nedochází však ke skenování mnoha portů na jedné adrese, ale naopak jednoho portu na mnoha adresách. Nejčastěji se jedná o skenování adres určité podsítě. Detektor pracuje podobně jako *Vertival port scan detector*.

2.2 Detekované události

Systém automatizovaného záchytu nahlášených událostí bude získávat události ze systému NEMEA. Pro optimální navržení systému je třeba analyzovat detekované události a stanovit tak minimální požadavky záchytu událostí. Informace v této sekci byly zjištěny analyzováním dlouhodobých záznamů detekčních modulů popsaných v sekci 2.1.4.

Analyzované moduly systému NEMEA vygenerují měsíčně zhruba 7 750 000 unikátních událostí. V případě dlouhodobě trvající nebo opakující se události může být stejná událost reportována vícekrát. Celkový počet reportovaných



Obrázek 2.3: Počet detekovaných událostí za měsíc

událostí je zhruba 10 600 000. To znamená, že každou vteřinu jsou nahlášeny zhruba 4 události. Obrázek 2.3 znázorňuje rozložení událostí na jednotlivé detektory. Není však potřeba ukládat data všech těchto událostí. V mnoha případech se jedná o falešná hlášení, opakující se stejné útoky nebo události, pro které záchyt dat nepřináší další informace. Události bude nutné filtrovat, tak abychom zachytávali pouze data přinášející obohacující informace o událostech.

Důležitým aspektem pro záchyt dat událostí je zpoždění detekce jednotlivých detektorů. To určuje potřebný čas, po který je třeba uchovávat data, aby mohli být v případě nahlášení události získány. Zpoždění bylo měřené jako časový interval mezi přijetím prvního paketu podezřelého síťového toku sondou a nahlášením události detekčním modulem. Zpoždění detekce je ovlivněno několika faktory:

1. Doba exportu síťového toku, která je maximálně 5 minut (aktuální hodnota používaná exportéry).

2. Doba přenosu mezi sondou a detekčním systémem, která je téměř zanedbatelná (méně než 1 s).

3. Doba detekce, která je u jednotlivých detektorů variabilní.

Jednotlivé detektory mají různou dobu detekce, ta závisí na konkrétním algoritmu. Algoritmy filtrující síťové toky na základě stanovených parametrů nahlásí událost ihned po přijetí a kontrole síťového toku. Doba detekce je minimální (≤ 1 s). Jedná se o detektory *Hoststats* a *IP blacklist filter*. Oproti tomu algoritmy založené na sofistikovanějších detekčních metodách, přijímají síťové

toky, ukládají z nich určité informace a na základě shromážděných informací detekují bezpečnostní incidenty. Jedná se o detektory *VoIP fraud detector*, *DNS tunnel detector*, *DNS amplification detector*, *Vertical port scan detector* a *Horizontal address scan detector*

Na obrázku 2.4 je graf znázorňující empirickou distribuční funkci zpoždění detekce jednotlivých detektorů. U modulů *Hoststats* a *IP blacklist filter* je vidět, že zpoždění detekce závisí především na době exportu síťových toků. Naproti tomu ostatní detektory, které shromažďují informace o tocích, mají vyšší detekční zpoždění. Černá čára znázorňuje průměr detekčního zpoždění všech měřených detektorů. Z grafu lze vyčíst, že pro zachycení dat 80 % detekovaných událostí je zapotřebí uložit zhruba 16 minut historického provozu. V takovém provozu by měly být všechny pakety, které se podílely na nahlášení události. V případě získání historického záznamu o délce alespoň 5 minut, máme k dispozici pakety posledního toku, který způsobil odeslání události.

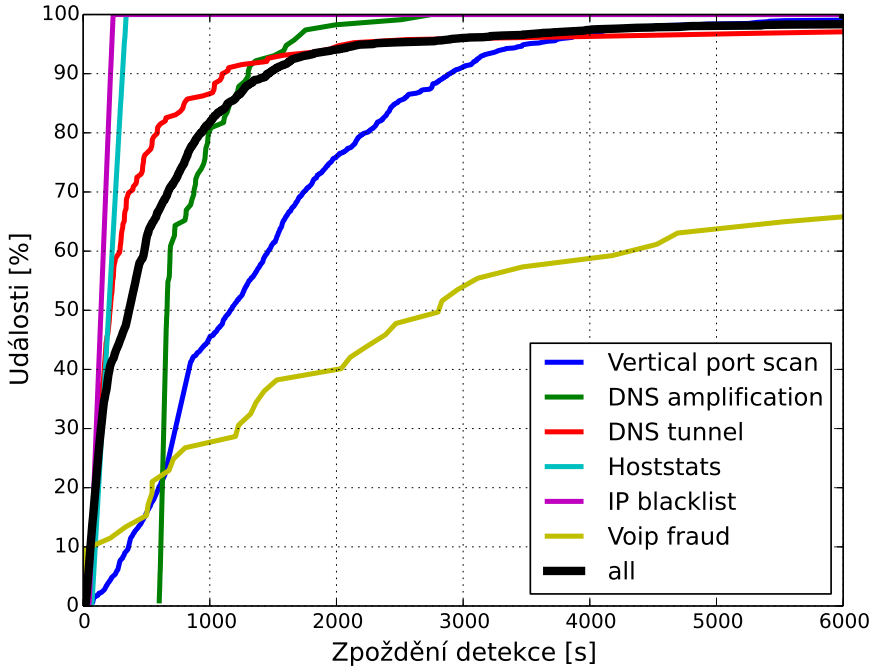
Analýzou detekovaných událostí byly stanoveny základní požadavky na vyvíjený systém. Ten bude krátkodobé ukládat historický provoz, aby byl schopen v případě nahlášení události zprostředkovat relevantní data. Za minimální délku historie, ze které lze získat část paketů podílejících se na události, lze považovat 5 minut. Při délce historie alespoň 16 minut, získáme veškerý provoz 80 procent nahlášených událostí.

2.3 Vliv velikosti začátku toku

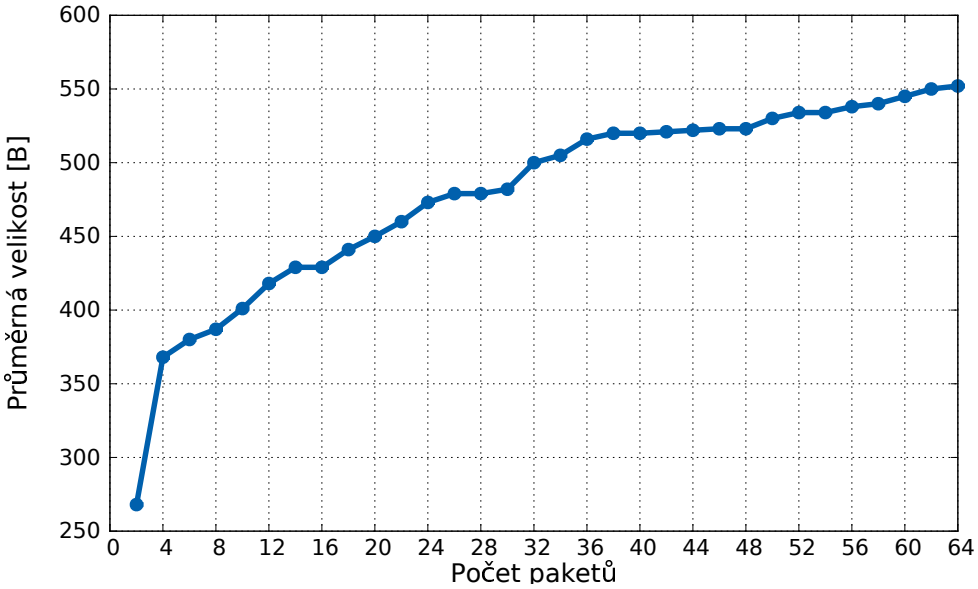
Dle [28], první pakety podezřelého síťového toku obsahují nedůležitější informace pro forenzní analýzu. Ty obsahují inicializaci komunikace, podle které lze například poznat typ požadované služby, úspěšnost spojení atd. Mnoho útoků je charakteristických krátkými toky, jedná se například o útoky hrubou silou, DDoS útoky, skenování portů a další.

Graf 2.5 znázorňuje velikost začátku toku na ose X (počet paketů) vůči průměrné velikosti jeho paketů na ose Y. Je vidět, že pakety krátkých toků (toky s nízkým počtem paketů) jsou průměrně mnohem menší než pakety delších toků. Krátké toky většinou neslouží k přenosu objemných dat, proto není třeba využívat větších délek paketů. Naproti tomu dlouhé toky přenášející objemná data využívají větší velikosti paketů. Tyto data jsou velmi často zašifrovaná, a proto nejsou pro forenzní analýzu přínosem.

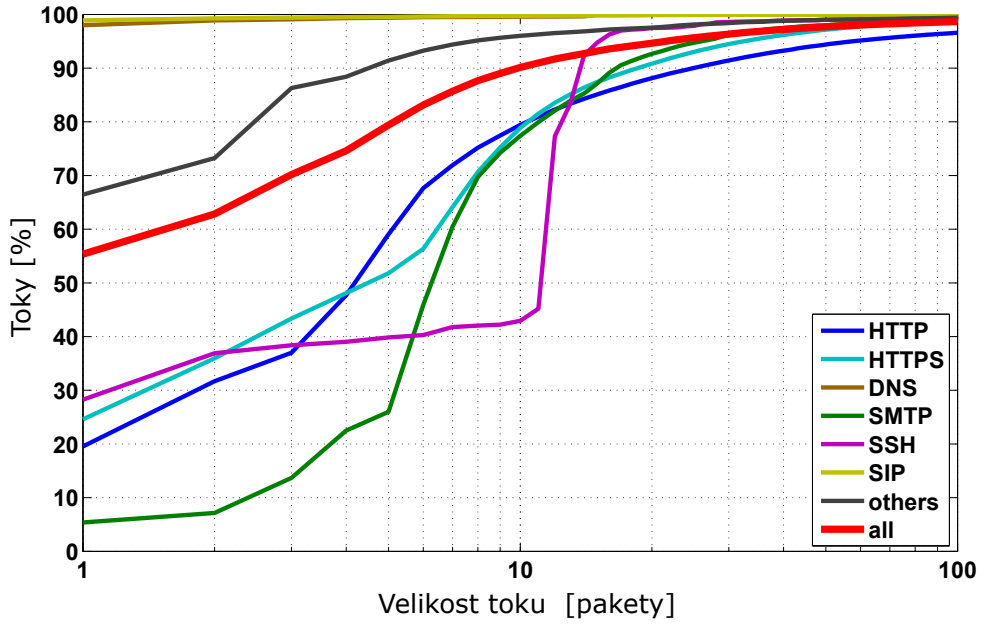
Graf 2.6, převzatý z článku [6], znázorňuje empirickou distribuční funkci počtu paketů síťového toku pro určité komunikační protokoly. Na ose Y je procento toků jejíž počet paketů je stejný nebo menší než hodnota na ose X. Například více jak 99 % všech síťových toků protokolu SIP nebo DNS obsahuje pouze jeden paket. Naproti tomu protokoly jako HTTP nebo SMTP, které přenášejí více dat, fragmentují data do více paketů a zvětšují tak velikost jednoho toku. Z grafu lze vyčíst, že 90 % síťových toků má velikost 10 paketů a méně.



Obrázek 2.4: Zpoždění detekce NEMEA modulů



Obrázek 2.5: Průměrná velikost paketů vůči velikosti začátku toku

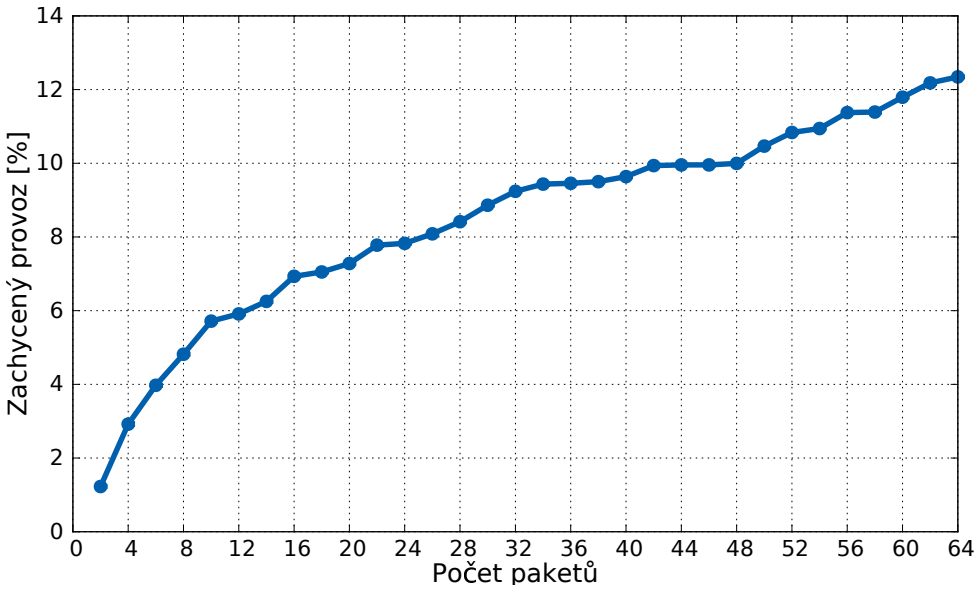


Obrázek 2.6: Velikosti toků v síťovém provozu

Graf 2.7 znázorňuje velikost začátku toku na ose X (počet paketů) vůči celkovému poměru dat na ose Y. Zpracováním začátků toků namísto celých toků se vysoce snižuje celková velikost zpracovaných dat. Tohoto principu využívá systém SDM popsaný v sekci 1.2.3. Například při zpracování pouze prvních 10 paketů síťového toku, zpracováváme pouze 6 % celkového objemu dat.

Z prezentovaných grafů lze dojít k závěru, že většina síťových toků je krátkých (obrázek 2.6). Přesto největší část provozu na síti tvoří dlouhé toky (obrázek 2.7), jejichž pakety jsou mnohem větší než pakety krátkých toků (obrázek 2.5). Navrhovaný systém bude založen na těchto poznatcích. Při vysokém objemu zachytávaných dat se systém bude omezovat na záchyt začátku toků, které budou pro pozdější analýzu nejdůležitější. Díky tomuto omezení dojde k prodloužení ukládané historie, která bude omezena fixní velikostí paměti.

Pro zajištění záchytu zhruba 90 % všech paketů lze zachytávat pouze 10 prvních paketů z každého toku. Pro získání 80 % dat všech detekovaných událostí je třeba krátkodobě uchovávat alespoň 16 minut provozu. Při zanedbání délek toků zachytávaných událostí lze předpokládat, že s těmito parametry zachytíme zhruba $0,9 * 0,8 = 72\,\%$ všech událostí. Pro linku přenášející provoz o maximální rychlosti 1 Gb/s bude třeba krátkodobě uchovávat historii o velikosti zhruba $1\,Gb/s * 60\,s * 16\,min * 0,06 = 57,6\,Gb = 7,2\,GB$ dat. Pro jiné parametry lze potřebnou velikost [MB] vypočítat analogicky pomocí vzorce



Obrázek 2.7: Poměr dat vůči velikosti začátku toku

$\frac{s*d*l}{8}$, kde s je maximální rychlost provozu na lince v Mb/s, d je délka uloženého provozu v sekundách a l je poměr zachyceného provozu při uchovávání prvních n paketů toku (hodnota lze odvodit z obrázku 2.7).

Kapitola 3

Analýza systémů pro záchyt dat událostí

Tato kapitola popisuje několik známých systémů pro ukládání dat nahlášených událostí. Přístup jednotlivých systémů je porovnám s požadavky pro záchyt na síti CESNET2 a použitím detekčního systému NEMEA.

3.1 Bro

Detekční systémy pro paketovou analýzu v reálném čase, jako je například *Bro*, popsaný v sekci 1.3.2, umožňují ukládat provoz po nalezení podezřelé komunikace. Tento přístup dlouhodobě nezatěžuje systémové prostředky (pouze v době záchytu), zato zachycený provoz již nemusí obsahovat požadovaná data.

Tento přístup je pro systém NEMEA nedostačující. Z důvodu detekce na základě síťových toků, jejichž export má zpoždění až 5 minut (v závislosti na délce agregace) a následná detekce může mít další zpoždění, bychom nemuseli zachytit relevantní data k události. Pro záchyt relevantních dat událostí systému NEMEA je nezbytné mít možnost přístupu k historickým datům.

3.2 McAfee Network Security Platform

Přístup k historickým datům umožňuje například nástroj *McAfee Network Security Platform* [29], který ukládá veškerý provoz na pevný disk a poté na něm aplikuje detekci anomálií. Jelikož se jedná o uzavřený software, není možné provést analýzu výkonnosti. Přesto lze předpokládat omezení propustnosti měřené sítě, z důvodu ukládání veškerého provozu na na pevný disk a s tím spojené vytěžování ostatních systémových prostředků. Proto se tento přístup nehodí pro použití na vysokorychlostních sítích.

3.3 Time Machine

Další systém umožňující přístup k historickým datům popisuje článek [28]. Autoři článku pojmenovali systém *Time Machine* neboli stroj času, protože umožňuje přístup k historickému provozu sítě.

Time Machine je navržen pro sítě s rychlostí do 1 Gb/s. Již při této rychlosti je z paměťových a výpočetních důvodů nemožné ukládat všechna data. Při plné rychlosti by bylo třeba uložit zhruba 7,5 GB každou minutu. Systém snižuje objem ukládaného provozu výběrem nejdůležitějších dat z provozu. Je využita vlastnost těžkých toků, která je popsaná již v sekci 2.3. Dochází tak k ukládání pouze prvních n paketů z každého toku.

Time Machine zprostředkovává tyto operace:

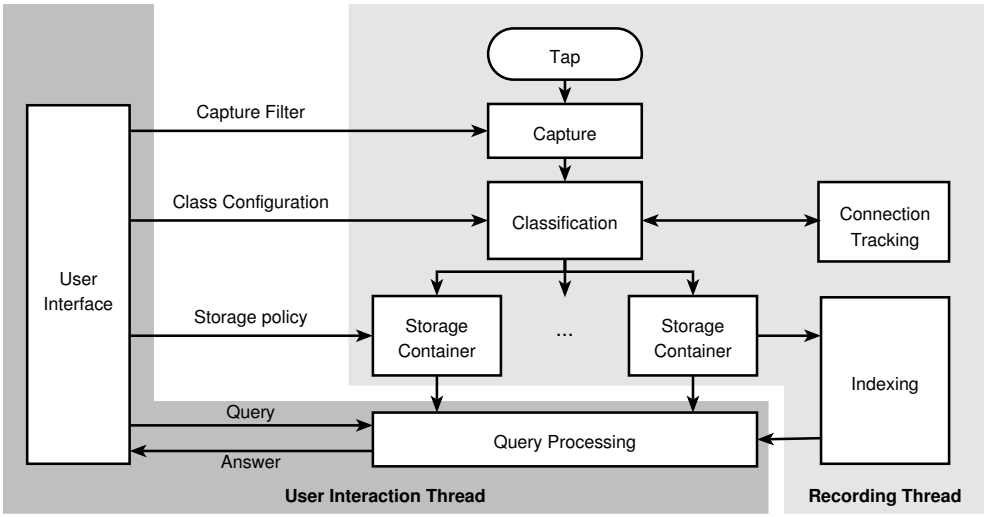
- Ukládání n paketů z každého toku. Systém parsuje pakety, přiřazuje je do síťových toků a do paměti RAM ukládá pouze prvních n paketů toku.
- Přesun paketů z paměti RAM na disk. Po vyčerpání dostupné paměti se nejstarší pakety filtrují a část z nich je nahrána na pevný disk.
- Vyhledávání v uložených datech. Systém zprostředkovává funkci efektivního vyhledávání v uložených datech.
- Změna nastavení. Systém obsahuje uživatelské prostředí pro konfiguraci parametrů záchytu.

Architektura stoje času je znázorněna na obrázku 3.1. Aplikace je rozdělena do dvou vláken. První vlákno zprostředkovává komunikaci s uživatelem a vyhledávání v uložených datech. Druhé vlákno se stará o příjem dat, jejich filtrování a ukládání. Toto rozdělení zajišťuje, že záchyt dat má vždy vyšší prioritu než vyhledávání v datech.

Vlákno uživatelského rozhraní (první vlákno) umožňuje nastavit parametry záchytu (velikost n, aktivace dalších filtrů pro ukládání dat, velikost paměti pro záchyt a indexované položky v hlavičce paketu) a dotazovat se *Query Processing* jednotky, která zprostředkovává vyhledávání v uložených datech.

Vlákno záchytu dat (druhé vlákno) se skládá z několika jednotek:

- Tap – Síťové rozhraní
- Capture – Jednotka pro záchyt dat ze síťového rozhraní, založená na knihovně libpcap. Umožňuje základní filtrování paketů.
- Connection Tracking – Jednotka pro výpočet síťových toků a dalších statistik o provozu. Na základě paketu vrací informace o jeho síťovém toku.
- Classification – Jednotka, která vybírá pouze prvních n paketů z každého toku. Informace o pořadí paketu v toku získává od jednotky *Connection*



Obrázek 3.1: Architektura stroje času od Stefana Kornexla

Tracking. Další funkcí je přidělování paketů do *Storage Container*, na základě specifikované kategorie.

- Storage Container – Jednotky pro ukládání dat. Každá jednotka obsahuje dva kruhové buffery. První kruhový buffer se nachází v operační paměti (velikost v řádu jednotek GB). Do tohoto bufferu jsou data vkládána jednotkou *Classification*. Po zaplnění prvního bufferu jsou nejstarší data opět filtrována a část z nich je přesunuta do druhého kruhového bufferu. Druhý kruhový buffer se nachází na pevném disku (velikost v řádu desítek až stovek GB), po jeho zaplnění jsou nejstarší data vymazána.

  Architektura dvou kruhových bufferů je volena kvůli maximalizování uložené historie. První buffer uchovává nejbližší historii, ale je omezen velikostí paměti. Druhý buffer ukládá starší a méně přesnou historii, ale díky větší kapacitě může uložit více dat.

- Indexing - Jednotka pro indexování obsahu *Storage Container*. Během ukládání paketů zde dochází k vytváření struktury indexů pro efektivní vyhledání požadovaných dat. Indexy mohou být vytvořeny nad libovolnou množinou položek hlavičky paketu.

Sytém je navržen na sítě o propustnosti do 1 Gb/s. Testován byl na síti s maximálním vytížením 320 Mb/s. Pro tuto propustnost systém nestihl zpracovat pouze 0,016 % provozu.

Myšlenka stroje času splňuje požadavky pro ukládání dat detekovaných událostí systémem NEMEA. Nevýhodou je vysoká složitost systému, která omezuje propustnost na 1 Gb/s. Proto je třeba optimalizovat, zjednodušit či

odstranit některé části systému a zároveň využít programovatelných hardwarově akcelerovaných monitorovacích sond pro maximální zvýšení propustnosti.

Kapitola 4

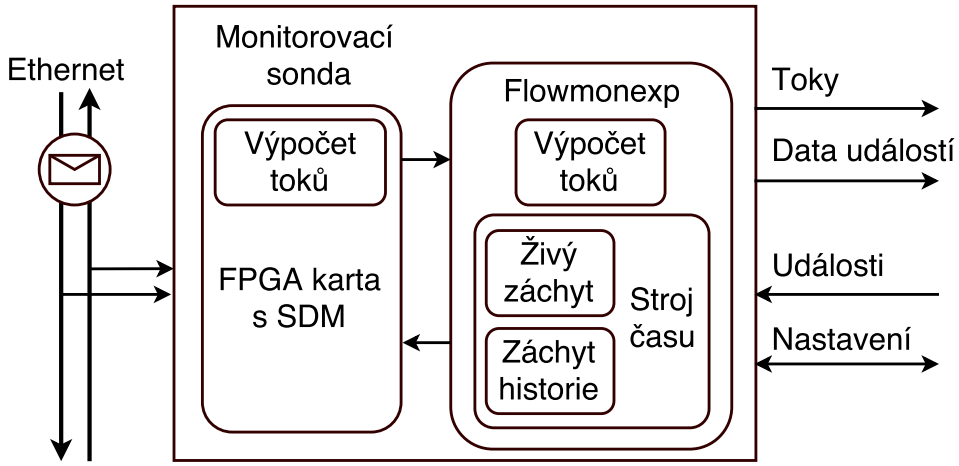
Návrh systému pro automatizovaný záchyt dat událostí

Tato kapitola popisuje návrh systému pro automatizovaný záchyt dat detekovaných událostí. Systém je rozšířením aktuální architektury monitorovací sondy Je inspirován přístupy detekčního systému Bro popsaného v sekci 3.1 a systému Time Machine popsaného v sekci 3.3. Umožňuje tedy jak živý záchyt, tak záchyt historických dat. Navrhovaný systém bude pojmenován stejně jako systém Stefana Kornexla [28], tedy stroj času.

Analýzou sítě CESNET2, používaných prostředků na síti CESNET2 a existujících řešení byly pro stroj času stanoveny následující požadavky:

- Krátkodobé ukládání dat začátků toků (pro zprostředkování historického provozu).
- Zprostředkování přístupu k historickým datům na základě přijaté události.
- Živý záchyt dat na základě přijaté události.
- Konfigurace pomocí uživatelského prostředí.
- Využití programovatelné a hardwarově akcelerované síťové karty pro zvýšení výkonnosti.

Obrázek 4.1 znázorňuje rozšíření architektury programovatelné a hardwarově akcelerované monitorovací sondy (popsané v sekci 2.1.1) o stroj času. Data jsou přijímány pomocí síťové karty, část jich je na základě SDM pravidel odeslána do softwaru. V softwaru jsou data přijaty aplikací Flowmonexp, která je využívá pro výpočet síťových toků a dále je poskytuje zásuvným modulům. Jedním ze zásuvných modulů je i stroj času. Ten se skládá ze dvou



Obrázek 4.1: Rozšíření architektury monitorovací sondy o stroj času

komponent, živého záchytu a záchytu historie. Obě komponenty pracují nezávisle na sobě a jejich návrh je popsán v následujících sekcích. Stroj času umožňuje zpětně pomocí SDM pravidel ovlivňovat data, která se odesílají do softwaru. Pro ovládání stroje času je navržen speciální komunikační protokol, který slouží pro zasílání událostí a nastavení stoje času. Výstupem stroje času jsou data událostí, které mohou být odeslány na jiný stroj nebo uloženy přímo na sondě.

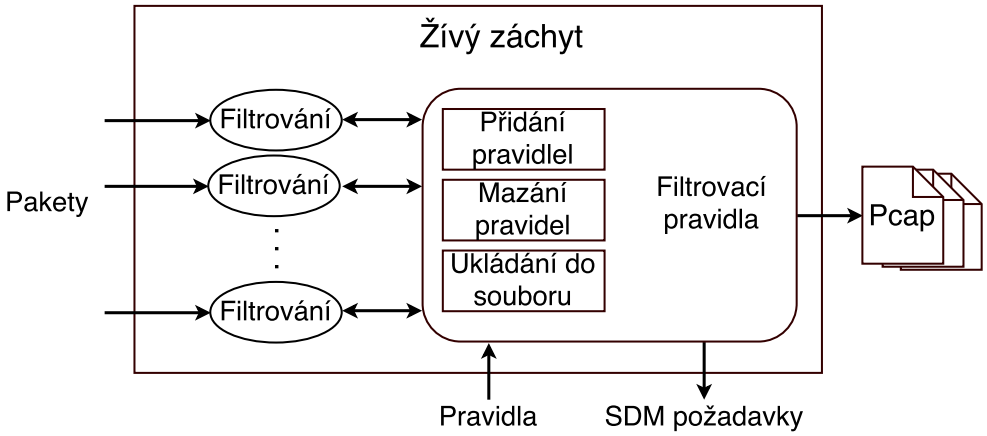
Navržený stroj času, lze použít i se síťovými kartami bez podpory SDM. Data, která by v případě SDM byla zpracována pouze v hardwaru nebudou strojem času použity. Výpočet toků bude probíhat pouze v softwaru, tudíž vytížení procesoru a datové sběrnice bude vyšší než případě s SDM.

4.1 Živý záchyt dat

Pro manipulaci s daty v softwaru je použit program Flowmonexp, který pro maximální využití hardwaru umožňuje přijímat data z DMA kanálů v několika vláknech. Jelikož je stroj času navržen jako zásuvný modul programu Flowmonexp, musí podporovat vícevláknové zpracování dat. Proto je třeba zajistit synchronizaci komunikace mezi jednotlivými instancemi živého záchytu.

Obrázek 4.2 znázorňuje Architekturu komponenty živý záchyt. Pakety jsou filtrovány každým vláknem zvlášť. Po přijetí paketu dojde k dotazu do tabulky s filtrovacími pravidly. Pokud je pravidlo v tabulce nalezeno, paket je uložen do souboru. Tabulka filtrovacích pravidel dále umožňuje přidávání/mazání pravidel a zasílání SDM požadavků.

Komponenta živého záchytu přijímá všechny pakety předané aplikací Flowmonexp. Pro zachycení paketů události je třeba vybrat pouze určitou část pro-



Obrázek 4.2: Architektura živého záchytu

vozu. Softwarové parsování paketů může být časově náročná operace, proto by měla být pro filtrování využita políčka parsovaná FPGA čipem. Mezi tyto políčka patří zdrojová/cílová IP adresa, zdrojový/cílový port, protokol a počet bajtů. Výběr zachyceného provozu bude z důvodu maximální efektivity probíhat pouze na základě IP adres. IP adresa je základním prvkem specifikujícím zdroj nebo cíl události a je obsažena ve výstupech všech detektorů systému NEMEA. Implementace živého záchytu by měla v případě potřeby umožnit jednoduché rozšíření políček pro filtrování.

Pro filtrování provozu jsou potřeba následující informace:

- IP adresa – hlavní klíč pro filtrování paketů.
- Směr – umístění IP adresy v paketu: zdroj, cíl, nespecifikováno.
- Počet paketů – Maximální počet zachycených paketů.
- Časový limit – Maximální doba záchytu.

Pro filtrování je vytvořena tabulka pravidel, která umožňuje záchyt dat k několika událostem současně. Každé pravidlo uchovává aktuální počet zachycených paketů, limit zachycených paketů, čas začátku záchytu, limit času záchytu a soubor se zachycenými pakety. Pravidlo je indexováno IP adresou a směrem. Po přijetí pravidla pro záchyt je přidán nový záznam do tabulky pravidel a je vytvořen soubor pro data záchytu. V případě použití síťové karty s SDM je zaslán SDM požadavek pro odesílání paketů dané IP adresy do softwaru. Po nasbírání potřebného počtu paketů nebo vypršení časového limitu je pravidlo odstraněno a záchyt ukončen. V případě použití síťové karty s SDM je zaslán požadavek pro zpracování zbytku toků dané IP adresy v hardwaru.

Tabulka pravidel je sdílená pro všechna vlákna živého záchytu. Pro umožnění sdíleného přístupu se uzamykají jednotlivá pravidla zvlášť. Díky tomu může libovolný počet vláken současně přistupovat do tabulky, ale do určitého pravidla smí současně přistupovat pouze jedno vlákno. Pravidla tak vždy mají aktuální informace a není třeba další synchronizace vláken. V případě vytvoření tabulky pro každé vlákno zvlášť, by nedocházelo k současnému přístupu více vláken ke stejnému pravidlu, ale musel by se synchronizovat počet zachycených paketů jednotlivými vlákny pro zastavení záchytu. Zároveň by pro každé vlákno vznikl samostatný soubor se záchytem. Tyto soubory by musely být následně sloučeny.

V případě zahlcení provozu jednou IP adresou (DoS/DDoS útok), dochází k serializaci přístupu vláken způsobené blokováním přístupu ke stejnému pravidlu. Serializace vláken vede ke zpomalení zpracování provozu a následnému zahazování paketů. Pro minimalizaci zahazování paketů dochází k automatickému vyrovnávání zátěže. Pro každé vlákno je vytvořena samostatná tabulka pravidel bez omezení souhrnného počtu zachycených paketů. Nedochází tak k žádné synchronizaci vláken. Po ukončení zahlcení provozu se komponenta živého záchytu vrací k původní verzi záchytu. Toto automatické vyrovnávání zátěže není v aktuální verzi práce implementováno, popis slouží pouze pro budoucí rozšíření systému.

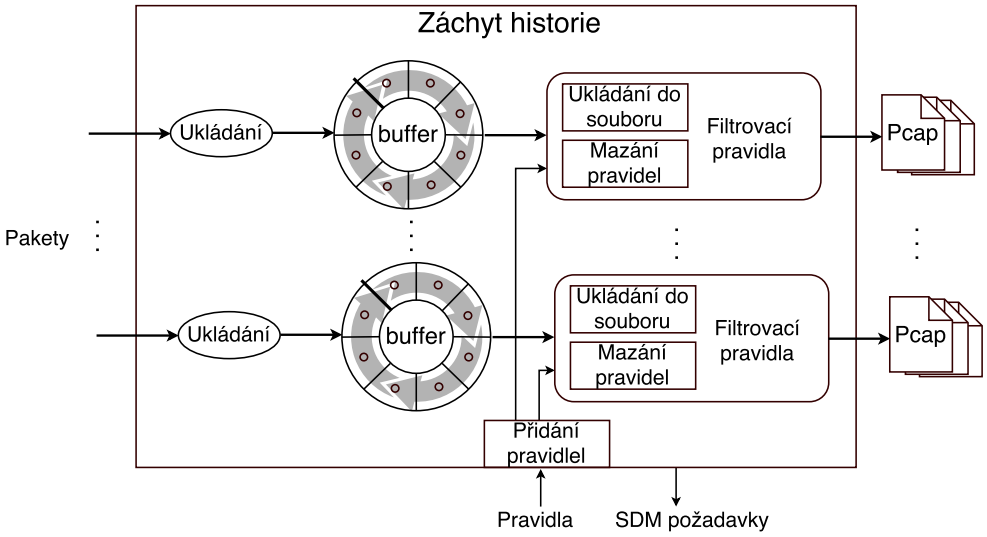
4.2 Záchyt historie

Komponenta záchytu historie zajišťuje výběr prvních n paketů toku, jejich krátkodobé uložení do paměti a následné vyhledání dat určité události. Výsledná architektura je zobrazena na obrázku 4.3. Každé vlákno má vlastní kruhový buffer pro ukládání historie a vlastní tabulku pravidel, která zprostředkovává prohledání bufferu a vytvoření PCAP souborů.

4.2.1 Bufferování dat

Bufferování neboli krátkodobé ukládání dat vyžaduje oproti živému záchytu více systémových prostředků. Jedná se především o dostatečně velkou paměť pro ukládání provozu. Jelikož je systém určen pro provoz vysokorychlostních sítí, je třeba ukládat jen nejdůležitější data. V opačném případě by došlo k téměř okamžitému vyčerpání systémových prostředků a k zahazování paketů, z důvodu přetížení systému.

Podobně jako systém od Stefana Kornexla, popsaný v sekci 3.3, se komponenta historického záchytu omezí na ukládání prvních n paketů každého toku. Hostitelská aplikace zásuvnému modulu zprostředkovává pořadí zpracovávaného paketu v toku. Záchyt historie tak jednoduše filtruje pakety pro uložení. V případě použití síťové karty s SDM, je po uložení prvních n paketů toku, odeslána žádost o zpracování zbytku toku v hardwaru.



Obrázek 4.3: Architektura historického záchytu

Stroj času od Stefana Kornexla využívá pro bufferování dat operační paměť i pevný disk. Jeho řešení je navrženo pro sítě s maximální propustností 1 Gb/s, což je $100x$ nižší propustnost než v případě použití 100 Gb/s karet. Z tohoto důvodu, komponenta historického záchytu využívá jen operační paměť, která umožňuje rychlý přístup. Pevný disk pro přímé ukládání provozu by způsobil mnohonásobné snížení propustnosti.

V operační paměti se vytvoří kruhový buffer, který ukládá pakety začátků toků. Pakety jsou uloženy v chronologickém pořadí, ve kterém jsou přijaty. Po zaplnění bufferu, dochází k automatickému přepisování nejstarších paketů nově přijatými pakety.

Velikost alokované paměti pro buffer přímo ovlivňuje délku uloženého provozu, která lze definovat jako rozdíl času mezi nejstarším a nejnovějším paketem uloženým v bufferu. Dalším faktorem ovlivňující délku uložené komunikace je velikost začátku toku (parametr n). Při fixním nastavení n se délka zachyceného provozu mění v závislosti na aktuálním vytížení sítě. Stroj času proto umožňuje nastavení délky uloženého provozu, podle kterého se dynamicky mění parametr n. Tím je zajištěna minimální délka uloženého provozu. V aktuální verzi práce není dynamická změna parametru n implementována, popis slouží pro možnost budoucího rozšíření systému.

Stejně jako živý záchyt, i záchyt historie je spouštěn v několika vláknech. Proto je třeba rozhodnout, které prostředky budou sdílené a které bude mít každé vlákno vlastní.

V případě přístupu několika vláken ke sdílenému kruhového bufferu se musí jednotlivá vlákna serializovat. Při využití SDM, dochází k velice častému

blokování vláken, protože většina přijatých paketů je zapisována do bufferu. To vede k degradaci vícevláknového přístupu a následného zahazování nově přijatých paketů. Proto je pro každé vlákno vytvořen vlastní kruhový buffer. Buffery všech vláken mají stejnou velikost, přesto může být délka uložených dat u jednotlivých vláken rozdílná v závislosti na velikosti paketů. Architektura více bufferů ovlivňuje i následné vyhledání dat událostí, které je popsáno v následující sekci.

4.2.2 Vyhledání dat událostí

Po uložení paketů do kruhových bufferů, následuje vyhledání specifických dat na základě přijatých událostí. V této sekci jsou popsány dvě možnosti vyhledávání dat v kruhovém bufferu. Jedná se o indexování a lineární průchod. Jsou srovnány výhody i nevýhody obou přístupů a v závislosti na požadavcích stroje času je vybrána optimální metoda. Stejně jako v živém záchytu se filtrování paketů omezí pouze na IP adresu a její umístění (zdrojová, cílová nebo nespecifikováno).

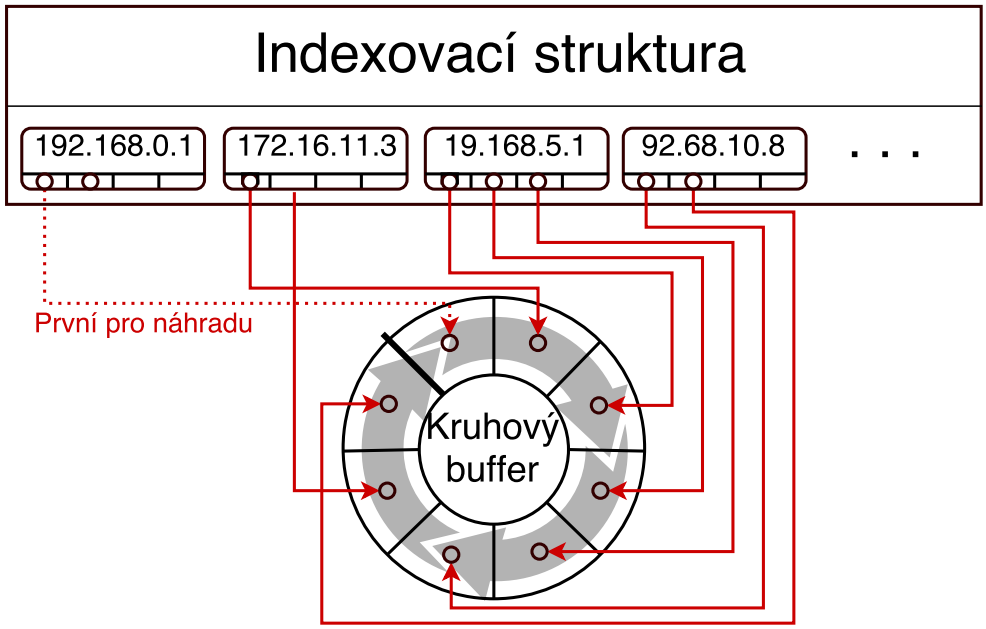
Indexování

Index [30] je speciální datová struktura, která slouží pro ukládání dat, během kterého je vytvářen soubor informací. Ten je poté použit pro efektivní vyhledání požadovaných dat. Index se vytváří na základě klíče, což je položka nebo skupina položek, které jednoznačně specifikují ukládaná data.

Vyhledávací systémy běžně využívají mnoho indexovacích metod. Jednotlivé metody se liší v závislosti na datovém formátu klíče: text, čísla, dokumenty, grafika, hlas atd. Data mohou být indexovány průběžně během vkládání, mazání nebo až v případě kdy je třeba něco vyhledat. Průběžné indexování zaručuje vždy efektivní přístup k datům, zato odložené indexování může narušit efektivitu přístupu. V obou přístupech je nevyhnutelné provést určité operace pro uchování aktualizovaného a použitelného indexu. Další nevýhodou může být paměťová náročnost indexu, která závisí na velikosti klíče a počtu uložených položek.

V případě stroje času je vhodné aktualizaci provádět při každém vkládání a mazání paketu. Nahodilá aktualizace by mohla způsobit blokování kruhového bufferu, které by vedlo k následné ztrátě dat. V případě, že je buffer zaplněn, může vložení nově přijatého velkého paketu způsobit přepsání až x paketů. V takovém případě bude třeba odstranit x záznamů a přidat jeden nový záznam do indexu.

Vyhledávání v bufferu by mělo co nejméně ovlivňovat funkcionalitu ukládání nových dat. V opačném případě by mohlo dojít k zahlcení DMA kanálů a ztrátě dat. Při použití indexování je třeba zamykat přístup k jednotlivým položkám indexu, aby nedošlo k přepsání požadovaných paketů. Vyhledání paketů se skládá z následujících operací:



Obrázek 4.4: Ukázka indexování kruhového bufferu

1. Přijetí pravidla (IP adresa, směr).
2. Vyhledání IP adresy v indexu.
3. Uzamknutí položky.
4. Zkopírování dat z bufferu do výstupního souboru.
5. Odemknutí položky.

Obrázek 4.4 znázorňuje příklad indexování. Index ukládá pouze IP adresy, které jsou momentálně v kruhovém bufferu. Pro zjednodušení jsou na obrázku indexovány pouze zdrojové adresy paketů. Každá IP adresa má svůj záznam, který obsahuje seznam ukazatelů na pakety v bufferu (červené šipky). V tomto příkladu jsou IP adresy 192.168.0.1, 192.16.11.3, 19.168.5.1 a 92.68.10.8. Černá tlustá linka na kruhovém bufferu znázorňuje konec a začátek fronty (místo mezi nejstarším a nejnovějším paketem). Nejstarší paket v bufferu, který má adresu 192.168.0.1 bude nahrazen nově příchozím paketem. Jelikož po přepsání tohoto paketu, adresa neobsahuje žádné další ukazatele do bufferu, bude vymazána z indexu.

Propustnost indexovaného kruhového bufferu je omezena především častou manipulací s indexem. Každý nově příchozí paket způsobuje průměrně 4

operace s indexem (2 vkládání do seznamu zdrojové a cílové adresy nového paketu a 2 operace mazání z listu zdrojové a cílové adresy přepisovaného paketu). Přidávání/mazání adres závisí počtu aktuálně komunikujících adres.

Majoritní operací kruhového bufferu je přidávání a přepisování paketů. Proto není vhodné vytvářet speciální strukturu pro udržování aktuálních informací o bufferu, která je využita minoritní operací vyhledání dat. Pro implementaci vyhledávání dat v kruhovém bufferu je třeba použit jednoduší způsob, který nevytěžuje operaci vkládání nových paketů.

Lineární průchod

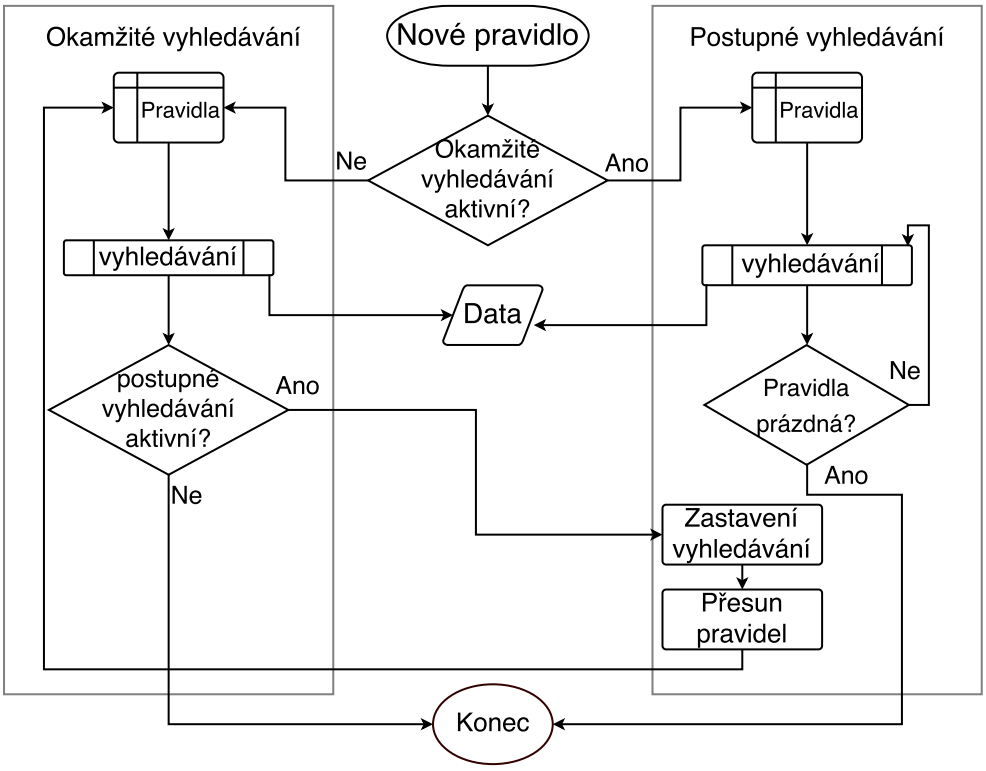
V tomto případě jsou pakety po přijetí a odfiltrování začátků toků ukládány do kruhového bufferu bez vytváření či aktualizování jiné datové struktury. Jedinou možností, jak vyhledávat data v neuspořádaném kruhovém bufferu, je porovnat hledané hodnoty se všemi položkami neboli lineární průchod. Hlavní výhodou tohoto přístupu je nezatěžování dalšími operacemi při ukládání nově příchozích paketů a žádná přidaná paměťová náročnost.

Lineární průchod využívá jednoduchý filtr paketů obsahující přijatá pravidla pro záchyt. Průchod bufferu můžeme provést ihned po přijetí požadavku na vyhledání anebo postupně v době odstraňování nejstarších paketů.

Varianta okamžitého prohledání prochází buffer od nejstaršího uloženého paketu, až po nejnovější paket přidaný před spuštěním prohledání bufferu. Prohledání bufferu může být časově náročné (závisí na velikosti bufferu a počtu nalezených položek), proto musí pracovat paralelně s vkládáním nových paketů. Buffer je sdílen oběma operacemi, tudíž je třeba synchronizace. Vkládání přijatých paketů nesmí přepsat pakety zpracovávané operací vyhledání, v takovémto případě by došlo ke čtení nekonzistentních dat. Výhodou okamžitého prohledání je rychlost vyhledání dat události. Nevýhodami jsou: potřeba synchronizace, složitější implementace, složité přidání dalšího požadavku na vyhledání a výpočetní náročnost (zvláštní vlákno pro vyhledávání).

Varianta postupného vyhledávání je založena na postupném přepisování kruhového bufferu. Pakety jsou filtrovány při přepisování nově příchozími pakety. Celý buffer je prohledán až poté, co dojde k jeho úplnému přepsání. Doba prohledání je přímo úměrná délce uložených dat (rozdíl časové známky nejnovějšího a nejstaršího paketu). Po přijetí nového pravidla pro záchyt, je pravidlo vloženo do filtrovací tabulky. Každé pravidlo obsahuje pozici konce prohledání, aby bylo zřejmé, kdy je možné pravidlo z filtru odstranit. Výhodami postupného vyhledávání jsou: není potřeba synchronizace (žádné další vlákno), nízká výpočetní náročnost a jednoduché přidání dalších pravidel. Nevýhodou je pomalé prohledání bufferu.

Navrhovaný stroj času klade důraz především na vysokou propustnost sítě. Dlouhé čekání na výsledek vyhledání však může negativně ovlivňovat další aplikace spolupracující se strojem času. Z tohoto důvodu je voleno spojení obou vyhledávacích přístupů.



Obrázek 4.5: Vývojový diagram lineárního prohledávání kruhového bufferu

Spojení přístupů postupného a okamžitého prohledání, znázorněného na vývojovém diagramu 4.5, má za účel zrychlit vyhledání požadovaných paketů a zároveň nadměrně nezatěžovat výpočetní prostředky. Hlavní nevýhodou okamžitého vyhledání je synchronizace s vláknem vkládání a složité přidání dalších pravidel. Z tohoto důvodu je použita pouze jedna instance okamžitého prohledání, která je buď ve stavu aktivní (vyhledávání v bufferu) nebo pasivní (čeká na práci). V případě přijetí nového vyhledávacího pravidla, je pravidlo přidáno do filtru postupného vyhledávání (okamžité vyhledávání je aktivní) nebo do filtru okamžitého vyhledávání (okamžité vyhledávání je pasivní). Instance okamžitého vyhledávání po uspokojení všech pravidel (vyprázdnění filtru) zkontroluje stav filtru postupného vyhledávání. Pokud filtr obsahuje pravidla, instance převezme všechna tato pravidla a dokončí jejich vyhledávání, v opačném případě přechází do pasivního stavu.

Lineární průchod na rozdíl od indexování nevytěžuje výpočetní prostředky dalšími operacemi při vkládání nových paketů. Naopak indexování zprostředkovává rychlejší vyhledání, při kterém nemusí procházet celý buffer. Jelikož stroj času cílí na maximální propustnost, je preferována metoda lineárního průchodu.

4.3 Komunikační protokol

Stroj času potřebuje komunikovat s ostatními aplikacemi, které jsou součástí monitorovací infrastruktury a zprostředkovat uživatelské rozhraní pro snadné manuální ovládání a testování. Pro lokální či vzdálenou komunikaci s jakoukoliv aplikací byl navržen jednoduchý protokol používající TCP spojení. Stroj času poskytuje otevřený port pro připojení a obsluhu nezávislého počtu klientů. Na základě přijatých zpráv, dochází k ovládání komponent živého záchytu a záchytu historie.

Komunikační protokol definuje dva typy zpráv, žádosti (zprávy od klienta ke stroji času) a odpovědi (zprávy od stroje ke klientovi). Všechny typy a podporovaných žádostí a odpovědí jsou popsány v příloze B

4.4 Ovládání systému

Stroj času je spuštěn současně s aplikací Flowmonexp jako jeho zásuvný modul. K následné manipulaci stroje času může být použit pouze komunikační protokol. Pro manuální ovládání a testování stroje času slouží klientský program, který je založen na navrženém komunikačním protokolu a všechny jeho funkcionality z něj vychází.

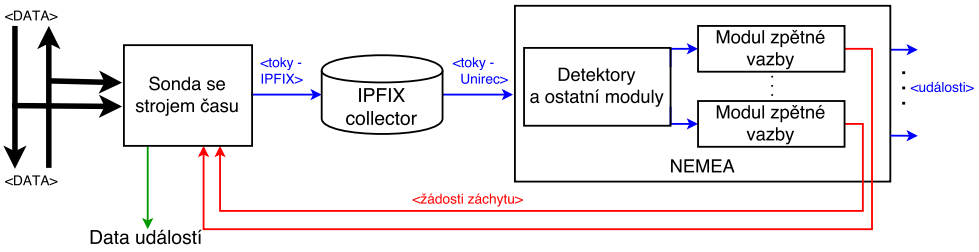
Při spuštění programu je zadána IP adresa monitorovací sondy a port, na kterém naslouchá stroj času. Po úspěšném připojení, je pomocí jednoduchého interpretru možné ovládat stroj času. Všechny příkazy interpretru jsou vypsány v příloze C.

4.5 Začlenění do monitorovací infrastruktury

Navržený stroj času umožňuje na základě zaslané žádosti zprostředkovat historický a živý záchyt. Pro záchyt dat nahlášených událostí musí být zapojen do monitorovací infrastruktury. Upravená architektura infrastruktury s jednou monitorovací sondou je znázorněna na obrázku 4.6. Zpětná vazba ze systému NEMEA do stroje času je zprostředkována přímým spojením pomocí navrženého komunikačního protokolu.

V systému NEMEA je třeba modul zpětné vazby, který konvertuje přijaté události z UniRec formátu do definovaného protokolu. Modul obsahuje jedno vstupní rozhraní, na které se připojí výstup detektoru nebo agregačního modulu. Požadovaným políčkem na vstupu modulu je pouze IP adresa, která je obsažena ve všech nahlášených událostech. Jméno souboru, směr IP adresy a limity záchytu definuje modul zpětné vazby.

Při použití více detektorů existují dvě možnosti propojení systému NEMEA a stroje času. První možností je spojení událostí určených pro záchyt speciálním modulem přímo v systému NEMEA. Události jsou poté odeslány jednou instancí modulu zpětné vazby. Druhou možností je připojení modulu



Obrázek 4.6: Upravená monitorovací architektura s jednou sondou

zpětné vazby ke každému detektoru. Tato možnost umožňuje definovat jméno a limity záchytu pro každý detektor zvlášť. Nevýhodou je více aktivních spojení mezi oběma systémy.

Kapitola 5

Návrh distribuované architektury automatizovaného záchytu dat událostí

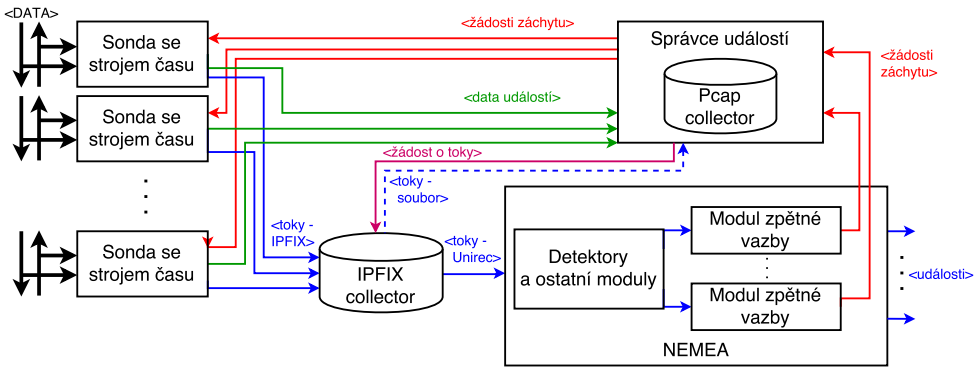
Větší sítě většinou obsahují více monitorovacích sond (viz sekce 2.1). Proto je třeba navrhnout distribuovanou architekturu automatického záchytu nahlášených událostí, která zprostředkuje záchyt na všech sondách současně.

Distribuovaný systém [31] je množina několika počítačů, které se jeví uživateli jako jeden koherentní systém. Práce je přijata hlavním uzlem, který podle vytížení jednotlivých uzlů rozděluje práci. V případě stroje času jsou počítači monitorovací sondy, které se starají o sběr dat z jednotlivých míst na síti. Žádost o záchyt je rozeslána hlavním uzlem na všechny monitorovací sondy. Nevýhodou je nerovnoměrnost vytížení sond, která je určena aktuálním vytížením dané linky. Proto je třeba umožnit změnu nastavení jednotlivých strojů času nezávisle na sobě.

5.1 Architektura

Obrázek 5.1 znázorňuje distribuovanou architekturu monitorovací infrastruktury se strojem času. Architektura je jednoduše rozšiřitelná na libovolné množství monitorovacích sond.

Data jsou přijaty monitorovacími sondami se strojem času. Zde jsou agregována do síťových toků a odeslána na IPFIX collector, který zajišťuje sloučení toků z více měřících bodů, jejich dlouhodobé uložení a přeposlání do systému NEMEA. Ten na základě detektorů analyzuje síťové toky a hlásí podezřelé události. Ty mohou být dále agregovány, filtrovány nebo jinak upraveny. Události určené pro záchyt jsou odeslány do modulu, který zprostředkovává zpětnou vazbu (červené šipky). Na rozdíl od infrastruktury s jednou monitorovací sondou (viz sekce 4.5), jsou žádosti odeslány správci událostí, který zajistí



Obrázek 5.1: Architektura automatizovaného záchytu z více sond

následnou distribuci a zpracování záchytu. Pro modul zpětné vazby systému NEMEA je transparentní, zda se připojuje k měřící sondě nebo ke správci událostí. Původní návrh protokolu zůstává pro zajištění konzistence neměnný.

Správce událostí je hlavním řídícím prvkem záchytu dat událostí. Komunikuje se všemi sondami, systémem NEMEA, IPFIX collectorem a volitelně dalšími systémy. Díky tomu umožňuje shromáždit veškerá data týkající se nahlášené události. Správce událostí zajišťuje tyto funkce:

- Udržení spojení se sondami - Správce udržuje spojení s monitorovacími sondami. V případě výpadku spojení, se snaží jej obnovit.
- Zprostředkování zpětné vazby - Přeposlání požadavků o záchyt všem připojeným sondám a aktivní sledování souhrnného stavu záchytu.
- Filtrování událostí - V případě přijetí velkého množství událostí dochází k jejich filtrování. Vybírají se nejvýznamnější události, jejichž záchyt prioritní.
- Příjem zachycených dat - Po dokončení záchytu jsou všechny zachycené soubory odeslány správci událostí. Ten data přiřadí k nahlášené události, sloučí data sondy do jednoho souboru a volitelně provede paketovou analýzu dat nástroji jako je například Bro (viz sekce 1.3.2) nebo Snort (viz sekce 1.3.1). Výsledky této analýzy uloží k nahlášené události.
- Získání síťových toků - Na základě přijaté události, správce zašle žádost IPFIX collectoru o záznam síťových toků určité IP adresy. Přijaté síťové toky uloží k nahlášené události.
- Spojení s dalšími systémy - Správce umožňuje libovolné rozšíření o spojení s dalšími systémy, které umožní získat rozšiřující data k nahlášené události. Získaná data přidá k záznamům o události.

- Vytvoření záznamu události - Pro každou přijatou událost je vytvořen adresář obsahující data události získaná z různých systémů (pcap soubory ze stroje času, síťové toky z IPFIX collectoru, analýza Bro, analýza Snort atd.). Po získání všech relevantních dat je adresář komprimován a dlouhodobě uložen. Manažer zprostředkovává ostatním aplikacím přístup k těmto záznamům.

Komunikace mezi správcem událostí a stroji času probíhá pomocí navrženého protokolu přes SSH tunel. Zachycená data jsou ze sond ke správci událostí odeslána pomocí protokolu SCP.

5.2 Ovládání

Připojovat se, nastavovat a testovat každou instanci stroje času zvlášť je zdlouhavé a neefektivní. Z tohoto důvodu je potřeba aplikace, která se připojí k více strojům času a obsluhuje je zároveň.

Základem je již navržená aplikace pro ovládání jedné instance stroje času popsaná v sekci 4.4. Aktualizovaný návrh zachovává všechny původní funkcionality. Jediným rozdílem je počet strojů, které aplikace ovládá. Jelikož rozšířená aplikace umožňuje ovládat i jednu instanci stroje času, plně nahrazuje původní aplikaci.

Při spuštění aplikace se z konfiguračního souboru načtou IP adresy a porty, ke kterým se aplikace připojí. V případě neúspěchu připojení k určité sondě, bude sonda z běžící konfigurace vyřazena. Pokud program zaznamená náhlý výpadek spojení se sondou, pokusí se automaticky spojení obnovit.

Pro ovládání strojů času jsou určeny stejné příkazy jako v původním návrhu aplikace. Zadané příkazy jsou aplikovány na všechny aktivní sondy. Pro specifikaci ovládaných sond přibyl příkaz *probes*. Pomocí tohoto příkazu lze definovat sondy, na které budou příkazy aplikovány (aktivní sondy). Zavoláním samotného příkazu *probes*, dojde k vypsání informací o připojených sondách. Informace o každé sondě obsahují: ID sondy, IP adresa, port, připojeno/odpojeno, aktivní/pasivní. Zavoláním příkazu *probes [ID sondy] [ID sondy] ...*, dojde k aktivaci specifikovaných sond. Ostatní sondy jsou nastaveny jako pasivní. Pro úspěšnou aktivaci sondy, musí být sonda připojená. Zavoláním příkazu *probes all* dojde k aktivaci všech připojených sond.

Kapitola 6

Implementace rozšíření monitorovací sondy

Automatický záchyt dat událostí byl navržen jako zásuvný modul aplikace Flowmonexp. Flowmonexp i jeho zásuvné moduly jsou implementovány v jazyce C. Proto i zásuvný modul pro automatický záchyt dat událostí (stroj času) je implementován v jazyce C. Implementace je rozdělena na několik funkčních částí, nazývaných komponenty.

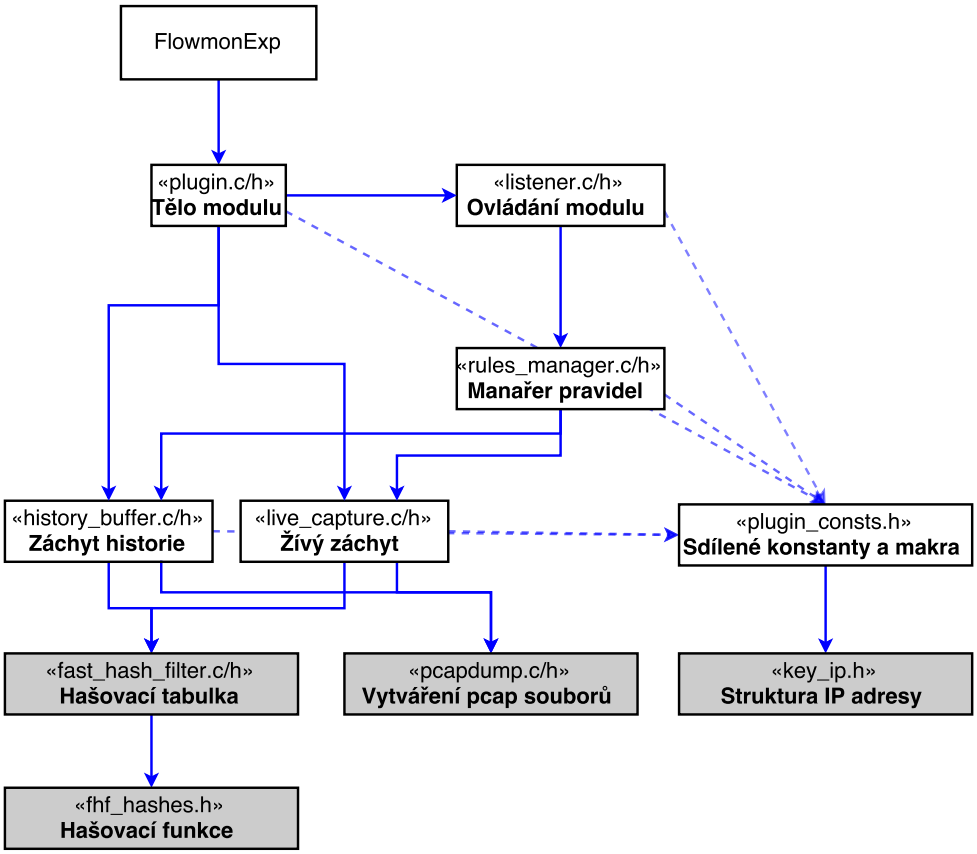
Komponenty jsou reprezentovány hlavičkovými soubory obsahující deklarace funkcí a struktur. Zdrojové soubory poté definují jednotlivé funkce. Toto rozdělení umožňuje nezávisle upravovat pouze danou komponentu bez zásahu do ostatních souborů.

Obrázek 6.1 znázorňuje rozdělení zásuvného modulu na jednotlivé hlavičkové soubory (komponenty). Modré šipky ukazují směr závislostí mezi jednotlivými soubory. Čárkované čáry vedoucí k souboru *plugin_consts.h* znázorňují použití sdílených struktur, maker a konstant v rámci celého modulu. Soubory s šedým podbarvením jsou převzaté ze zásuvného modulu *Sdmcap* vytvořeného sdružením CESNET. Ten byl dosud používán pro záchyt dat určité IP adresy. Jelikož tuto funkcionalitu vykonává i stroj času pomocí komponenty živého záchytu, plně tak nahrazuje existující modul *Sdmcap*.

Následující seznam popisuje převzaté soubory.

- Hašovací tabulka - Obsahuje metody pro vytvoření hašovací tabulky, vložení prvku, odstranění prvku a vyhledání hodnoty na základě klíče. Hašovací tabulka se skládá z libovolného počtu řádků a osmi sloupců. Každý řádek tabulky obsahuje zámek typu *Test and Set Lock* [32], který zajišťuje atomický přístup k položkám stejného řádku. Tato struktura je tedy vhodná pro paralelní přístup.

- Hašovací funkce - Definuje tři hašovací funkce pro hašovací tabulku. Jednotlivé funkce se liší v délce hašovaného klíče.



Obrázek 6.1: Diagram hlavičkových souborů

- Vytváření pcap souborů - Funkce pro vytvoření souboru s pcap hlavičkou, následné vkládání paketů a uzavření souboru.
- Struktura IP adresy - Definice struktury pro uložení IP adresy, funkce pro konverzi IP adresy na textový řetězec a kontrola verze IP adresy. Struktura IP adresy je při filtrování použita jako klíč hašovací tabulky.

Implementace jednotlivých komponent modulu je popsaná v následujících sekcích.

6.1 Tělo modulu

Tělo modulu definuje funkce volané hostitelskou aplikací FlowmonExp. Jedná se o funkce pro zjištění informací o modulu (*PLUGIN_PROCESS_DESC*), inicializaci modulu (*PLUGIN_PROCESS_INIT*), paralelní zpracování příchozích paketů (*PLUGIN_PROCESS_UPDATE*) a ukončení modulu (*PLU-*

GIN_PROCESS_SHUTDOWN). Implementace těla modulu je v souborech *plugin.c/h*.

6.1.1 Popis modulu

PLUGIN_PROCESS_DESC je funkce, jejímž cílem je vrátit strukturu definující jméno modulu, popis modulu, vstupní parametry modulu a požadavky na data předávané přijímacím vláknům.

Vstupní parametry pro modul jsou:

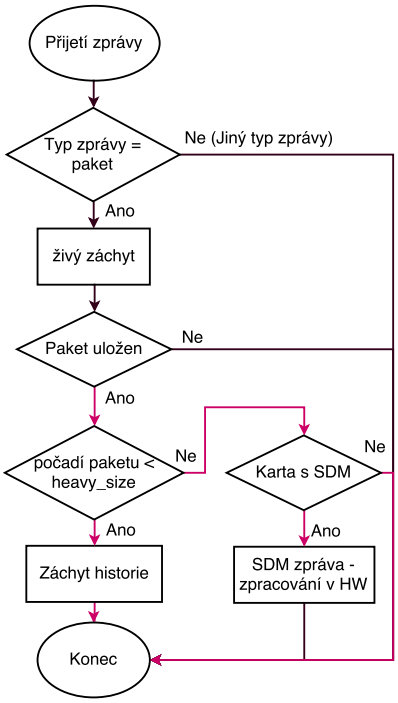
- port=<číslo portu> - Definuje port na kterém bude naslouchat komunikační rozhraní modulu.
- sdm_context=<ID kontextu> - Definuje ID SDM kontextu. Pokud je ID rovno -1, je vytvořen nový SDM kontext. Tento parametr může být definován pouze při použití SDM.
- verbose=<0-5> - Definuje obsáhlost výpisu stavových informací. 0 = žádné informace, 5 = všechny dostupné informace.
- capture_path=<cesta> - Definuje cestu, kde budou ukládány pcap soubory se zachycenými daty. Pokud není cesta zadána, soubory jsou ukládány do aktuálního adresáře.
- capture_script=<jméno skriptu> - Definuje volitelný skript, který je spuštěn po dokončení záchytu. Skriptu jsou parametrem předána jména zachycených souborů. Pomocí skriptu lze například odeslat zachycená data nebo spustit analýzu dat.

6.1.2 Inicializace modulu

PLUGIN_PROCESS_INIT je funkce pro inicializaci modulu. Je volána v jednom vlákně, během inicializace hostitelské aplikace. V této funkci probíhá načtení vstupních parametrů, připojení k SDM (pokud je SDM použito) a inicializace komponent, se kterými tělo modulu přímo komunikuje. Jedná se o *Živý záchyt*, *Záchyt historie* a *Komunikační rozhraní*. Instance živého záchytu je společná pro všechna vlákna přijímající pakety (přijímající vlákna). Naopak záchyt historie je vytvořen pro každé přijímající vlákno zvlášť.

6.1.3 Přijetí dat

PLUGIN_PROCESS_UPDATE je funkce volaná po přijetí zprávy (paketu, hlavičky nebo síťového toku) některým z přijímacích vláken. Diagram 6.2 znázorňuje akce vykonané touto funkcí. Nejprve se kontroluje, zda se jedná o paket či jiný typ zprávy. Pokud o jiný typ zprávy, funkce končí. V opačném případě dochází k předání komponentě živého záchytu. Pokud je paket živým záchytem



Obrázek 6.2: Diagram aktivit - Přijetí paketu

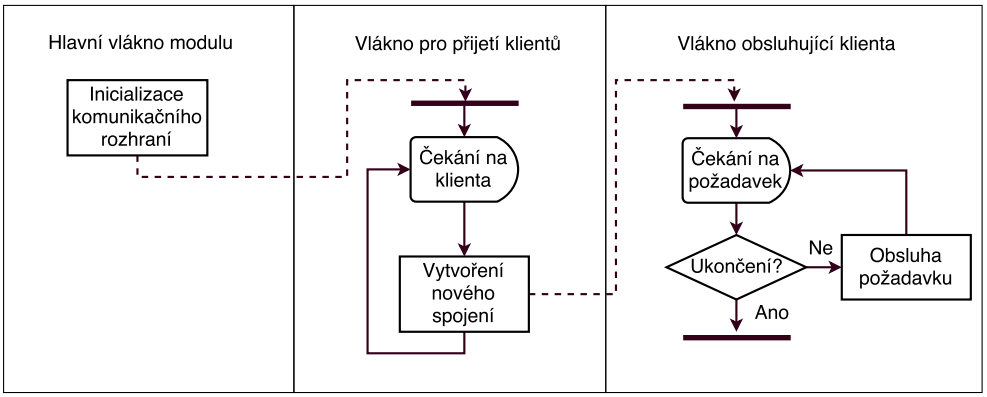
uložen, funkce končí. V opačném případě dojde ke zkontrolování pořadí paketu v síťovém toku. Pokud je pořadí menší než hranice definovaná proměnnou *heavy_size*, paket je předán komponentě záchytu historie. V opačném případě je odeslán SDM požadavek, aby všechny pakety tohoto toku byly zpracovány v hardware. Pokud je použita karta bez podpory SDM, nedochází k odeslání SDM požadavku a funkce okamžitě končí.

6.1.3.1 Ukončení modulu

PLUGIN_PROCESS_SHUTDOWN je funkce volaná při ukončení Flowmonexp. Cílem funkce je ukončit práci všech komponent modulu a uvolnit alokovanou paměť. Dochází postupně k ukončení SDM spojení, živého záchytu, záchytu historie a komunikačního rozhraní.

6.2 Komunikační rozhraní

Hlavním ovládacím prvkem zásuvného modulu je TCP komunikace definovaná komunikačním protokolem popsaným v příloze B. Komponenta vytváří server pro příjem spojení s novými klienty a jejich následnou obsluhu. Obrázek 6.3 znázorňuje využití vláken pro přijímání a obsluhu klientských požadavků. Implementace komunikačního rozhraní je v souborech *listener.c/h*.



Obrázek 6.3: Server pro komunikaci s klienty

6.2.1 Inicializace

Během inicializace komunikačního rozhraní jsou od těla modulu přijaty potřebné komponenty pro ovládání stroje času. Jedná se o otevřené SDM spojení (pokud je použita karta s SDM), instanci živého záchytu a pole ukazatelů na instance záchytu historie. Nad těmito komponentami je inicializován manažer pravidel, který se v závislosti na přijatých příkazech stará o jejich ovládání. Nakonec je vytvořeno vlákno pro příjem a obsluhu nových spojení.

6.2.2 Komunikace s klientem

Vlákno vytvořené během inicializace komunikačního rozhraní slouží pro přijetí nových klientů. Jakmile se klient připojí, je pro něj vytvořeno komunikační vlákno naslouchající požadavkům klienta. Každému obsluhujícímu vláknu jsou předány ukazatele na spojení s SDM a manažera pravidel. Obsluhující vlákno přijímá žádosti od klienta, vykonává je pomocí komponenty manažera pravidel a zpět posílá odpovědi definované komunikačním protokolem. Po odpojení klienta je vlákno ukončeno.

6.2.3 Ukončení

Při ukončení komponenty dochází k odpojení všech aktivní spojení s klienty, ukončení jejich komunikačních vláken a vlákna přijímající nová spojení. Nakonec je ukončen manažer pravidel, který zodpovídá za okamžité dokončení rozpracovaných záchytů.

6.3 Manažer pravidel

Jedná se o komponentu řídící životní cyklus pravidel pro záchyt (viz obrázek 6.4). Komponenty živý záchyt a záchyt historie jsou implementačně odděleny, tak aby pracovali nezávisle na sobě. Manažer pravidel zprostředkovává jednotný přístup k záchytu ovládáním obou těchto komponent. Manažer pravidel je implementován v souborech *rules_manager.c/h*

6.3.1 Inicializace

Inicializace přijímá ukazatel na instanci živého záchytu a pole ukazatelů na instance záchytu historie. Funkce vytváří zámek pro sdílený přístup k pravidlům, vlákno pro kontrolu a aktualizaci stavu pravidel, seznam aktuálně zpracovávaných pravidel a seznam klientů registrovaných pro odebírání aktualizací záchytu. Je použit zámek typu RWL (Read Write Lock) [33], který umožňuje současný přístup několika vláken ke čtení pravidel, ale výlučný přístup při jejich editaci.

6.3.2 Přidání pravidla

Pro přidání nového pravidla záchytu jsou vyžadovány argumenty: IP adresa, směr záchytu, maximální počet paketů pro záchyt, maximální dobu záchytu a pojmenování záchytu. Přidání pravidla se skládá z následujících kroků:

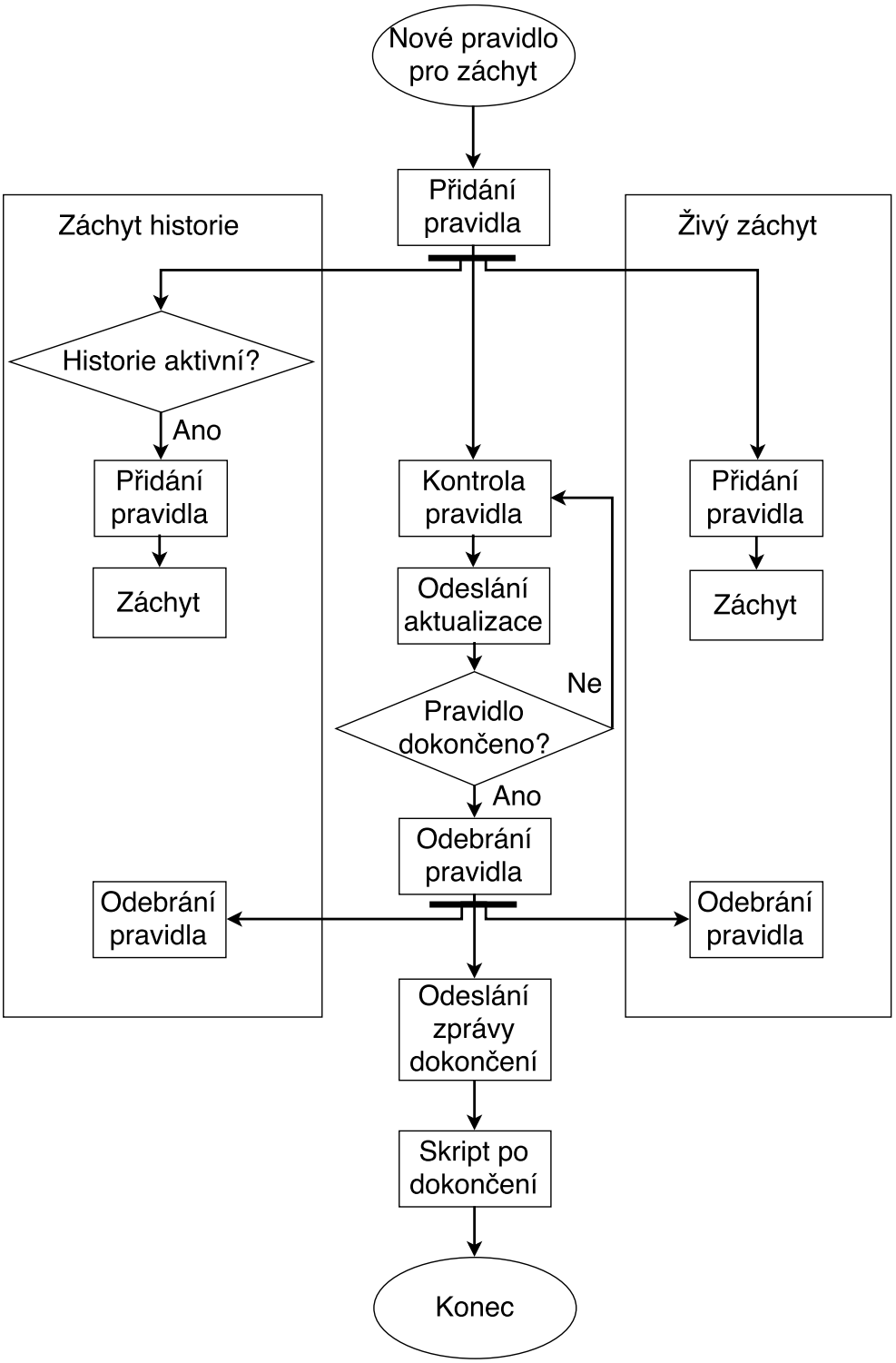
1. Uzamknutí přístupu k datům manažera pravidel pro zápis (přidáním pravidla dochází k editaci seznamu s pravidly).

2. Kontrola existence pravidla záchytu. Pokud jsou IP adresa, směr i pojmenování záchytu totožné s již existujícím pravidlem, dochází pouze k editaci stávajícího pravidla. Maximální počet zachycených paketů se nastaví na součet aktuálního počtu zachycených paketů s nově přijatou hodnotou maximálního počtu zachycených paketů. Maximální doba záchytu se nastaví na součet aktuální doby záchytu s nově přijatou hodnotou maximální doby záchytu. Změny ovlivňují pouze živý záchyt. V případě, že se směr či pojmenování záchytu liší, funkce končí s chybou.

3. Kontrola existence souborů se záchytem. Pokud již existuje záchyt se stejným pojmenováním, název se rozšíří o *_i*, kde i je číslo verze. Verze se zvyšuje od 1 až do hodnoty, při které je jméno unikátní.

4. Přidání pravidla do komponenty živý záchyt.

5. Pokud je záchyt historie aktivní, přidání pravidla do všech instancí komponenty záchyt historie. Pro odlišení jmen souborů jednotlivých instancí, je jméno rozšířeno o *_hb(j)*, kde *j* je číslo instance.



Obrázek 6.4: Životní cyklus pravidel pro záchyt

6. Jména souborů, IP adresa a její směr jsou uloženy spolu s pravidlem které je následně uloženo na začátek spojového seznamu s pravidly.

7. Uvolnění zámku pravidel.

6.3.3 Odebrání pravidla

Pro odebrání pravidla jsou vyžadovány argumenty IP adresa a směr záchytu. Jelikož se jedná o editaci seznamu pravidel, sdílený zámek je uzamčen pro zápis. V seznamu pravidel je vyhledáno pravidlo s odpovídající IP adresou a směrem, pokud takové neexistuje, funkce končí s chybou. Voláním příslušných funkcí komponent živého záchytu, respektive záchytu historie dojde k okamžitému ukončení záchytu. Pravidlo je v seznamu označeno za ukončené a sdílený zámek je uvolněn.

6.3.4 Seznam pravidel

Manažer pravidel zprostředkovává funkci předávající aktuální seznam zachytávaných IP adres v daném směru. Nejprve dochází k uzamčení sdíleného zámku pro čtení pravidel. Poté je alokován seznam adres, do kterého se uloží všechny aktuálně zachytávané IP adresy daného směru. Nakonec dochází k uvolnění sdíleného zámku.

6.3.5 Informace o záchytu

Funkce pro získání aktuálních informací o záchytu určité IP adresy (směr záchytu, limity záchytu a aktuální stav záchytu). Nejprve dochází k uzamčení sdíleného zámku pro čtení pravidel. Pokud adresa není nalezena v seznamu pravidel, funkce končí s chybou. V opačném případě jsou z příslušných komponent zjištěny detaily živého záchytu, respektive záchytu historie. Informace jsou uloženy do speciální struktury pro tento typ operace. Nakonec dochází k uvolnění sdíleného zámku.

6.3.6 Pravidelné aktualizace záchytu pravidel

Manažer pravidel pravidelně kontroluje stav záchytu jednotlivých pravidel a umožňuje tyto informace zprostředkovat připojeným klientům. Informace jsou odesílány pouze klientům, kteří pomocí komunikačního rozhraní odešlou příslušnou žádost. Komunikační sokety těchto klientů jsou uloženy v seznamu u manažera pravidel a pro přístup k nim je třeba uzamknout sdílený zámek. Během kontroly, přidání a mazání pravidel jsou všem registrovaným klientům (soketům ze seznamu) zaslány aktualizované informace o záchytu.

Manažer pravidel zprostředkovává funkce pro registraci klienta k odběru informací, a naopak k odhlášení odběru informací. V obou případech dochází

k uzamčení sdíleného zámku pro zápis a k přidání či odebrání soketu ze seznamu. Pokud je v případě registrace klient již v seznamu, funkce končí s chybou. Podobně v případě odhlášení, pokud klient není v seznamu, funkce končí s chybou. Nakonec je uvolněn sdílený zámek.

6.3.7 Obsluha pravidel

Pro kontrolu stavu jednotlivých pravidel, dokončení pravidel a odesílání aktualizací je během inicializace vytvořeno samostatné vlákno. To v pravidelných intervalech (každých 5 s) iteruje přes všechna pravidla a provádí operace popsané v následujícím seznamu. Během iterace je uzamčen sdílený zámek manažera pravidel pro zápis.

1. Zjištění stavu živého záchytu. Pokud je živý záchyt dokončen, pravidlo je z komponenty automaticky odebráno.

2. Pokud bylo pravidlo vloženo do instancí záchytu historie (v době přijetí pravidla byl záchyt historie aktivní), dochází ke zjištění jeho stavu ze všech instancí. V případě, že byl záchyt ukončen, je pravidlo z dané instance automaticky odebráno.

3. Celkový stav záchytu pravidla je určen jako minimum ze stavu živého záchytu a stavů záchytu historie.

4. Pokud se stav od poslední kontroly změnil, je registrovaným klientům odeslána informace o aktualizaci stavu.

5. Pokud je celkový stav pravidla roven 100 %, jedná se o ukončený záchyt, který je již odebrán z komponent záchytu. Toto pravidlo je odebráno ze seznamu pravidel a je vloženo do seznamu pravidel určených pro ukončení.

Po dokončení iterace všech pravidel je uvolněním sdíleného zámku manažera dokončena kontrola a aktualizace stavu pravidel.

Posledním krokem je zpracování seznamu pravidel určených pro ukončení. Ukončení pravidel je z důvodu minimalizace času stráveného v kritické sekci vykonáváno samostatně. Pro všechna tato pravidla se provádí následující operace:

1. Pokud je definován skript pro zpracování záchytu, je tento skript spuštěn. Parametry skriptu jsou jména souborů se záchytem.

2. Odeslání informace o ukončení záchytu registrovaným klientům.

3. Odstranění pravidla.

6.3.8 Ukončení

V této funkci dochází k ukončení a odstranění záchytu všech pravidel, vyprázdnění seznamu registrovaných klientů, ukončení běhu vlákna pro obsluhu pravidel a uvolnění alokované paměti manažerem pravidel.

6.4 Živý záchyt

Komponenta živý záchyt slouží pro filtrování a následné ukládání paketů na základě zdrojové či cílové IP adresy. Instance živého záchytu je sdílená pro všechna vlákna přijímající pakety. Pravidla záchytu jsou uložena v hašovací tabulce, klíčem je IP adresa. Při filtrování paketu dojde k hašování zdrojové a cílové IP adresy a kontrole, zda v tabulce existuje pravidlo s jednou z těchto adres.

Z důvodu vysoké míry koherence při přístupu k instanci živého záchytu nedochází k uzamčení přístupu k celé instanci, ale pouze k určitým prvkům hašovací tabulky. Implementace současného přístupu k prvkům je v režii hašovací tabulky.

Živý záchyt je implementován v souborech *live_capture.c/h*

6.4.1 Inicializace

Funkce vytváří prázdnou hašovací tabulku pro pravidla záchytu.

6.4.2 Přidání a editace pravidla

Pro přidání či editaci pravidla jsou vyžadovány argumenty IP adresa, směr záchytu, maximální počet zachycených paketů, maximální doba záchytu a jméno souboru s výslednými daty.

V případě přidání nového pravidla je nejprve vytvořen soubor s pcap hlavičkou, která jednoznačně definuje typ souboru. Soubor je spolu s limity záchytu uložen ve struktuře pravidla, které je vloženo do hašovací tabulky.

V případě editace pravidla může být upraven směr záchytu, nová hranice počtu paketů a nový časový limit. Pokud IP adresa není v pravidlech, funkce končí s chybou.

6.4.3 Odebrání pravidla

Záchyt lze kdykoliv ukončit voláním této funkce. Vyžadovaným argumentem je IP adresa a její směr. Pravidlo je vyhledáno v hašovací tabulce, zápis souboru je ukončen, a nakonec je pravidlo je odebráno z hašovací tabulky. V případě, že není pravidlo nalezeno, funkce končí chybou.

6.4.4 Filtrování

Přijímající vlákna modulu volají tuto funkci, která filtruje pakety a ukládá je do definovaných souborů. Argumenty funkce jsou zdrojová IP adresa, cílová IP adresa, data paketu, délka paketu a čas přijetí paketu.

Filtrování je provedeno zahašováním IP adres a kontrolou existence odpovídajícího pravidla v hašovací tabulce. Pokud je pravidlo nalezeno, dochází k jeho uzamčení sdíleným zámkem. Dále jsou na základě obsahu pravidla testovány další podmínky záchytu:

- směr pravidla = pozice IP adresy v paketu (zdroj, cíl)
- aktuální počet paketů $\leq$ maximální počet paketů
- aktuální čas $\leq$ čas spuštění záchytu + maximální doba záchytu

Pokud jsou všechny požadavky splněny, paket je zapsán do souboru a je zvýšen počet uložených paketů. Po dokončení filtrování dochází k uvolnění sdíleného zámku pravidla a jeho zpřístupnění ostatním vláknům.

6.4.5 Informace o záchytu

Tato funkce poskytuje podrobné informace o živém záchytu určité IP adresy. Na základě argumentů IP adresy a jejího směru je nalezeno konkrétní pravidlo, které je po dobu čtení informací uzamčeno. Z pravidla jsou získány limity záchytu a aktuální počet zachycených paketů, které jsou předány ve struktuře návratovou hodnotou.

Komponenta živého záchytu dále implementuje funkci pro získání procentuálního stavu záchytu. Ta na základě IP adresy získá limity pravidla a aktuální počet zachycených paketů. Z těchto informací spočítá stav dokončení záchytu v procentech. V případě, že je záchyt dokončen, uzavře zápis souboru a odstraní pravidlo z hašovací tabulky.

6.4.6 Ukončení

Tato funkce postupně ukončí záchyt všech pravidel a vyčistí hašovací tabulku s pravidly.

6.5 Záchyt historie

Komponenta záchytu historie zajišťuje krátkodobé ukládání paketů, jejich filtrování a uložení do souboru. Pro každé přijímací vlákno je vytvořena instance záchytu historie. Jednotlivá vlákna se tak při ukládání paketů nemusí střídat při zápisu do bufferu.

Hlavním prvkem komponenty je kruhový buffer. Ten se skládá z paměti fixní délky a dvou ukazatelů, první určující začátek bufferu (nejstarší paket) a

druhý určující konec bufferu (místo pro vložení nově přijatého paketu). Protože je buffer alokován při povolení záchytu historie, vkládání nového paketu pouze kopíruje data na předem připravené místo.

Pro filtrování dat bufferu slouží samostatné vlákno a dvě hašovací tabulky obsahující filtrovací pravidla. První hašovací tabulka je určena pro filtrování dat během přepisování nejstarších paketů. Druhá hašovací tabulka je využita vyhledávajícím vláknem, které nezávisle na vkládání nových paketů prochází buffer.

Záchyt historie používá několik zámků pro zajištění koherence dat při sdíleném přístupu. Následující seznam popisuje použití jednotlivých zámků a jejich typ.

- Zámek hlavní struktury záchytu historie

  Zámek typu RWL (Read Write Lock), který umožňuje hromadný přístup pro čtení a výlučný přístup pro zápis.

  Ve funkcích záchytu historie, ve kterých nedochází k alokaci, uvolnění či přesouvání ukazatelů sdílené paměti je zámek uzamčen pro čtení. Naopak při alokaci, uvolnění nebo přesunu ukazatelů sdílené paměti záchytu historie je zámek uzamčen pro zápis. Jedná se především o zapnutí historie (alokace bufferu), vypnutí historie (uvolnění bufferu), resetování vyhledávajícího vlákna (přesun hašovacích tabulek).

- Zámek posuvu bufferu

  Zámek typu TSL (Test and Set Lock), který aktivně čeká na přístup do kritické sekce.

  Vkládání nových paketů a mazání nejstarších paketů mění hodnotu ukazatelů začátku a konce bufferu. Tyto hodnoty jsou použity při vkládání nových pravidel záchytu pro specifikaci začátku a konce vyhledávání. Při jejich čtení a zápisu dochází k uzamykání zámku.

- Podmíněná proměnná průchodu bufferu

  Zámek typu mutex doplněný o podmíněnou proměnou [33], která uzamyká mutex dokud není splněna určitá podmínka.

  Při filtrování bufferu je využito samostatné (vyhledávající) vlákno, které prochází a filtruje pakety bufferu. Pro zajištění konzistence filtrovaných paketů, musí být pozice filtrování vždy za ukazatelem začátku bufferu. Mutex tedy uzamyká mazání nejstaršího paketu (změna začátku fronty), pokud je paket momentálně filtrován. Jakmile dojde k posunu filtrování na další paket, je mutex uvolněn a nejstarší paket nahrazen.

- Semafor vyhledávajícího vlákna

  Zámek typu semafor [33] je použit pro pasivní čekání určitého vlákna, dokud jiné vlákno nezvýší hodnotu semaforu.

Pokud v záchytu historie není aktivní žádné pravidlo pro záchyt, vyhledávající vlákno pasivně čeká na přidání pravidla. Jakmile je přidáno první pravidlo, zvýší se hodnota semaforu, vyhledávající vlákno převezme pravidla a začne procházet a filtrovat pakety bufferu. Po dokončení průchodu bufferu vyhledávajícím vláknem, dochází ke snížení hodnoty semaforu.

6.5.1 Inicializace

Pro inicializaci komponenty je vyžadován argument počáteční velikost bufferu v bajtech. Funkce vytváří dvě hašovací tabulky určené pro filtrování bufferu a výše popsané zámky bufferu. Pokud je počáteční velikost bufferu (přijatý argument) větší než nula, dochází k alokování kruhového bufferu a zapnutí bufferování.

6.5.2 Zapnutí a vypnutí bufferování

Komponenta obsahuje funkce pro explicitní zapnutí či vypnutí bufferování. Pro zapnutí je vyžadována velikost bufferu v bajtech. Postupně dochází k uzamčení zámku záchytu historie pro zápis, alokaci bufferu o specifikované velikosti, vytvoření vyhledávajícího vlákna, povolení bufferování a nakonec k uvolnění zámku.

Vypnutí bufferování nevyžaduje žádné argumenty. Prvním krokem je zamknutí záchytu historie pro zápis. Poté je ukončeno vlákno pro vyhledávání, během kterého dochází k ukončení záchytu všech pravidel v hašovací tabulce vlákna. Dále je ukončen záchyt všech pravidel druhé hašovací tabulky. Nakonec dochází k dealokaci bufferu, zakázání bufferovaní dat a uvolnění zámku.

6.5.3 Bufferování paketů

Funkce pro vkládání nových paketů do bufferu. Její argumenty jsou zdrojová IP adresa, cílová IP adresa, ukazatel na data paketu, délka paketu a čas přijetí paketu. Pro každý paket je vytvořena hlavička obsahující metadata, která je uložena do bufferu před samotný paket. Obsahuje zdrojovou i cílovou IP adresu, čas přijetí a délku paketu. Uložené IP adresy slouží pro urychlení filtrování, během kterého nemusí být paket čten a parsován. Délka paketu umožňuje jednoduchý posun na další paket v bufferu. Čas záchytu je použit při ukládání paketu do souboru.

Při samotném vložení do bufferu dojde k uzamknutí zámku posuvu bufferu. Paket se vkládá na konec kruhového bufferu (ukazatel konce bufferu). Hodnota ukazatele začátku i konce bufferu je vždy modulárním zbytkem velikosti bufferu. Díky tomu je zajištěno, že jsou pakety ukládány v kruhu (po překročení limitu hodnoty ukazatele, se začne zapisovat od nuly). V případě nedostatku místa v bufferu dojde k odstranění potřeného množství nejstarších

paketů a tím k posunutí začátku kruhového bufferu. Během odstranění paketu dochází k jeho filtrování na základě pravidel, které nebyly dosud převzaty vyhledávajícím vláknem. Odstranění paketu může být blokováno vyhledávajícím vláknem, pokud momentálně zpracovává paket určený pro odstranění.

6.5.4 Přidání pravidla

Pro přidání nového pravidla jsou vyžadovány argumenty IP adresa, směr záchytu a jméno výstupního souboru. Prvním krokem je uzamčení hlavní struktury záchytu historie pro čtení. Dále se vytváří soubor pro záchyt, do kterého se zapisuje pcap hlavička. Vytvoří se pravidlo pro filtrování, do kterého se uloží ukazatel na souborový proud, IP adresa, směr záchytu, aktuální začátek a konec bufferu. Pravidlo je následně vloženo do hašovací tabulky pro filtrování během přepisování nejstarších paketů, tím je zahájeno filtrování. Pokud se jedná o první pravidlo v této tabulce, je zvýšena hodnota semaforu vyhledávacího vlákna. Nakonec je uvolněn zámek komponenty.

6.5.5 Odebrání pravidla

Pro odebrání pravidla jsou vyžadovány argumenty argumenty IP adresa a směr záchytu. Prvním krokem je zamknutí hlavní struktury záchytu historie pro čtení. Následuje vyhledání položky v hašovací tabulce pro filtrování během přepisování nejstarších paketů. Pokud je pravidlo nalezeno, je vymazáno z tabulky, soubor se záchytem ukončen a zámek uvolněn.

Z důvodu zajištění rychlého průchodu bufferu vyhledávajícím vláknem, nejsou odebírány pravidla v jeho hašovací tabulce.

6.5.6 Filtrování

Pro optimální filtrování dat je použita kombinace dvou přístupů popsaných v sekci 4.2.2. Prvním přístupem je filtrování přepisovaných paketů. Druhým přístupem je využití samostatného vlákna pro filtrování bufferu. Obě metody filtrují pakety na základě hašovacích tabulek.

Vyhledávající vlákno je blokováno semaforem, dokud nejsou přidána pravidla pro filtrování. Jakmile je vlákno aktivováno, uzamyká hlavní strukturu záchytu historie pro zápis a přehazuje ukazatele na hašovací tabulky (přebírá pravidla filtru přepisovaných paketů). Dále si poznamenává aktuální začátek a konec bufferu, který bude vláknem filtrován. Dochází k uvolnění zámku komponenty pro zápis a její uzamčení pro čtení. Vlákno zahajuje průchod bufferu. Při posunu na další paket vždy aktualizuje pozici filtrovaného paketu (podmíněná proměnná průchodu bufferu), tak aby mazání nejstarších paketů nepřepsalo filtrovaný paket. Po dokončení průchodu bufferu jsou pravidla ukončena, hašovací tabulka vyčištěna a snížena hodnota semaforu (restartování stavu vyhledávajícího vlákna).

Samotné filtrování paketu probíhá podobně jako u živého záchytu (viz sekce 6.4.4). Rozdílem je obsah pravidel a načítání paketů z bufferu.

6.5.7 Informace o kruhovém bufferu

Funkce pro získání aktuálních informací o kruhovém bufferu. Během získání informací je uzamčen zámek hlavní struktury záchytu historie pro čtení. Z bufferu jsou získány informace o celkové velikosti, velikosti volného místa, počtu uložených paketů a počtu dosud nezpracovaných filtrovacích pravidlech.

6.5.8 Informace o záchytu

Pro zprostředkování aktuálních informací o záchytu určitého pravidla jsou vyžadovány argumenty IP adresa a směr záchytu. Během zjišťování stavu pravidla dochází k uzamknutí zámku hlavní struktury pro čtení. Po nalezení příslušného pravidla je vypočten aktuální stav dokončení, jako poměr mezi celkovou velikostí bufferu určenou k filtrování vůči velikosti dosud filtrované části. Výsledkem funkce je procento dokončení záchytu a počet dosud zachycených paketů.

6.5.9 Ukončení

Funkce ukončí běh vyhledávajícího vlákna, všechna pravidla záchytu, dealokuje kruhový buffer, hašovací tabulky a sdílené zámky.

Kapitola 7

Implementace ovládacích prvků

Tato kapitola popisuje implementaci prvků pro ovládání stroje času. Jedná se o klienta pro manuální ovládání stroje času. Dále o manažera distribuované správy, který zprostředkovává distribuovaný záchyt nahlášených událostí na všech měřících bodech sítě. Nakonec o modul systému NEMEA, který překládá žádosti z formátu UniRec do protokolu stroje času popsaného v příloze B.

7.1 Klient pro manuální ovládání systému

Pro manuální ovládání stroje času slouží klientská aplikace *time_machine_cli*. Tato aplikace je implementací návrhu ovládání jedné instance stroje času popsané v sekci 4.4, rozšířená o návrh distribuovaného ovládání více instancí popsané v sekci 5.2.

Pro implementaci klientské aplikace byl zvolen jazyk Python verze 2.7 [34]. Klient může být implementován v jakémkoliv jiném programovacím jazyce, jediným omezením je striktní dodržení komunikačního protokolu.

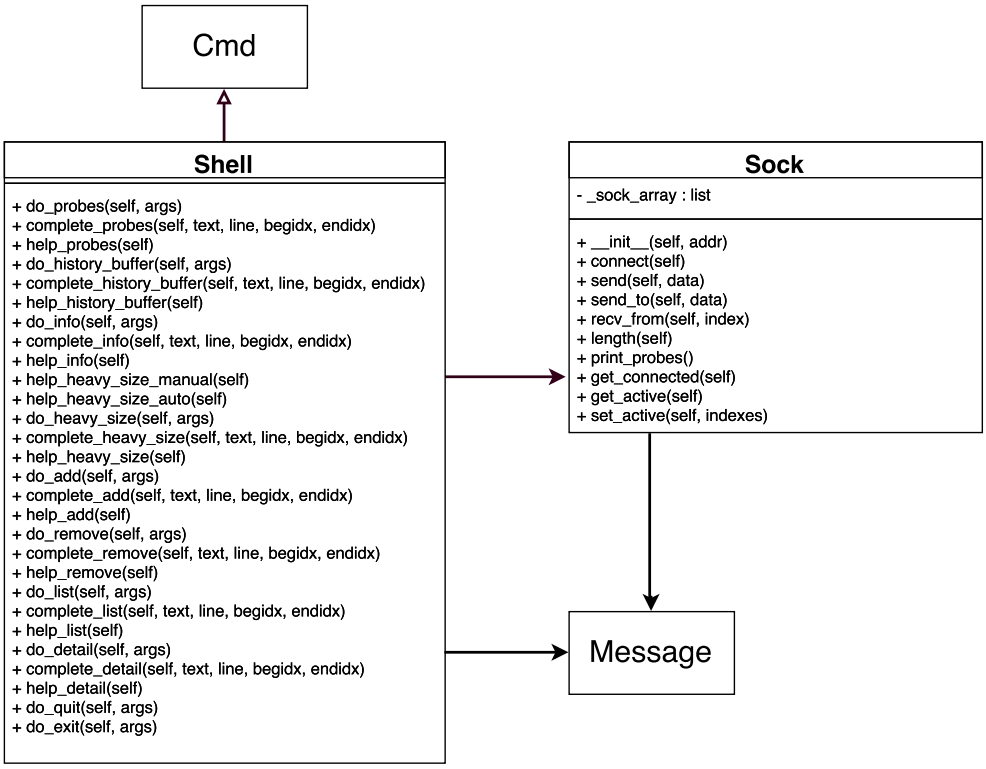
7.1.1 Diagram tříd

Jednotlivé části programu jsou rozděleny na samostatné třídy. Diagram tříd je znázorněn na obrázku 7.1.

Hlavní komponentou klientské aplikace je knihovna *cmd*, která zprostředkovává vzhled příkazového řádku. Rozšířením této knihovny o podtřídu *Shell*, jsou do příkazové řádky přidány potřebné příkazy.

Komunikační protokol je implementován jako skupina tříd, které jsou potomci třídy *Message*. Každý typ zprávy je implementován jako samostatná třída. V klientské aplikaci jsou tyto třídy použity pro vytváření žádostí a přijímání odpovědí.

Komunikaci se sondami zajišťuje třída *Sock*. Ta se spojuje se sondami, udržuje informace o sondách a pomocí třídy *Message* zasílá a přijímá zprávy.



Obrázek 7.1: Diagram tříd - Klient pro ovládání záchytu

7.1.2 Textový interpretr

Příkazy jsou implementovány ve třídě *Shell*.

Kromě příkazů *probes* a *info* mají všechny příkazy podobné zpracování. Prvně dochází k vytvoření žádosti pomocí třídy *Message*, odeslání žádosti a následné přijmutí odpovědi pomocí třídy *Sock* a zobrazení odpovědi pomocí třídy *Message*.

Příkaz *probes* zobrazuje informace o připojených sondách (ID sondy, IP adresa, port, připojeno/odpojeno a aktivní/pasivní). Tyto informace jsou získány ze třídy *Sock*. Pokud jsou za příkazem specifikovány ID sond, třída *Sock* zkontroluje, zda jsou specifikované sondy připojeny a nastaví je jako aktivní. Zbytek sond je nastaven jako pasivní.

Příkaz *info* zašle žádost o průběžné zasílání informací o záchytu. Pro přijímání a zobrazování informací, je pro každou připojenou a aktivní sondu vytvořeno speciální vlákno. Stisknutím klávesy *ENTER* dochází k odeslání žádosti o ukončení zasílání informací a ukončení vytvořeného vlákna.

7.1.3 Připojení sond

Spojení se sondami zajišťuje třída *Sock*, která udržuje seznam všech sond definovaných při spuštění programu. U každé sondy si ukládá IP adresu, port, soket, zda se podařilo se sondou spojit a zda se mají na sondu odesílat žádosti.

Hlavní funkcionality třídy jsou:

- Připojení k definovaným sondám.
- Odeslání zprávy všem sondám, které jsou připojené a aktivní
- Odeslání zprávy specifikované sondě
- Přijetí zprávy od specifikované sondy.
- Seznam všech připojených sond.
- Seznam aktivních sond (sondy nastavené pro odesílání žádostí).
- Nastavení aktivních sond.

7.1.4 Komunikační protokol

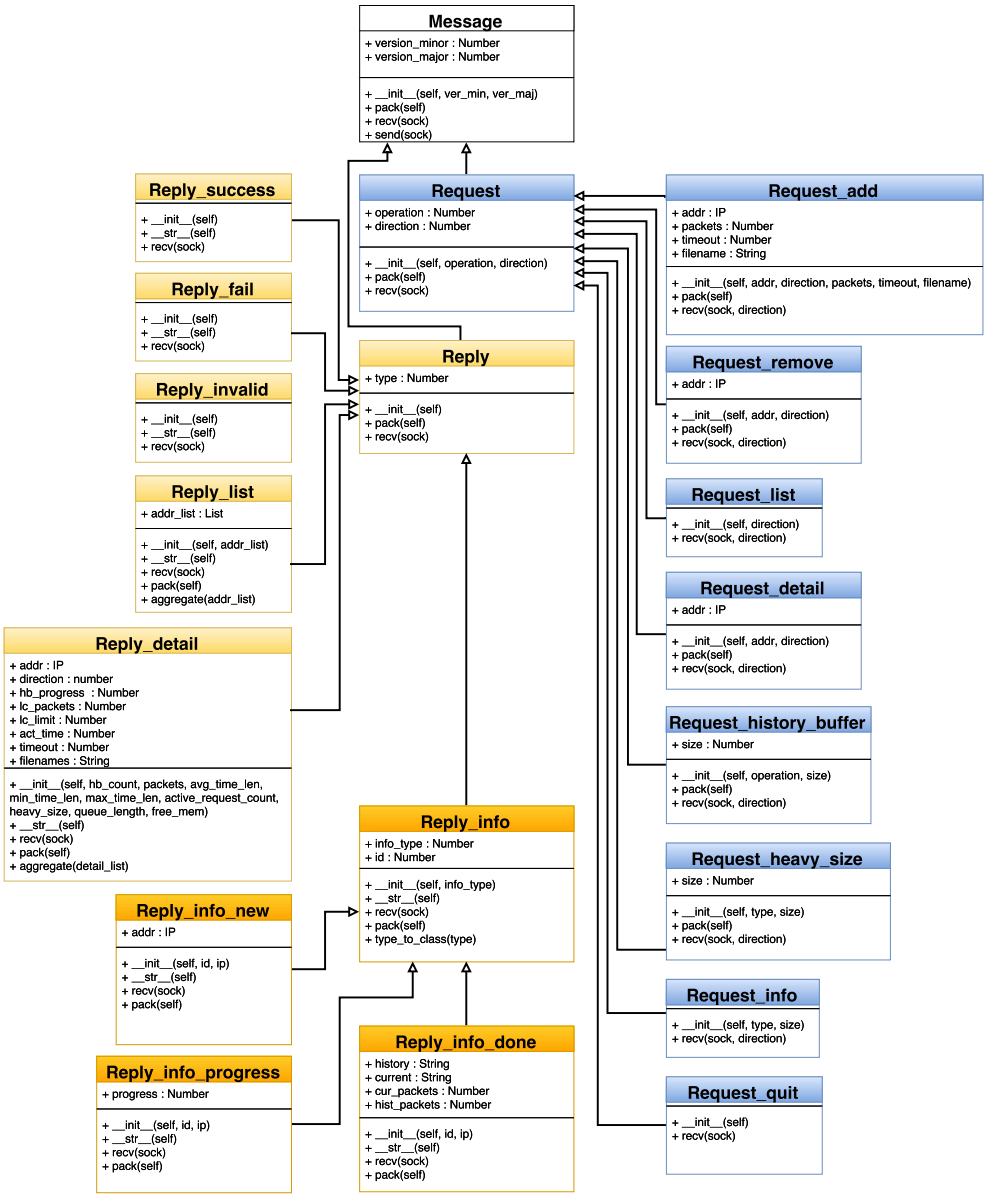
Komunikační protokol je implementován souborem tříd, které dědí společné vlastnosti a metody. Digram všech tříd komunikačního protokolu je znázorněn na obrázku 7.2.

Hlavní třídou spojující všechny typy zpráv je *Message*. Ta uchovává verzi protokolu, která je společná pro žádosti i odpovědi. Metody třídy umožňují odeslat zprávu na specifikovaný soket, přijmout zprávu ze soketu anebo konvertovat zprávu do binární podoby (pro odeslání). Podtřídami této třídy jsou *Request* a *Reply*. Ty obsahují společná data a metody všech žádostí, respektive odpovědí. Podtřídy těchto tříd implementují jednotlivé typy žádostí, respektive odpovědí definované komunikačním protokolem.

7.1.5 Ovládání programu

Program se spouští pomocí Python interpreteru. Při spuštění programu s přepínačem *-h* dojde k vypsání nápovědy, která popisuje použití ostatních přepínačů (Celá nápověda je v příloze D.1). Pomocí přepínačů lze nastavit:

- IP adresa a port sondy. Pro připojení k jedné sondě.
- Soubor s adresami a porty sond. Pro připojení k více sondám.
- Zapnutí/Vypnutí bufferování dat. Přímý příkaz bez spuštění textového interpretru.
- Vypsání informací o bufferech s historií. Přímý příkaz bez spuštění textového interpretru.



Obrázek 7.2: Diagram tříd - Komutační protokol - Python

- Nastavení začátku velikosti toku. Přímý příkaz bez spuštění textového interpretru.

Po spuštění programu (bez přepínačů, které zakazují spuštění textového interpretru) se program připojí k definovaným sondám a spustí textový interpreter. Všechny příkazy podporované interpreterem jsou popsány v příloze C.

7.2 Manažer distribuované správy systému

Jedná se o aplikaci pro distribuované ovládání systému navržené v kapitole 5. Na rozdíl od návrhu implementovaná aplikace získává data relevantní k událostem pouze ze strojů času, nikoliv z IPFIX collectoru nebo jiných systémů a neumožňuje filtrování přijatých událostí. Aplikace je proto pojmenována podle stroje času *time_machine_manager*.

Hlavní funkcionality jsou:

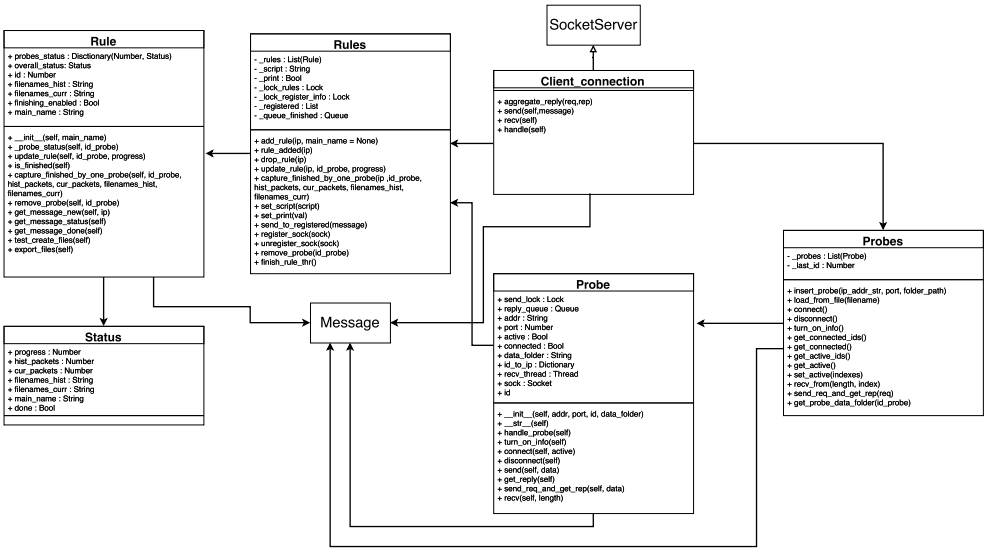
- Připojení k libovolnému počtu sond.
- Připojení libovolného počtu klientů.
- Přeposílání žádostí od klientů k sondám.
- Agregace odpovědí do jedné zprávy.
- Zprostředkování souhrnných informací o záchytu.
- Sloučení dat záchytu z jedné sondy.
- Vytvoření adresáře pro záchyt.
- Spuštění analýzy dat (libovolného skriptu) po dokončení záchytu.
- Logování přijatých žádostí a odpovědí.

Pro implementaci aplikace byl zvolen jazyk Python verze 2.7, může tak být použita stávající implementace komunikačního protokolu pro ovládání stroje času. Díky obecnému komunikačnímu protokolu je volba jazyku nezávislá na stroji času a připojených klientech, mohl by tedy být zvolen jakýkoliv jiný jazyk.

7.2.1 Diagram tříd

Jednotlivé části programu jsou rozděleny na samostatné třídy. Diagram tříd je znázorněn na obrázku 7.3.

Třída *Client_connection* zprostředkovává server pro připojení libovolného počtu klientů zasílajících žádosti. Tato třída dědí vlastnosti a metody třídy *SocketServer*. Třída *Probes* zprostředkovává komunikaci se sondami a udržuje



Obrázek 7.3: Diagram tříd - Manažer distribuované správy systému

seznam připojených sond. Každá sonda je reprezentovaná instancí třídy *Probe*. Pomocí třídy *Rules* jsou během přijímání a odesílání zpráv vytvářeny záznamy o pravidlech pro záchyt. Každé pravidlo je reprezentováno instancí třídy *Rule* a jeho stav je určen třídou *Status*. Třída *Message* a její podtřídy, které jsou popsány v sekci 7.1.4, zprostředkovávají vytváření, odesílání a příjem zpráv.

7.2.2 Komunikace s klienty

Připojení a obsluhu klientů zajišťuje třída *Client_connection*, která je podtřídou třídy *SocketServer*. Po inicializaci dochází k vytvoření portu, na kterém jsou přijímány nová spojení s klienty. Pro každého nového klienta je vytvořeno obsluhující vlákno.

Vlákno pro obsluhu klienta čeká na příchozí žádost. Ta je po přijetí přeposlána všem sondám. Následně přijatý seznam odpovědí je agregován do jedné zprávy, která je odeslána klientovi.

Pokud se jedná o žádost o záchyt či ukončení záchytu, dochází současně k přidání či odebrání pravidla v lokálním seznamu pravidel třídy *Rules*.

Pokud se jedná o žádost o zasílání aktuálních informací o záchytu, není tato žádost přeposlána sondám. Aplikace si udržuje seznam pravidel pro záchyt a jejich aktuální souhrnný stav. Dochází tedy k registraci klienta u třídy *Rules*, která tyto informace při změně stavu záchytu zasílá.

7.2.3 Komunikace se sondami

Načtení souboru s informacemi o sondách, připojení sond a komunikaci s nimi zajišťuje třída *Probes*. Během inicializace je načten textový soubor specifikovaný vstupním parametrem, který popisuje jednotlivé sondy. Každá sonda je popsána IP adresou, portem a cestou k exportovaným záchytům. Po načtení vstupního souboru je pro každou sondu vytvořena instance třídy *Probe*. Poté dojde k pokusu o připojení k jednotlivým sondám. Pro úspěšně připojené sondy je vytvořeno vlákno pro příjem zpráv a je jim zaslána žádost o odesílání aktuálních informací o záchytu. Pokud vlákno přijme zprávu s aktualizací záchytu, aktualizuje lokální seznam pravidel. Pokud přijme zprávu jiného typu, uloží jí do fronty přijatých zpráv.

Metoda pro odeslání žádosti a přijetí relevantní odpovědi odesílá zprávu jednotlivým sondám a čeká na příjem odpovědi blokujícím odebráním prvku z fronty přijatých zpráv. Zasílání zpráv sondám je přístupné všem klientským vláknům.

Pro legitimní ukončení spojení se sondou slouží metoda, která sondě odešle žádost o ukončení spojení, ukončí vlákno přijímající odpovědi a odebere sondu ze stavu všech pravidel. Při náhlém odpojení některé se sond dochází k pokusu o obnovení spojení. Probíhá celkem 10 pokusů, s časovým odstupem $i * 5$ sekund, kde i, je pořadí pokusu. Pokud jsou všechny pokusy neúspěšné, dochází ke stejné proceduře jako při legitimním ukončení spojení. V případě selhání spojení se všemi sondami, dochází k ukončení programu.

7.2.4 Správa pravidel

Pro zprostředkování aktuálních informací o záchytu, přesunu zachycených dat, sloučení dat jedné sondy a spuštění skriptu pro další analýzu zachycených dat slouží třída *Rules*. Ta udržuje seznam aktuálně zachytávaných pravidel, kde je každý záchyt reprezentován instancí třídy *Rule*.

Pravidla jsou přidávána, odebírána a aktualizována třídami *Probe* a *Client_connection*. Každé pravidlo udržuje seznam sond s aktuálním stavem záchytu. Pokud dojde ke změně celkového stavu záchytu daného pravidla, je odeslána aktualizace všem registrovaným klientům. Pokud je záchyt dokončen na všech sondách, je pravidlo přesunuto do fronty dokončených záchytů. Z této fronty jsou dokončené záchyty postupně odebírány a zpracovávány speciálním vláknem. Vlákno pro každý dokončený záchyt provede následující akce:

- Sloučí pcap soubory záchytu z jedné sondy pomocí volání aplikace *Mergecap* [35].
- Přesune soubory do nově vytvořeného adresáře pojmenovaného názvem záchytu.
- Vytvoří textový soubor se jmény všech zachycených souborů, počtem jejich paketů a jmen sond, ze kterých jsou exportovány.

- Spustí volitelný skript, kterému se jako parametr předá jména sloučených souborů.
- Odešle informace o dokončeném záchytu registrovaným klientům.

7.2.5 Ovládání programu

Program se spouští pomocí Python interpreteru. Při spuštění programu s přepínačem *-h* dojde k vypsání nápovědy, která popisuje použití ostatních přepínačů (Celá nápověda je v příloze D.2).

Pomocí přepínačů lze ovlivnit:

- Cestu k souboru s informacemi o sondách.
- Port serveru, na který se připojují klienti.
- Soubor pro záznam chování programu (přijaté a odeslané zprávy, výpadky spojení, prováděné akce).

Po spuštění programu, dojde k připojení definovaných monitorovacích sond a otevření portu pro příjem klientských spojení. Aplikace může být ukončena klávesovou zkratkou CTRL + C.

7.3 NEMEA modul pro přeposílání událostí

Přeposílání událostí detekovaných systémem NEMEA do systému pro záchyt dat zajišťuje NEMEA modul pojmenovaný *nemea2time_machine*. Modul je implementovaný v jazyce C a jeho hlavní funkcionalitou je převod přijaté události ve formátu UniRec do formátu komunikačního protokolu stroje času.

Modul obsahuje pouze hlavní funkci *main()*, ve které postupně vykonává:

1. Inicializace knihovny Libtrap pro příjem událostí ve formátu UniRec.
2. Přijetí vstupních parametrů funkce. Modul umožňuje nastavit:
   - IP adresu a port monitorovací sondy nebo manažera distribuované správy.
   - Jméno záchytu, které bude rozšířené o zachytávanou IP adresu.
   - Směr záchytu.
   - Počet zachycených paketů k události.
   - Časový limit záchytu.
3. Vytvoření vstupní UniRec šablony. Šablona obsahuje IP adresu a pokud není specifikováno vstupním parametrem, tak: jméno záchytu, směr záchytu, počet zachycených paketů a časový limit záchytu. Jména UniRec položek a jejich datový formát je v příloze D.3.1.

4. Vytvoření spojení se sondou či manažerem distribuované správy.

5. Hlavní smyčka modulu.

V hlavní smyčce modulu jsou přijímány události na vstupním rozhraní. Pomocí definovaného protokolu jsou odesílány na vytvořené spojení se sondou či manažerem pravidel jako žádost o přidání pravidla pro záchyt. Po úspěšném odeslání žádosti je přijata odpověď, která je vypsána na standardní výstup.

Nápověda programu je zobrazena v příloze D.3.

Kapitola 8

Testování rozšířené monitorovací sondy

8.1 Testovací prostředí

Implementovaný mechanismus automatického zachytávání byl testován na síti CESNET2. Pro testování automatického záchytu událostí se systémem SDM byla vyhrazena jedna sonda nacházející se na hranici sítě CESNET2 a ACONET [36]. Sonda monitoruje oba směry a pro sběr dat používá 80 Gb/s hardwarově akcelerovanou síťovou kartu s podporou SDM. Na sondě je k dispozici 64 GB paměti RAM s dvojicí procesorů Intel Xeon E5-2620v2.

Během měření byla maximální propustnost měřené sítě zhruba 12 Gb/s v obou směrech (16 000 000 paketů/s, 60 000 toků/s). Dlouhodobý průměr poté činil zhruba 2 Gb/s v obou směrech (250 000 paketů/s, 15 000 toků/s).

8.2 Dynamická analýza kódu

Zásuvný modul byl otestován v programu Valgrind [37]. Tento program neidentifikoval žádné chyby a ani neuvolněnou paměť po ukončení běhu modulu.

8.3 Testování funkčnosti

Jednotlivé části zásuvného modulu byly otestovány v reálném prostředí komunikační linky. Následující seznam popisuje testování jednotlivých funkčních prvků.

- Komunikační protokol

  Komunikační protokol byl otestován pomocí vytvořeného klienta. Byly vyzkoušeny všechny typy žádostí a přijaty všechny typy odpovědí.

- Počet klientů

  Bylo otestováno současné připojení 100 klientů k zásuvnému modulu, což je maximální počet definovaný konstantou modulu. Všichni klienti bez problémů zasílaly žádosti a přijímaly odpovědi od zásuvného modulu.

- Příjem paketů

  Všechna data přijatá aplikací Flowmonexp byla filtrována živým záchytem a následně byly pakety začátku toku uloženy do bufferu. Při testování nedocházelo k zahazování paketů.

- Živý záchyt

  Záchyt byl otestován přidáním pravidel pro záchyt a následným validováním vytvořených pcap souboru. Všechny zachycené soubory byly validní a obsahovaly data určité adresy a směru. Podle záznamů síťových toků, počet uložených paketů odpovídal počtu paketů této adresy na lince.

- Filtrování bufferu

  Do aplikace byla přidána možnost kompilace v testovacím režimu pomocí přidání konstanty *TEST_BUFFER*. V tomto režimu je ke každé hlavičce paketu přidána speciální hodnota, která je při průchodu bufferu testována. Tímto testem je zaručen správný přístup k datům bufferu. Stejně jako při testování živého záchytu, byl otestován obsah pcap souborů se zachycenými daty.

- Počet současně zachytávaných událostí

  Maximální počet zachytávaných pravidel je omezen dvěma faktory. Prvním omezením je počet pravidel v hašovací tabulce. Ta je v základu nastavena na pojmutí maximálně 128 prvků s rozdílnou haší a 1024 prvků celkově (při zaplnění každého řádku i sloupce tabulky). Při současném zachytávání 1024 událostí a použití osmi instancí záchytu historie, by bylo definováno celkem 8192 pravidel pro komponentu záchyt historie a 1024 pravidel pro komponentu živý záchyt. Každé pravidlo vytváří vlastní soubor, tudíž pro 1024 událostí by bylo vytvořeno 9216 současně otevřených souborů.

  Dalším omezením je maximální počet současně otevřených souborů jedním procesem, který je na testovaném systému roven 1024 (lze zvětšit). Při použití 8 instancí komponenty záchyt historie, je pro každou událost vytvořeno celkem 9 souborů. Maximální počet současně zachytávaných událostí je tedy 113.

  Nejvíce omezujícím faktorem je propustnost pevného disku. Při zvětšujícím se objemu provozu určeného k zápisu na pevný disk dochází k většímu zpoždění a možnosti zahazování nově přijatých paketů.

Na sondu bylo zasláno 113 pravidel pro záchyt, které odpovídaly 26 % dat provozu. Ze všech pravidel byly úspěšně vytvořeny pcap soubory se záchyty, a to bez zahazování paketů. Při zaslání většího počtu žádostí o záchyt, nebylo možné vytvořit požadované množství souborů, a proto nebyly žádosti úspěšně potvrzeny.

8.4 Velikost začátku toku

Pro různou velikost začátku toku byla naměřena délka uložených dat. Referenční velikost bufferu s historií byla 8 GB pro všechny instance. Velikost začátku toku se pohybovala od 2 paketů až po 200 paketů a zvyšovala se po dvou paketech. Celé testování trvalo 12 hodin a bylo opakováno 3x za sebou. Během testování se měnila velikost průtoku dat a to od 0,5 Gb/s až po 2,68 Gb/s. Tyto výkyvy ovlivňovali výsledné hodnoty měření, proto byly výsledky normalizovány pro průměrnou rychlost měření 1,5 Gb/s. Obrázek 8.1 znázorňuje výsledky měření testovaných velikostí začátku toku. Jelikož délka uložených dat od hodnoty 32 paketů klesá velice pomalu, obrázek obsahuje 2 grafy. První graf zobrazuje všechny měřené hodnoty a druhý graf pouze velikosti toku od 2 do 32 paketů.

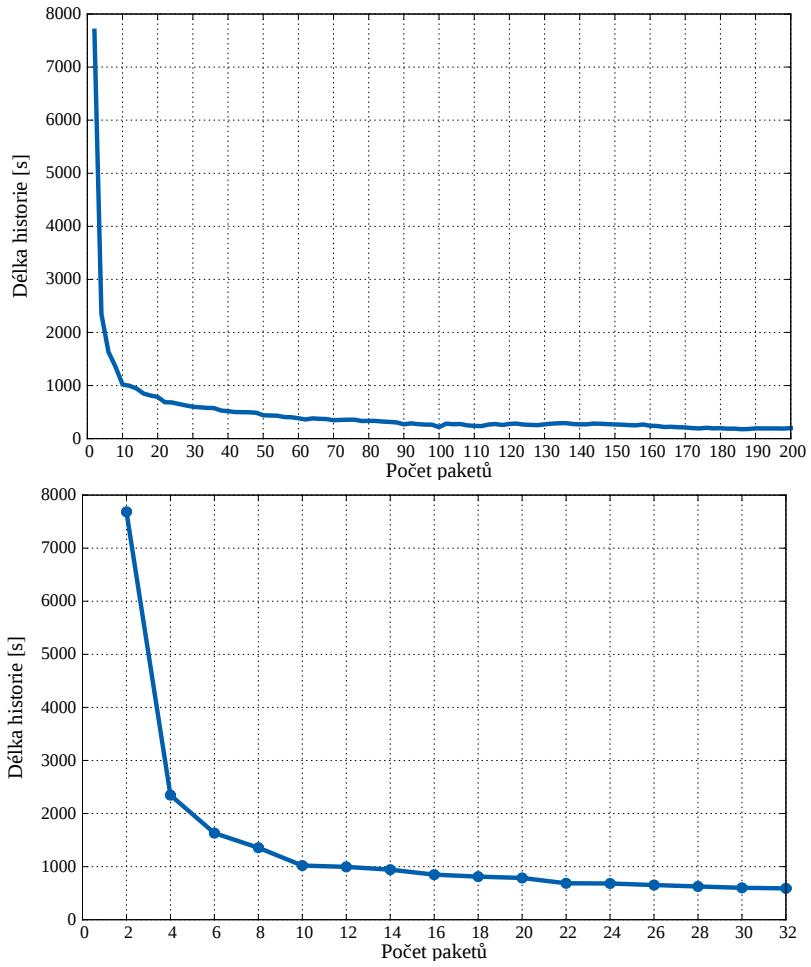
8.5 Velikost bufferu

Kruhový buffer pro ukládání začátků toků využívá paměť RAM. Celková paměť testované sondy je 64 GB. Dočasné ukládání historie bylo zprostředkováno pomocí osmi instancí komponenty záchyt historie. Byly testovány velikosti 0,5 GB, 1 GB, 2 GB, 4 GB, 8 GB, 16 GB, 32 GB, 56 GB a 64 GB. Pro velikosti od 0,5 GB až 56 GB nedocházelo ke změnám vytížení procesoru nebo k zahazování paketů. Při použití 64 GB pro paměť bufferů již docházelo ke swapování paměti na pevný disk. Toto swapování zpomalovalo ukládání do bufferu a příchozí pakety začaly být zahazovány.

Velikost paměti je vůči délce historie lineární. Délka uložené historie závisí na aktuální propustnosti linky, velikosti paketů a konkrétních tocích.

8.6 Vytížení procesoru

Vytížení procesoru se na testované sondě dlouhodobě pohybuje na průměru 35 %. Obrázek 8.2 znázorňuje vytížení procesoru po zapnutí Flowmonexp se zásuvným modulem pro záchyt dat. Zelená čára symbolizuje paměť RAM, její postupné zvyšování odpovídá zaplňování bufferů. Po zaplnění bufferů, začne přepisování nejstarších paketů. Na začátku přepisování se krátkodobě zvýší vytížení procesoru, které je způsobeno přístupem do jiné části paměti (na začátek bufferu). Dalším bodem na grafu je přijetí pravidla pro záchyt, které

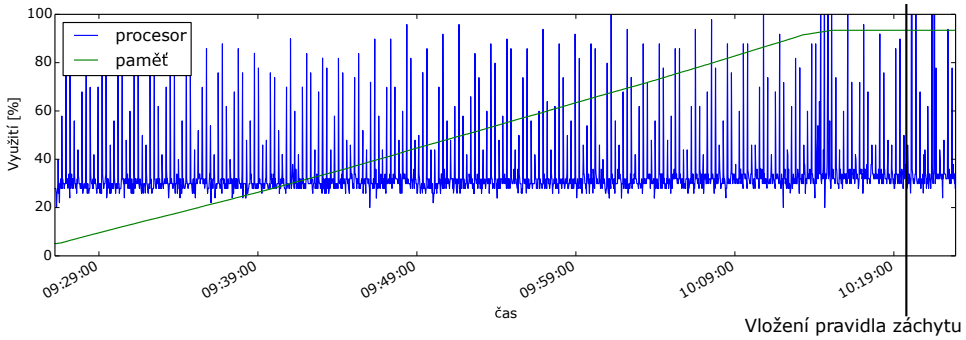


Obrázek 8.1: Délka uložených dat v závislosti na velikosti začátku toku (Velikost bufferu 8 GB)

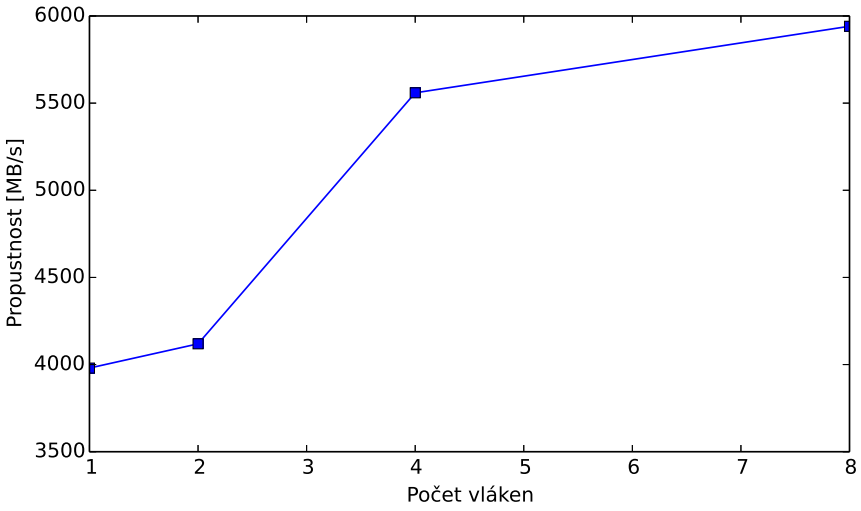
způsobí krátkodobé zvýšení využití procesoru při prohledávání kruhových bufferů. Živý záchyt významně neovlivňuje vytížení procesoru.

8.7 Datová propustnost

Pro měření datové propustnosti bylo vytvořeno testovací prostředí mimo aplikaci Flowmonexp. Jako vstupní data byly použity informace o paketech z reálné sítě CESNET2, které odpovídají popisu v kapitole 2. Informace obsahovaly anonymizované IP adresy, velikosti paketů a pořadí paketů v toku. Obsah paketů byl náhodně generován. Celkem byl takto simulován provoz o velikosti zhruba 13 GB, 18 milionů paketů. Každý test byl 5x opakován a výsledné hodnoty byly zprůměrovány.



Obrázek 8.2: Vytížení procesoru a paměti

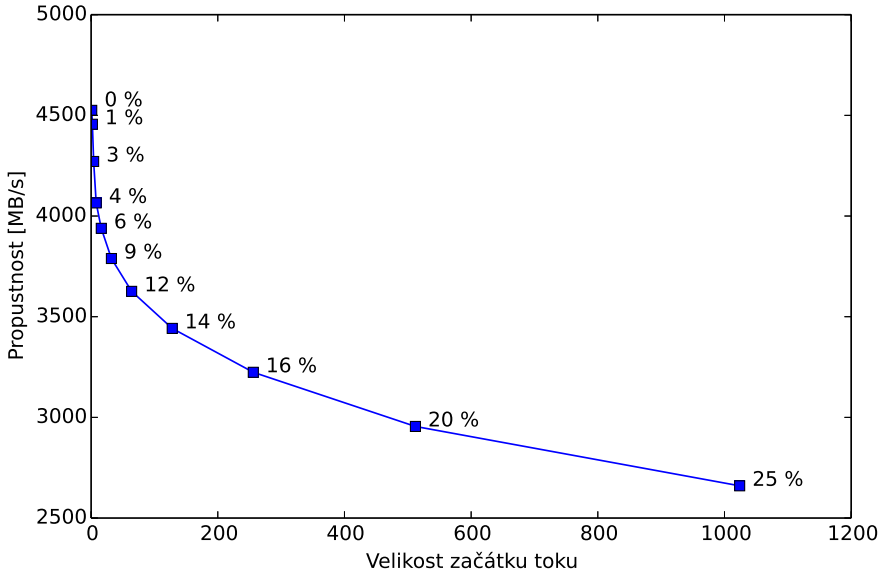


Obrázek 8.3: Datová propustnost stroje času v závislosti na počtu vláken

8.7.1 Počet vláken

Tento test měří maximální propustnost stroje času při rozdílném počtu ukládajících vláken. Byly testovány hodnoty 1, 2, 4 a 8 vláken. Velikost bufferu s historií byla nastavena na 8 GB, při zvýšení počtu vláken se tato velikost rozloží na více instancí záchytu historie. Velikost začátku toku byla nastavena během celého měření na 20 paketů.

Výsledek testu je na obrázku 8.3. Dle naměřených hodnot je vidět, že více vláken zvyšuje maximální propustnost jen velice pozvolně. To je způsobené maximální propustností komunikace s pamětí RAM.



Obrázek 8.4: Datová propustnost stroje času v závislosti na velikosti začátku toku

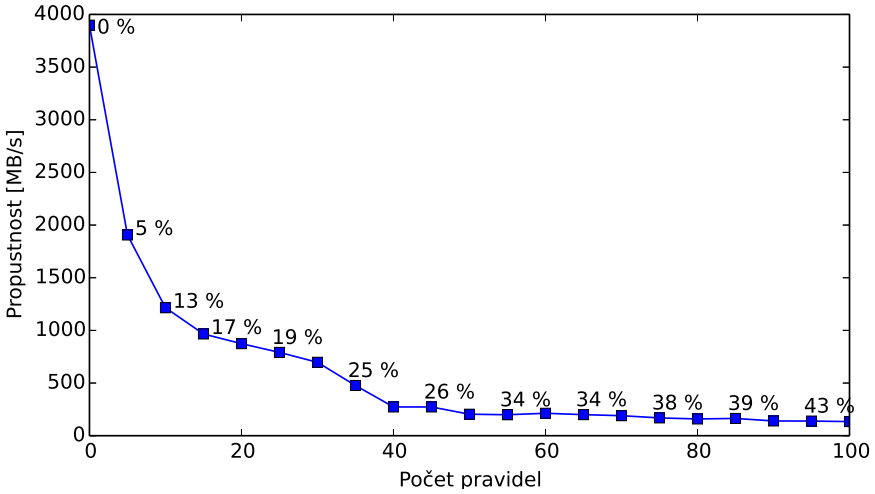
8.7.2 Velikost začátku toku

Tento test měří maximální propustnost stroje času v závislosti na nastavení velikosti začátku toku. Jelikož se do bufferu s historií ukládá jen začátek toku, čím vyšší je tato hodnota, tím více provozu se musí uložit do bufferu. Byly testovány násobky dvou a to od 1 až 1024 paketů. Velikost bufferu s historií byla nastavena na konstantních 8 GB. Vlákno pro ukládání bylo pouze jedno.

Výsledek testu je na obrázku 8.4. Na ose X je znázorněn počet paketů, který je nastaven jako velikost začátku toku. Hodnoty u jednotlivých bodů poté znázorňují poměr velikosti bufferovaného provozu vůči celkové velikosti provozu. Největší rozdíl v propustnosti je vidět u nízkého počtu paketů, který je způsobený velkým množstvím krátkých toků na síti. Se zvyšujícím se počtem paketů klesá propustnost a poměr uložené komunikace mnohem pomaleji.

8.7.3 Počet vložených pravidel

Tento test měří maximální propustnost stroje času v závislosti na počtu vložených pravidel. Čím více pravidel je ve stroji času, tím více provozu je třeba zachytávat a ukládat na disk. Byly testovány hodnoty od 0 do 100 po pěti pravidlech. Pro každé pravidlo byla náhodně vybrána unikátní IP adresa ze vstupních dat. Velikost bufferu s historií byla nastavena na konstantních 8 GB.



Obrázek 8.5: Datová propustnost stroje času v závislosti na počtu pravidel

Vlákno pro ukládání bylo pouze jedno.

Výsledek testu je na obrázku 8.5. Na ose X je znázorněn počet vložených pravidel. Hodnoty u jednotlivých bodů poté znázorňují poměr velikosti zachyceného provozu vloženými pravidly vůči celkové velikosti provozu. Snižování propustnosti je ovlivněné především velikostí zachytávaného provozu. Záleží tedy především na provozu konkrétních IP adres, které jsou v pravidlech.

Kapitola 9

Záchyt dat a jejich analýza

9.1 Testovací architektura

Distribuovaný záchyt dat byl testován na osmi měřících bodech sítě CESNET2. Na všech těchto sondách byla nastavena velikost bufferu s historií 8 GB a velikost začátku toku 20 paketů.

Pro manažera pravidel byl vyhrazen virtuální stroj, který udržuje spojení se sondami a detekčním systémem NEMEA, dále se stará o zpracování zachyceného provozu. Události pro záchyt byly nahlášeny detekčními moduly systému NEMEA popsanými v sekci 2.1.4. Z důvodu vysokého počtu nahlášených událostí byly za detekční moduly napojeny filtry, které odesílají maximálně jednu událost každých 10 minut. Filtr preferuje události s nejvyšším počtem nahlášení. V případě více událostí se stejným počtem nahlášení je preferována nejnovější událost. Po odeslání události není v následující hodině tato událost odeslána znovu. Pro všechny události je nastaven časový limit záchytu 5 minut a 10 000 paketů.

Po dokončení záchytu jsou data odeslána manažeru pravidel, který nad nimi spouští automatickou analýzu pomocí systémů Bro a Tranalyzer [38]. Tranalyzer je analyzátor paketů, který na základě pcap souboru vytvoří soubory s informacemi o provozu (použité protokoly, statistické informace z hlaviček jednotlivých protokolů, a další). Výsledky této analýzy jsou uloženy k zachycenému provozu.

9.2 Sběr dat a jejich vyhodnocení

Za jeden měsíc bylo celkem nashromážděno 15 162 záchytů událostí, které odpovídají zhruba 72 GB dat. Největší událost byla zachycena pomocí hlášení DNS tunnel detektoru. Jednalo se o DDoS útok na DNS server uvnitř sítě CESNET2. Celková velikost zachyceného provozu byla 30 GB, z toho 104 MB dat živého záchytu a zbytek ze záchytu historie. Útok procházel skrz všechny monitorovací sondy.

Pomocí nástroje Bro se podařilo verifikovat pouze síťové skeny. U některých událostí Bro rozpoznalo neplatný SSL certifikát serveru. Ostatní události nebyly rozpoznány i přesto, že manuální analýzou paketů byla ověřena správnost detekce.

Následující seznam popisuje získaná data jednotlivými detektory.

- DNS amplification

  Pomocí získaných dat, se dá jednoduše ověřit, že se jedná o útok DNS amplifikací. Pokud je zdroj i cíl útoku v síti CESNET2, lze v datech vypozorovat žádosti typu *ANY* na DNS server a následně několik desítek paketů odpovědí na tento dotaz. Pro úspěšné rozpoznání útoku není třeba dlouhý záznam provozu, stačí záznam historie.

- DNS tunnel detector

  Detektor umožňuje nahlásit DNS tunely nebo DoS útoky přes DNS. U tunelů lze v datech vypozorovat, ve kterých políčkách dochází ke kódování dat jak v žádostech, tak v odpovědích. Lze tak jednoduše verifikovat že se jedná o tunel. Takto podrobné informace nelze získat z rozšířených síťových toků. DNS tunel nemusí obsahovat mnoho provozu, nedá se tedy spolehnout pouze na záchyt historie a je třeba nastavit delší časový limit živého záchytu pro získání potřebné velikosti komunikace.

  Nalezené DoS útoky přes DNS obsahující stovky žádostí o překlad stejného doménového jména ze stejné zdrojové adresy. Tento typ útoku lze verifikovat pomocí rozšířených síťových toků. Není tedy třeba tento typ událostí zachytávat.

- Hoststats

  Většina detekcí modulu Hoststats obsahuje síťové skeny. Jelikož je tento typ útoku velice jednoduché verifikovat i ze základních síťových toků, není třeba tento typ události zachytávat. U zbytku útoků hlášených pomocí Hoststats (DoS, SSH brutforce a DNS amplifikace) nám pro ověření stačí záchyt historie. Tyto útoky mají vysokou frekvenci paketů a detekční zpoždění je nízké (do 5 minut), pokud je v bufferu uložených alespoň 5 minut, není třeba spouštět živý záchyt.

- Vertical and Horizontal scan detector

  Horizontální i vertikální skeny lze jednoduše ověřit pomocí základních síťových toků. Tyto události jsou velice časté a jejich záchyt nám nepřináší nové informace. V některých případech může po oskenování sítě docházet k následnému útoku, proto je vhodnější zachytávat provoz adres, které skenují síť a následně je z této adresy detekován určitý druh útoku.

- Voip fraud detector

  Samotné hádání prefixu vytáčeného čísla lze ověřit pomocí rozšířených síťových toků. Záchyt dat těchto událostí nám umožní rozpoznat další aktivity útočníka, zda dokázal zjistit tajný prefix a zda využil znalost prefixu pro vytvoření hovoru. V zachycených datech se objevilo několik útoků, kdy byl uhodnut tajný prefix a vytvořen hovor. VoIP fraud útok může být nízkofrekvenční, pro jeho úspěšné zachycení tedy nelze spoléhat pouze na záchyt historie a je vhodné nastavit větší časový limit živého záchytu.

Pro automatickou verifikaci zachyceného provozu je třeba najít či vytvořit vhodný detektor, kterému nebude vadit chybějící zbytek síťového toku a rozpozná anomálie v malém vzorku již vyfiltrovaných dat. Bro se pro automatickou verifikaci takto zachyceného provozu neosvědčilo.

Použité filtrování událostí preferuje vždy události, které mají největší pravděpodobnost dlouhodobého výskytu na síti. Tímto filtrováním se nezachytí data událostí, které byly nahlášeny pouze jednou, ale přesto by jejich záchyt poskytl cenné informace. Další nevýhodou tohoto filtrování je zanedbání závislostí mezi adresami nahlášenými z různých detektorů. Z těchto důvodů je třeba nahradit stávající filtrování rozšířením manažera událostí o vstupní komponentu inteligentního filtrování. Filtr by měl přijímat všechny nahlášené události, ohodnotit přínos jejich záchytu a spustit záchyt jen u událostí s vysokým hodnocením.

Závěr

Na vysokorychlostních sítích se často používají nástroje pro analýzu a detekci anomálií založené na síťových tocích, které agregují informace o provozu. Problém nahlášených bezpečnostních událostí pomocí těchto nástrojů je neexistence důkazního materiálu.

Cílem práce bylo vytvořit systém pro sběr důkazního materiálu k nahlášeným bezpečnostním událostem a následně tento systém integrovat do stávající monitorovací architektury sítě CESNET2. Přínos této práce je možné rozdělit do následujících kroků.

Prvním krokem byla analýza nástrojů pro monitorování počítačových sítí. Zde byly popsány možnosti extrakce dat ze sítě, vytváření síťových toků a několik systémů pro analýzu provozu a detekci anomálií.

Druhým krokem byla analýza monitorovací infrastruktury a provozu sítě CESNET2. Jelikož systém cílí na použití v této síti, bylo velice důležitým krokem popsat jednotlivé prvky, které jsou součástí monitorovací infrastruktury. Jednalo se především o monitorovací sondy využívající aplikaci Flowmonexp a detekční systém NEMEA. Pomocí analýzy událostí nahlášených systémem NEMEA byly stanoveny základní požadavky pro návrh výsledného systému.

Třetím krokem byla analýza existujících řešení. V tomto kroku bylo popsáno několik systémů, které se snaží řešit problém záchytu dat k nahlášeným událostem. U každého systému byl diskutován přínos použití v síti CESNET2. Žádný ze systémů však nesplňoval veškeré požadavky sítě CESNET2.

Čtvrtým krokem byl návrh rozšíření monitorovacích sond o možnost záchytu vybraného provozu. Navržený systém se skládá z živého záchytu a záchytu historie. Byly popsány možnosti realizace obou těchto částí, tak aby byly operace záchytu efektivní. Následně byl popsán komunikační protokol pro vzdálené ovládání monitorovací sondy a klientská aplikace pro manuální konfiguraci sondy.

Pátým krokem byl návrh distribuované architektury pro automatizovaný záchyt dat událostí. Jelikož je na větších sítích několik monitorovacích sond, je třeba umožnit záchyt na každé z nich. Z tohoto důvodu byl navržen systém,

který se stará o komunikaci s jednotlivými sondami, sbírá zachycená data a provádí nad nimi automatickou paketovou analýzu. Systém přijímá události od detekčního systému.

Šestým krokem byla implementace systému. Prvně bylo implementováno rozšíření monitorovací sondy, následně systém pro distribuovanou správu sond, a nakonec klientská aplikace pro ovládání záchytu a konfiguraci jednotlivých sond.

Posledním krokem bylo základní testování funkčnosti systému, měření parametrů systému a analýza zachycených dat. Během analýzy zachycených dat byly popsány nedostatky systému, které jsou způsobené nevhodným filtrováním událostí pro záchyt a nevhodnými nástroji pro automatickou analýzu zachyceného provozu. Oba tyto nedostatky řeší budoucí rozšíření systému záchytu dat nahlášených událostí o inteligentní filtr událostí a analyzátor pcap souborů.

Implementovaný systém byl úspěšně nasazen do prostředí sítě CESNET2, kde aktivně slouží pro záchyt dat nahlášených událostí. Tato práce byla popsána v článku [39] a prezentována na odborné konferenci o počítačových sítích v Kanadě.

Literatura

[1] Brownlee, N.; Mills, C.; Ruth, G.: Traffic flow measurement: Architecture. *Traffic*, 1999.

[2] Claise, B.: Cisco Systems NetFlow Services Export Version 9. RFC 3954, RFC Editor, October 2004. Dostupné z: `http://www.rfc-editor.org/rfc/rfc3954.txt`

[3] Trammell, B.; Boschi, E.; Mark, L.; aj.: Specification of the IP Flow Information Export (IPFIX) File Format. RFC 5655, RFC Editor, October 2009. Dostupné z: `http://www.rfc-editor.org/rfc/rfc5655.txt`

[4] Puš, V.; Kekely, L.; Špinler Martin; aj.: Hardware accelerator for 100 Gbps network traffic monitoring. Tech. rep, CESNET, z.s.p.o., 2014. Dostupné z: `https://www.cesnet.cz/wp-content/uploads/2015/01/hanic-100g.pdf`

[5] Duffield, N.; Lund, C.; Thorup, M.: Charging from sampled network usage. In *Proceedings of the 1st ACM SIGCOMM Workshop on Internet Measurement*, ACM, 2001, s. 245–256.

[6] Kekely, L.; Puš, V.; Kořenek, J.: Software defined monitoring of application protocols. In *INFOCOM, 2014 Proceedings IEEE*, IEEE, 2014, s. 1725–1733.

[7] Kekely, L.; Kučera, J.; Puš, V.; aj.: Software Defined Monitoring of Application Protocols. *IEEE Transactions on Computers*, ročník 65, č. 2, 2016: s. 615–626, ISSN 0018-9340, doi:10.1109/TC.2015.2423668.

[8] Puš, V.; Kekely, L.; Kořenek, J.: Design methodology of configurable high performance packet parser for FPGA. In *Design and Diagnostics of Electronic Circuits & Systems, 17th International Symposium on*, IEEE, 2014, s. 189–194.

[9] CESNET, z.s.p.o.: COMBO card family. 2016. Dostupné z: https://www.liberouter.org/technologies/cards/

[10] Roesch, M.; aj.: Snort: Lightweight Intrusion Detection for Networks. In *LISA*, ročník 99, 1999, s. 229–238.

[11] Libpcap. 2016. Dostupné z: http://www.tcpdump.org/

[12] Khalil, G.: Open Source IDS High Performance Shootout. In *InfoSec Reading Room*, SANS Institute, 2015.

[13] Bro.org: Bro Documentation. 2016. Dostupné z: https://www.bro.org/sphinx/intro/index.html

[14] Čejka, T.; Bartoš, V.; Švepeš, M.; aj.: NEMEA: A Framework for Network Traffic Analysis. In *12th International Conference on Network and Service Management (CNSM 2016), Montreal, Canada*, 2016.

[15] Shafranovich, Y.: Common Format and MIME Type for Comma-Separated Values (CSV) Files. RFC 4180, RFC Editor, October 2005. Dostupné z: http://www.rfc-editor.org/rfc/rfc4180.txt

[16] Haag, P.: Nfdump. 2010. Dostupné z: http://nfdump.sourceforge.net

[17] Liberouter: Nemea-Framework. https://github.com/CESNET/Nemea-Framework.

[18] Švepeš, M.: *Systém pro konfiguraci a monitorování distribuovaného systému NEMEA*. Bakalářská práce, České vysoké učení technické v Praze, 5 2014.

[19] CESNET, z.s.p.o.: Sít CESNET2. 2016. Dostupné z: https://www.cesnet.cz/sluzby/pripojeni/sit-cesnet2/

[20] FLOWMON NETWORKS, A.S. 2016. Dostupné z: https://www.flowmon.com

[21] IPFIX collector. https://github.com/CESNET/ipfixcol.

[22] Goode, B.: Voice over internet protocol (VoIP). *Proceedings of the IEEE*, ročník 90, č. 9, 2002: s. 1495–1517.

[23] Rosenberg, e. a.: SIP: Session Initiation Protocol. RFC 3261, RFC Editor, June 2002. Dostupné z: http://www.rfc-editor.org/rfc/rfc4180.txt

[24] Čejka, T.; Bartoš, V.; Truxa, L.; aj.: Using Application-Aware Flow Monitoring for SIP Fraud Detection. In *Intelligent Mechanisms for Network Configuration and Security*, Springer, 2015, s. 87–99.

[25] Mockapetris, P. V.: Domain names: Implementation specification. 1983.

[26] Čejka, T.; Rosa, Z.; Kubátová, H.: Stream-wise detection of surreptitious traffic over dns. In *Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2014 IEEE 19th International Workshop on*, IEEE, 2014, s. 300–304.

[27] Čejka, T.; Švepeš, M.: Analysis of Vertical Scans Discovered by Naive Detection. In *IFIP International Conference on Autonomous Infrastructure, Management and Security*, Springer, 2016, s. 165–169.

[28] Kornexl, S.; Paxson, V.; Dreger, H.; aj.: Building a time machine for efficient recording and retrieval of high-volume network traffic. In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, USENIX Association, 2005, s. 23–23.

[29] McAfee Network Security Platform. 2016. Dostupné z: `http://www.mcafee.com/us/products/network-security-platform.aspx`

[30] Tutorials Point. Dostupné z: `http://www.tutorialspoint.com/index.htm`

[31] Tinetti, F. G.: Distributed systems: principles and paradigms. *Journal of Computer Science & Technology*, ročník 11, 2011.

[32] Herlihy, M.: Wait-free synchronization. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, ročník 13, č. 1, 1991: s. 124–149.

[33] Butenhof, D. R.: *Programming with POSIX threads*. Addison-Wesley Professional, 1997.

[34] Python. Dostupné z: `https://www.python.org/`

[35] Mergecap. Dostupné z: `https://www.wireshark.org/docs/man-pages/mergecap.html`

[36] ACONET: 2016. Dostupné z: `https://www.aco.net/topologie`

[37] Valgrind. Dostupné z: `http://valgrind.org/`

[38] Rühl, T.; Bühlmann, F.; Burschka, S.: Tranalyzer. Dostupné z: `https://tranalyzer.com/`

[39] Rosa, Z.; Čejka, T.; Žádník, M.; aj.: Building a Feedback Loop to Capture Evidence of Network Incidents. In *12th International Conference on Network and Service Management (CNSM 2016), Montreal, Canada*, 2016.

Příloha A

Seznam použitých zkratek

DMA Direct Memory Access

FPGA Field Programmable Gate Array

HANIC Hardware Accelerated Network Interface Card

IP Internet Protocol

IPFIX Internet Protocol Flow Information Export

PCIe Peripheral Component Interconnect Express

SDM Software Defined Monitoring

TRAP Traffic Analysis Platform

VoIP Voice over Internet Protocol

Příloha B

Komunikační protokol

Komunikační protokol definuje dva typy zpráv, žádosti (zprávy od klienta ke stroji času) a odpovědi (zprávy od stroje ke klientovi). Každá zpráva obsahuje hlavičku identifikující verzi komunikačního protokolu. Server a s ním komunikující klient musí mít stejnou verzi protokolu, v opačném případě server ukončí s klientem spojení. Zbytek zprávy žádosti a odpovědi je odlišný.

B.0.1 Žádosti

Žádosti jsou určeny pro ovládání stroje času a zadání pravidla pro vyhledání paketů určité IP adresy. Struktura žádosti je znázorněná v tabulce B.1. První dva bajty na začátku zprávy jsou pro specifikaci verze. Další dva bajty určují požadovanou operaci. Typ žádosti specifikuje druh požadované operace (například přidání IP adresy pro záchyt) a cíl žádosti upřesňuje tuto operaci (například zdrojová IP adresa). Zbytek zprávy je specifikován konkrétní operací.

Tabulka B.1: Struktura zprávy - Hlavička žádosti

Bajty	**0**	**1**	**2**	**3**
0-3	majoritní verze	minoritní verze	typ žádosti	cíl žádosti

Následující seznam popisuje všechny podporované typy žádostí indexované jejich hodnotou.

0. Záchyt – (Tabulka B.2) Pravidlo pro zachycení dat IP adresy. Pravidlo je předáno jak živému záchytu, tak záchytu historie. Cíl žádosti určuje směr ve kterém je IP adresa vyhledávána (0 - nespecifikovaný, 1 - zdroj, 2 - cíl). Zbytek zprávy obsahuje IP adresu (všechny IP adresy jsou v pořadí big-endian), limit paketů pro živý záchyt, časový limit pro živý záchyt [s], délku jména záchytu (počet znaků) a jméno záchytu. Jméno záchytu je použito pro pojmenování souborů se záchytem. V případě záchytu

historie je vytvořeno více souborů. Jména těchto souborů jsou rozšířena o koncovku *_hc_i*, kde i je ID vlákna. Jméno záchytu je omezeno délkou 255 znaků, protože je pro něj v komunikačním protokolu vyhrazen pouze jeden bajt.

Tabulka B.2: Struktura zprávy - Žádost o záchyt

Bajty	**0**	**1**	**2**	**3**
4-7	IP adresa			
8-11				
12-15				
16-19				
20-23	limit paketů			
24-27	časový limitů			
28-...	délka jména	jméno záchytu		

1. Ukončení záchytu – (Tabulka B.3) Pravidlo pro okamžité ukončení záchytu. Stroj času ukončí a odstraní pravidla určité IP adresy ze všech filtrů. Dosud zachycená data jsou zpracována stejně, jako při automatickém ukončení záchytu. Cíl žádosti určuje směr IP adresy v paketech (0 - nespecifikovaný, 1 - zdroj, 2 - cíl). Zbytek žádosti obsahuje IP adresu.

Tabulka B.3: Struktura zprávy - Žádost o ukončení záchytu

Bajty	**0**	**1**	**2**	**3**
4-7	IP adresa			
8-11				
12-15				
16-19				

2. Seznam – Žádost o zaslání seznamu IP adres a jejich směru, které jsou ve filtrech stroje času. Cíl žádosti určuje směr IP adres (0 - nespecifikovaný, 1 - zdroj, 2 - cíl, 3 - vše).

3. Detail – Žádost o zaslání detailů o záchytu určité IP adresy. Cíl žádosti určuje směr IP adresy (0 - nespecifikovaný, 1 - zdroj, 2 - cíl). Zbytek žádosti obsahuje IP adresu. Struktura zprávy je stejná, jako žádost o ukončení záchytu (tabulka B.3).

4. Historie – Manipulace s komponentou záchyt historie. Cíl žádosti specifikuje konkrétní operaci:

   0. Zapnutí historie – (Tabulka B.4) Inicializace kruhových bufferů. Zbytek zprávy obsahuje celkovou velikost bufferů v MB.

Tabulka B.4: Struktura zprávy - Zapnutí bufferů s historií

Bajty	**0**	**1**	**2**	**3**
4-7	velikost bufferu			

1. Vypnutí historie – Odstranění kruhových bufferů.
2. Informace – Žádost o aktuální informace o bufferech.

5. Velikost začátku toku – (Tabulka 5) Natavení velikosti začátku toku. Cíl žádosti specifikuje 0 - fixní velikost začátku toku a dynamická délka záchytu, 1 - dynamická velikost začátku toku a fixní délka záchytu. Zbytek zprávy obsahuje fixní hodnotu.

Tabulka B.5: Struktura zprávy - Nastavení velikosti začátku toku

Bajty	**0**	**1**	**2**	**3**
4-7	velikost toku / délka záchytu			

6. Aktualizace – Žádost o automatické zasílání aktualizací ke stavu záchytu jednotlivých IP adres. Cíl žádosti specifikuje 0 - zapnutí zasílání informací, 1 - vypnutí zasílání informací.
7. Odpojení – Ukončení komunikace se strojem času.

B.0.2 Odpovědi

Odpovědi jsou zprávy od stroje času ke klientovi, které reagují na příchozí žádosti. Po přijetí žádosti je zkontrolována verze protokolu. V případě nekonzistentní verze je spojení okamžitě ukončeno. V opačném případě dojde k vykonání požadovaného typu žádosti, která je vždy potvrzena příslušnou odpovědí. Struktura hlavičky odpovědi, znázorněna v tabulce B.6, obsahuje v prvních dvou bajtech verzi protokolu a v dalších dvou bajtech typ odpovědi, který určuje tvar zbytku zaslané zprávy. Tabulka B.13 mapuje typy a cíle žádostí na typy odpovědí.

Tabulka B.6: Struktura zprávy - Hlavička odpovědi

Bajty	**0**	**1**	**2**	**3**
0-3	majoritní verze	minoritní verze	typ odpovědi	

Následující seznam popisuje všechny podporované typy odpovědí indexované jejich hodnotou.

0. Potvrzení – Potvrzení úspěšného vykonání žádosti.
1. Chyba – Přijatá žádost nebyla vykonána.

2. Neznámý příkaz – Přijatá žádost obsahuje neznámý typ nebo cíl. Po odeslání odpovědi je spojení okamžitě ukončeno, protože došlo k porušení definovaného protokolu.

3. Seznam – (Tabulka B.7) Seznam zachytávaných IP adres a jejich směru. Velikost zprávy je specifikována počtem zachytávaných adres daného směru, který je specifikován prvními 4 bajty. Následuje seznam položek, každá položka se skládá z IP adresy a jejího směru (17 Bajtů).

Tabulka B.7: Struktura zprávy - Seznam adres

Bajty	**0**	**1**	**2**	**3**
4-7	počet IP adres			
8-11	IP adresa			
12-15				
16-19				
20-23				
24-27	směr			

4. Detail – (Tabulka B.8) Aktuální stav záchytu určité IP adresy. Zpráva obsahuje IP adresu, směr adresy, stav prohledání bufferu s historií [%], počet zachycených paketů živého záchytu, limit paketů živého záchytu, aktuální dobu zachytávání [s], časový limit živého záchytu [s], délku jména záchytu a jméno záchytu dané délky. Jméno záchytu je specifikováno v žádosti o záchyt a jeho maximální délka je 255 znaků.

Tabulka B.8: Struktura zprávy - Souhrnné informace o záchytu

Bajty	**0**	**1**	**2**	**3**
4-7	IP adresa			
8-11				
12-15				
16-19				
20-23	směr	buffer - progres	počet zachycených paketů	
24-27			limit zachycených paketů	
28-31			délka záchytu	
32-35			časový limit záchytu	
36-...			délka jména	jméno

5. Buffer – (Tabulka B.9) Souhrnné informace o komponentě záchytu historie. Zpráva obsahuje počet instancí bufferu, aktuální počet uložených paketů, průměrnou délku historie všech bufferů [s], minimální délku historie [s], maximální délku historie [s], počet aktivních vyhledávání v bufferu, velikost začátku toku [pakety], celkovou velikost všech bufferů [B] a velikost volné paměti [B].

Tabulka B.9: Struktura zprávy - Informace o kruhových bufferech

Bajty	0	1	2	3
4-7	počet instancí	počet paketů		
8-15		průměrná délka historie		
16-19		minimální délka historie		
20-23		maximální délka historie		
24-27		aktivní vyhledávání		velikost začátku toku
28-31		velikost bufferů		
32-35				
36-39		volné místo v bufferech		
40-43				
44				

6. Aktualizace – Informace o aktuálním stavu záchytu. Tyto zprávy jsou zasílány, pokud klient poslal žádost o automatickou aktualizaci stavu záchytu. Stroj času periodicky zasílá změny stavu záchytu jednotlivých IP adres. Prvním políčkem je typ aktualizace, který určuje zbytek zprávy. Následující seznam popisuje typy aktualizace indexované jejich hodnotou.

   0. Nový záchyt – (Tabulka B.10) Přijetí nového pravidla pro záchyt. Zpráva obsahuje: ID záchytu, IP adresu a její směr. ID záchytu je v dalších zprávách používáno jako identifikátor záchytu.

Tabulka B.10: Struktura zprávy - Automatické aktualizace - Nový záchyt

Bajty	0	1	2	3
4-7	typ aktualizace	ID záchytu		
8-15		IP adresa		
16-19				
20-23				
24-27				
28-29		směr		

   1. Stav záchytu – (Tabulka B.11) Informace o stavu záchytu. Zpráva obsahuje: ID záchytu a stav záchytu [%], který je určen jako minimum ze stavu živého záchytu a záchytu historie.

Tabulka B.11: Struktura zprávy - Automatické aktualizace - stav

Bajty	0	1	2	3
4-7	typ aktualizace	ID záchytu		
8-15		stav		

2. Dokončení záchytu – (Tabulka B.12) Souhrnné informace o dokončeném záchytu. Zpráva obsahuje ID záchytu, počet paketů ze záchytu historie, počet paketů z živého záchytu, délky jmen souborů historického/živého záchytu a jména souborů. V případě, že živý záchyt nebo záchyt historie vytvoří více souborů, jsou jména jednotlivých souborů odděleny čárkou.

Tabulka B.12: Struktura zprávy - Automatické aktualizace - Výsledek

Bajty	**0**	**1**	**2**	**3**
4-7	typ aktualizace	ID záchytu		
8-15		počet zachycených paketů - historie		
16-19		počet zachycených paketů - živý záchyt		
20-23		délka jmen hist.		délka jmen živ.
24-...		jména souborů		

Tabulka B.13: Mapování žádostí na odpovědi

Typ žádosti	**Cíl žádosti**	**Typ Odpovědi**
0 - Záchyt	0 - Nespecifikováno 1 - Zdrojová IP 2 - Cílová IP	0 - Potvrzení 1 - Chyba
1 - Ukončení záchytu	0 - Nespecifikováno 1 - Zdrojová IP 2 - Cílová IP	0 - Potvrzení 1 - Chyba
2 - Seznam	0 - Nespecifikováno 1 - Zdrojová IP 2 - Cílová IP 3 - Vše	1 - Chyba 3 - Seznam
3 - Detail	0 - Nespecifikováno 1 - Zdrojová IP 2 - Cílová IP	1 - Chyba 4 - Detail
4 - Historie	0 - Zapnutí historie 1 - Vypnutí historie	0 - Potvrzení 1 - Chyba
	2 - Informace	1 - Chyba 5 - Buffer
5 - Velikost začátku toku	0 - Fixní velikost toku 1 - Fixní délka historie	0 - Potvrzení 1 - Chyba
6 - Aktualizace záchytu	0 - Zapnutí zasílání 1 - Vypnutí zasálání	0 - Potvrzení 1 - Chyba

Příloha C

Příkazy klientského programu

Při spuštění programu je zadána IP adresa monitorovací sondy a port, na kterém naslouchá stroj času. Po úspěšném připojení, je pomocí jednoduchého interpretru možné ovládat stroj času.

Následující stromová struktura představuje všechny příkazy pro ovládání klientského programu. Ty jsou vykonány zasláním příslušné žádosti a zobrazením přijaté odpovědi.

- ? - zobrazí nápovědu
- add - záchyt dat
  - src_ip <adresa> <jméno souboru> <limit paketů> <časový limit>
  - dsp_ip <adresa> <jméno souboru> <limit paketů> <časový limit>
  - bidir_ip <adresa> <jméno souboru> <limit paketů> <časový limit>
- remove - ukončení záchytu
  - src_ip <adresa>
  - dsp_ip <adresa>
  - bidir_ip <adresa>
- list - zobrazení seznamu aktivních záchytů
  - src_ip
  - dsp_ip
  - bidir_ip
- history_capture - manipulace s komponentou záchytu historie

  - turn_on <velikost> - inicializace bufferů
  - turn_off - vypnutí bufferů
  - info - zobrazení informací o bufferech
- heavy_size - nastavení velikosti toku
  - manual <počet paketů> - fixní velikost začátku toku
  - auto <délka historie>- fixní délka historie
- exit - odpojení a vypnutí aplikace

Příloha D

Nápovědy programů

D.1 time_machine_cli

```
Usage: time_machine-cli [options]

Options:
  -h, --help            Show this help message and exit.
  -i [ip address], --ip=[ip address]
                        IP of running time machine.
  -p [port], --port=[port]
                        Port of running time machine.
  -a [filename], --input_file_addresses=[filename]
                        File with IPs and ports of running time
                        machines.
  -t, --history_capture_info
                        Info about history buffer (this option
                        wil not open shell).
  -n [size in MB], --history_capture_turn_on=[size in MB]
                        Turn on the history capture (this option
                        wil not open shell).
  -f, --history_capture_turn_off
                        Turn off the history buffer (this option
                        wil not open shell).
  -s [number of packets], --history_capture_heavy_flow_manual=
                            [number of packets]
                        Change size of beginning of the flow
                        (number of packets of the flow) which
                        will be stored in history capture
                        (this option wil not open shell).
```

D.2 time_machine_manager

```
Usage: time_machine_manager [options]

Options:
  -h, --help            Show this help message and exit.
  -l [filename], --log=[filename]
                        Filename for log.
  -p [port], --port=[port]
                        Port of running server.
  -a [filename], --input_file_addresses=[filename]
                        File with IPs and ports of running
                        time machines.
```

D.3 nemea2time_machine

```
Name: Nemea2time_machine
Inputs: 1
Outputs: 0
Description:
  Socket wrapper for communication between Nemea module and
  time_machine_manager or time_machine. Wrapper sends UniRec
  records through TCP to time_machine or time_machine_manager.
  Mandatory field in UniRec input template is SRC_IP, the rest
  (SDM_CAPTURE_FILE_ID, TIMEOUT, PACKETS) depends on the given
  parameters.

Usage:  nemea2time_machine [COMMON]... [OPTIONS]...

Parameters of module [OPTIONS]:
```

-p --port <string>	Specify port.
-n --name <string>	Specify event name. The name will be extended by capturing IP address. If the parameter is set, field SDM_CAPTURE_FILE_ID is not required in UniRec input template.
-d --direction <number>	Specify direction of capturing. Source IP = 0, destination IP = 1, bidirectional IP (both directions) = 2. If the parameter is set, field DIRECTION is not required in UniRec

```
                                input template.

  -t  --timeout <number>        Specify timeout of the capturing
                                rule. If the parameter is set, field
                                TIMEOUT is not required in UniRec
                                input template.

  -a  --packets <number>        Specify number of packets that will
                                be maximally captured. If the
                                parameter is set, field PACKETS is
                                not required in UniRec input
                                template.

Common TRAP parameters [COMMON]:
  -h [trap,1]                   If no argument, print this message.
                                If "trap" or 1 is given, print TRAP
                                help.

  -i IFC_SPEC                   Specification of interface types and
                                their parameters, see "-h trap"
                                (mandatory parameter).

  -v                            Be verbose.

  -vv                           Be more verbose.

  -vvv                          Be even more verbose.

Environment variables that affects output:
  LIBTRAP_OUTPUT_FORMAT         If set to "json", information about
                                module is printed in JSON format.

  PAGER                         Show the help output in the set
                                PAGER.
```

D.3.1 UniRec položky modulu nemea2time_machine

```
UR_FIELDS (
   ipaddr SRC_IP,
   uint8 DIRECTION,
```

```
    uint32 PACKETS,
    uint32 TIMEOUT,
    string SDM_CAPTURE_FILE_ID
)
```

Příloha E

Instalační manuál

Instalační manuál byl sepsán a otestován na Scientific Linux 6.5. Podobnou instalaci je možné provést na některé jiné distribuci, na které je nainstalováno gcc a software Flowmonexp 4 [20].

```
$ tar -zxvf time_machine-1.0.tar.gz -C [cílové umístění]
$ cd [cílové umístění]
$ autoreconf -i
$ ./configure
$ make install
```

E.1 Spuštění zásuvného modulu aplikace Flowmonexp

Pro zjištění podporovaných modulů aplikace Flowmonexp lze zavolat příkaz:

```
$ flowmonexp -l
```

Dojde k vypsání všech nainstalovaných modulů včetně jejich popisu.

Spuštění Flowmonexp se strojem času se základním nastavením lze pomocí příkazu:

```
$ flowmonexp -P time_machine
```

E.2 Spuštění klientské aplikace

V rozbaleném adresáři se nachází aplikace time_machine-cli. Zavolání programu bez parametrů spustí klientskou aplikaci se základním nastavením. Pro spuštění je třeba Python 2.

```
$ python time_machine-cli
```

E.3 Spuštění Manažera distribuované správy

V rozbaleném adresáři se nachází aplikace time_machine_manager. Zavolání programu bez parametrů spustí klientskou aplikaci se základním nastavením. Pro spuštění je třeba Python 2.

```
$ python time_machine_manager
```

Příloha F

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `time_machine-1.0.tar.gz` ..... balík obsahující zdrojové kódy
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX