ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

Název:	Exportér síťových toků s podporou aplikačních informací
Student:	Jiří Havránek
Vedoucí:	Ing. Tomáš Čejka
Studijní program:	Informatika
Studijní obor:	Teoretická informatika
Katedra:	Katedra teoretické informatiky
Platnost zadání:	Do konce letního semestru 2017/18

Pokyny pro vypracování

Prostudujte současné technologie monitorování síťového provozu včetně rozšíření základních informací o aplikační protokoly.
Seznamte se s open-source systémem NEMEA [1,2] pro analýzu síťového provozu a detekci anomálií, soustřeďte se na existující softwarový exportér [3].
Proveďte analýzu použitých datových struktur a algoritmů pro ukládání a aktualizaci statistik v tzv. "Flow cache".
S ohledem na efektivní ukládání a vyhledání dat navrhněte rozšíření exportéru (tzn. především Flow cache) pro podporu aplikačních protokolů.
Jako ukázku použití nových datových struktur a algoritmů implementujte rozšíření např. pro ukládání HTTP nebo DNS informací.
Proveďte analýzu potřebných výpočetních zdrojů a pokuste se nalézt možnosti optimalizace exportéru.
Výsledný SW otestujte a změřte výkonnost rozšířeného exportéru pomocí dat, která dodá vedoucí práce.

Seznam odborné literatury

[1] Čejka, T., Bartoš, V., Švepeš, M., Rosa, Z., Kubátová, H.: NEMEA: A Framework for Network Traffic Analysis. In 12th International Conference on Network and Service Management (CNSM 2016), Montreal, Canada, 2016.
[2] https://github.com/CESNET/Nemea
[3] https://github.com/CESNET/Nemea-Modules/tree/master/flow_meter

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

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 31. ledna 2017

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

Bakalářská práce

Exportér síťových toků s podporou aplikačních informací

Jiří Havránek

Vedoucí práce: Ing. Tomáš Čejka

15. května 2017

Poděkování

Rád bych tímto poděkoval svému vedoucímu Ing. Tomášovi Čejkovi za trpělivost, odborné rady a připomínky při tvorbě bakalářské 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 15. května 2017 .....

Odkaz na tuto práci

Havránek, Jiří. *Exportér síťových toků s podporou aplikačních informací.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

Abstrakt

Monitorování síťového provozu je nezbytnou součástí správy dnešních počítačových sítí. Získané informace slouží nejen k zajištění základní funkcionality a včasné detekování potenciálních problémů, ale především k bezpečnostní analýze.

Z důvodu soukromí uživatelů a snížení objemu sbíraných dat je dnes standardem agregace provozu do tzv. síťových toků. Tato práce se zabývá otázkou exportu síťových toků s rozšířením o informace z aplikačních vrstev. Výsledkem práce je rozšíření a vylepšení open source exportéru toků z projektu NEMEA. Tento softwarový modul byl po optimalizacích původního kódu úspěšně portován na platformu vestavných zařízení se systémem OpenWrt. Díky tomuto přínosu je možné vytvořit monitorovací sondu i z nevýkonných levných domácích směrovačů a zvýšit tak přehled o provozu na síti včetně detekce škodlivého provozu.

Mimo paměťových a výkonnostních optimalizací vnitřních struktur je modul rozšířen o možnost číst pakety ze síťového rozhraní pomocí knihovny libpcap. Vnitřní struktura pro ukládání toků, flow cache, je upravena pro ukládání informací z aplikačních protokolů. Použití tohoto rozšíření je demonstrováno na dvou vytvořených ukázkových parsovacích pluginech pro HTTP a DNS provoz. Exportér je díky této práci schopen exportovat toky ve formátu IPFIX.

Klíčová slova exportér, síťový tok, aplikační protokol, optimalizace, systém NEMEA, OpenWrt, IPFIX, HTTP, DNS

Abstract

Network traffic monitoring is a necessary part of nowadays computer networks administration. Gathered information is not only used to provide basic network functionality and problem detection, but also for security analysis.

Due to user privacy and reduction of data volume, approaches based on network flows are used. This work focuses on exporting flow records with application protocol extension. Contribution of this work is a new version of existing open source flow exporter from NEMEA project. This software module was optimized and successfully ported to embedded devices with OpenWrt system. It is possible to create network monitoring probe from low performance cheap home routers and enhance awareness of traffic on network including malicious traffic detection.

Besides memory and performance optimizations, the module is extended of capability reading packets from network interface with the libpcap library. Flow cache, that is used to store flow records during the computation, was improved in order to handle application protocol information. The implemented version of the flow exporter contains two new example plugins for parsing HTTP and DNS protocols. In addition, the exporter is now able to export data in the IPFIX format.

Keywords exporter, network flow, application protocol, optimization, system NEMEA, OpenWrt, IPFIX, HTTP, DNS

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Monitorování síťového provozu je činnost, bez které se správa počítačových sítí v dnešní době neobejde. Zachycený provoz je nutné z různých důvodů zpracovávat a analyzovat. Jedná se například o zajištění základní funkcionality sítě a detekce potencionálních problémů spojených s konfigurací zařízení. Neméně důležité je odhalování škodlivého provozu na síti a předcházení tak nežádoucím situacím, ztrátám na majetku a dobrého jména. Z důvodu předcházení podobným incidentům není dobré brát bezpečnost na lehkou váhu.

Jeden ze způsobů monitorování síťového provozu je export paketů sloužící k paketové analýze. Jednalo by se o ideální řešení, protože paket v sobě obsahuje všechny informace. Bohužel je tento přístup velmi náročný po hardwarové stránce. Pakety je nutné celé ukládat a jejich zpracování vytěžuje procesor. Proto se často toto řešení nepoužívá a je využíváno jiných metod.

Druhý, více využívaný, způsob je monitorování síťových toků neboli flows. Síťový tok jsou agregované pakety pomocí pětice zdrojový port, cílový port, protokol transportní vrstvy, zdrojová a cílová IP adresa. Navíc obsahuje základní informace o provozu jako je čas začátku a konce komunikace, počet přenesených bajtů a paketů. K záznamům o tocích lze dále přidávat další informace nejčastěji z aplikační vrstvy ISO/OSI modelu. Na rozdíl od analýzy paketů je sběr síťových toků po hardwarové stránce jednodušší a mnohdy ani celé pakety nejsou při analýze provozu potřeba. Díky sledování metadat a ne obsahu komunikace, je navíc tento způsob monitorování pro uživatele bezpečnější z hlediska soukromí.

Existují síťové hrozby pro jejichž detekci základní informace o tocích nestačí. Příkladem mohou být skryté DNS tunely [1], útoky na SIP ústředny [2, 3], a detekce nakažených zařízení [4] na síti. Pro detekci takových hrozeb je potřeba mít k dispozici informace z aplikační vrstvy. Tyto informace poté poslouží k detailnější analýze při odhalování nebezpečné komunikace na síti.

Existuje mnoho nástrojů, které síťové toky exportují. Jen malá část z nich je open source a dokáže navíc exportovat aplikační rozšíření toků. Výstupem práce je nástroj, který bude popsanými vlastnostmi disponovat.

Kapitola 1

Cíl práce

Práce si klade za cíl vytvořit exportér rozšířených síťových toků pro sběr síťového provozu. Pro tento účel bude upraven existující exportní modul z projektu NEMEA[5]. Provoz na síti bude agregován do toků obohacených o nové informace z aplikačních protokolů HTTP a DNS. Výsledný nástroj bude open source a volně dostupný pro výzkumné, testovací a vzdělávací účely.

Exportér bude schopen získávat pakety ze síťové karty a také přečíst a zpracovat síťový provoz uložený v souboru na disku. Toky vytvořené z paketů exportér dokáže předat dalším systémům pro analýzu. Jednou z možností předání bude export na kolektor toků ve formátu IPFIX. Druhou bude odeslání toků přes speciální síťové rozhraní ostatním modulům v NEMEA systému.

Posledním požadavkem je zajistit, aby exportér mohl fungovat na různých platformách a na zařízeních se slabším hardwarem. To se týká zejména provozu na vestavných zařízeních s rozšířenou linuxovou distribucí OpenWrt. Exportér fungující na OpenWrt přinese možnost zachytávat síťový provoz například na snadno dostupných nevýkonných směrovačích, které používají běžní uživatelé doma.

Kapitola 2

Analýza

2.1 Síťový tok

Celá tato práce se pohybuje kolem technologie monitorování počítačových sítí pomocí síťových toků. Nejprve je potřeba popsat, co síťový tok vlastně je a co obsahuje jeho záznam. V této sekci budou popsány dva typy záznamů používané v této práci, základní a rozšířené.

Síťový tok reprezentuje jeden směr komunikace v síti v daném čase. Jedná se o sekvenci paketů, které mají stejnou zdrojovou a cílovou IP adresu, zdrojový a cílový port a protokol transportní vrstvy. Sekvenci lze identifikovat i pomocí dalších položek, než jen zmíněné pětice, například navíc pomocí 802.1 tagu, MPLS labelu atd. V této práci bude používaná výše popsaná pětice.

Pakety může exportér po určité době přestat zachytávat a tok ve formě záznamu exportovat. To se může stát v případě, že komunikace byla ukončena u TCP spojení nebo u UDP pokud po určité době nebyl zachycen žádný paket.

Základní záznam o síťovém toku, používaný v této práci, obsahuje mimo zmíněné pětice IP adres, portů a protokolu také základní informace o toku. Mezi ně patří čas začátku a konce toku, počet paketů a součet velikostí paketů v toku. Dalšími položkami jsou informace až do transportní vrstvy.

Základní záznam může být exportérem rozšířen o nové informace z aplikačních protokolů. Takový záznam bude v této práci označován jako rozšířený záznam o síťovém toku.

2.2 Čtení paketů

Základem každého exportéru je přijímat pakety ze sítě. Řešení tohoto problému určuje, jak moc bude exportér výkonný a použitelný na rychlých sítích. Efektivnost získání paketů má spolu s volbou vhodných datových struktur a algoritmů největší vliv na výkonnost. Tato sekce se bude zabývat analýzou existujících způsobů, knihoven a frameworků pro čtení paketů ze síťových karet.

AF_PACKET [6] je síťový Unix soket umožňující odesílat a přijímat pakety na úrovni ovladače zařízení. Poskytuje možnost implementovat v user space aplikaci vlastní protokoly nad fyzickou vrstvou. Typ soketu je SOCK_RAW pro pakety obsahující hlavičku linkové vrstvy nebo SOCK_DGRAM bez hlavičky. Tento způsob čtení paketů je poměrně neefektivní vzhledem k množství kopírování mezi síťovou kartou a aplikací.

Možného zrychlení lze docílit pomocí asynchronního čtení paketů díky kruhovému bufferu od linux kernelu verze 2.6.2x. Nastavením volby SOL_PACKET na PACKET_RX_RING u soketu pomocí volání funkce `setsockopt()`, kernel alokuje kruhový buffer, do kterého jsou poté ukládány pakety nezávisle na aplikaci (aniž by se volala funkce `read()`). Aplikace si dále namapuje buffer do svého virtuálního adresního prostoru pomocí funkce `mmap()`. Čtení paketu probíhá nejprve kontrolou příznaku TP_STATUS_USER v hlavičce položky v ring bufferu. Pokud obsahuje zmíněný příznak, paket je k dispozici, jinak se čeká na změnu příznaku voláním `poll()` dokud se paket v bufferu neobjeví. Více informací lze dohledat v [7].

libpcap [8] je známá multiplatformní open source knihovna pro nízkoúrovňový záchyt provozu. Je standardem a využívá ji celá řada komerčních i open source projektů například tcpdump, wireshark, iftop. Knihovna poskytuje API pro čtení paketů nejen ze síťového rozhraní, ale i ze souborů ve formátů pcap a pcapng. Pakety jsou ze síťového rozhraní získávány pomocí síťového soketu AF_PACKET v kombinaci s dostupnými zrychleními popsanými výše. Výhodou je snadné nasazení, portabilita a podpora síťových zařízení. Nevýhodou je naopak pomalý záchyt, kdy libpcap nemusí v porovnání s ostatními řešení na rychlejších sítích stačit.

PF_RING vanilla [9] je framework, který slouží pro vysokorychlostní čtení a odesílání paketů. Skládá se z modulu, který se nahraje do jádra operačního systému (od linux kernelu verze 2.6.32 výše), a knihovny libpfring. Kernel modul obsahuje kruhový buffer, do kterého se ukládají zachycené pakety ze síťové karty. Pakety jsou poté zkopírovány knihovnou do aplikace, kde jsou k dispozici ke zpracování. Dále umožňuje provádět load balancing mezi několika procesorovými vlákny při čtení paketů z jednoho síťového rozhraní. Záchyt lze bez problému provádět na 10 Gb/s linkách. Tato verze PF_RING je distribuovaná pod GNU GPLv2 licencí a lze ji využívat spolu s knihovnou libpcap.

PF_RING Zero Copy [10] je vylepšenou verzí předchozího řešení umožnující rychlejší získání paketu díky eliminaci kopírování dat mezi kruhovým bufferem a knihovnou. Zrychlení je docíleno pomocí speciálně upravených ovladačů síťových karet. Paket je ze síťové karty uložen přímo do vnitřního

bufferu knihovny libpfring a je k dispozici aplikaci bez užití systémových volání. Řešení lze využívat na 40 Gb/s linkách a nevýhodou je, že ne všechny síťové karty jsou podporovány. Užívání tohoto řešení vyžaduje zpoplatněnou licenci.

netmap [11] je framework určený pro vysokorychlostní čtení a odesílání paketů podobný jako PF_RING. Také se skládá z jednoho modulu určeného pro jádro operačního systému a user space knihovny. Přidává Virtual Local Ethernet (VALE) pro rychlou komunikaci s virtuálními stroji nebo mezi procesy a netmap pipes, což jsou kanály pro přenos dat vytvořené pomocí regionu sdílené paměti. Co se týká výkonu, je rychlejší než standardní systémové mechanismy, například sokety, a dosahuje rychlosti až 14,88 milionu paketů za sekundu saturováním celé 10 Gb/s linky s 1 procesorovým jádrem. Na 40 Gb/s linkách až 30 Mpps [12]. Dodáván je jako součást standardní FreeBSD distribuce a zdrojové kódy jsou volně k dispozici. Lze využít s knihovnou libpcap.

Intel DPDK [13] Data Plane Development Kit je set knihoven a optimalizovaných ovladačů síťových zařízení určených pro rychlé zpracování paketů. DPDK poskytuje framework systémům s Intel x86, ARM a IBM Power procesory pro snadný vývoj vysokorychlostních síťových aplikací. K rychlému zpracování paketů je docíleno pomocí více vláken, kruhových bufferů, prealokací bufferů, eliminace přerušení procesoru pomocí poll mode ovladačů (PMD) a používáním stránek s větší velikostí ve virtuální paměti. Řešení je poskytováno a vyvíjeno pod open source BSD licencí.

Sniffer 10G [14] je řešení od firmy Myricom určené pro jimi vyvíjený vlastní hardware. Stejně jako ostatní řešení, i Sniffer 10G používá kruhový buffer, modifikované ovladače a snižuje počet paměťových operací nutných k získání paketu ze síťové karty. K užívání je potřeba zpoplatněná licence.

2.3 Formáty pro export toků

Pro zajištění kompatibility mezi různými systémy existuje několik standardizovaných protokolů pro ukládání a odesílání záznamů o síťových tocích. Každý protokol definuje, v jakém formát jsou záznamy ukládány, co obsahuje datový paket a jak probíhá komunikace mezi exportérem a kolektorem. Tato sekce se věnuje průzkumu dostupných formátů pro ukládání síťových toků, které budou v této práci zmíněny. Popis komunikace a podoba paketů je dostatečně vysvětlena v citovaných dokumentech.

NetFlow v5 [15] je vytvořen společností Cisco pro záchyt síťových toků pomocí Cisco směrovačů a přepínačů. V odesílaných zprávách podporuje

pouze pevně danou množinu exportovaných políček. Z toho vyplývá, že s tímto formátem nelze exportovat rozšířené síťové toky. Jedná se o nejpoužívanější verzi NetFlow.

NetFlow v7 přidává některá informační políčka navíc oproti verzi 5.

NetFlow v9 [16] je vylepšená verze NetFlow formátu, která přináší odstranění nevýhod předchozích verzí zavedením šablon. Díky šablonám není export omezen fixním počtem informačních políček ve zprávě. Exportní sondy si mohou dle potřeby definovat několik šablon s libovolnými políčky a záznamy poté exportovat na kolektor.

IPFIX [17] neboli Internet Protocol Flow Information Export je standardizovaný protokol pro export síťových toků. Někdy také označován jako NetFlow v10. Přináší možnost tvorby šablon pro flexibilní definování políček k exportu. Tento protokol podporuje export rozšířených síťových toků o informace z aplikačních vrstev. V jedné zprávě je možné exportovat více šablon.

UniRec [18] je formát pro efektivní přenos dat z projektu NEMEA. Umožňuje definovat libovolný počet vlastních políček pomocí šablony. V posílaných zprávách je podporován export pouze jedné šablony v rámci jednoho spojení.

2.4 Existující exportéry

Na světě existuje velké množství exportérů síťových toků. Jednat se může buď o fyzické zařízení, speciální hardware určený k zachycování paketů, nebo o software běžící na stroji spolu s jinými službami. V této sekci bude věnováno pár slov představení existujících řešení.

Flowmon Probe [19] je výkonný síťový exportér od Flowmon Networks podporující export informací z aplikační vrstvy. Jedná se o komerční řešení, které je dodáváno jako specializovaný samostatný hardware a nebo v podobě virtualizovaného nástroje. Zvládá sítě při rychlostech 10 Mb/s až 100 Gb/s, záleží na použitém modelu. Nevýhodou je nutnost používat placenou licenci.

nProbe [20] je řešení vyvíjené společností ntop. K provozu je zapotřebí licence pokud se nejedná o neziskovou organizaci nebo univerzitu. Dokáže exportovat rozšířené síťové toky ve formátu NetFlow nebo IPFIX. Lze pořídit jako samostatné softwarové řešení s výkonem 1 Gb/s a nebo v podobě vestavného zařízení, které dokáže provádět záchyt i nad 100 Gb/s sítí. K záchytu provozu je použita knihovna libpcap nebo mechanismus PF_RING, případně PF_RING Zero Copy.

Cisco zařízení [21] umožňuje exportovat síťové toky ve formátu NetFlow přímo z Cisco směrovače nebo přepínače. Jedná se o placené řešení, jelikož je potřeba vlastnit jak zařízení, tak i licenci pro provoz operačního systému IOS.

fprobe [22] je open source vyvíjená sonda umožňující export v NetFlow formátu. Nevýhodou je nemožnost zpracovávat aplikační vrstvy. Síťový provoz je získáván pomocí knihovny libpcap.

softflowd [23] open source exportér nabízející stejné funkce jako fprobe.

nfcapd [24] neboli NetFlow capture daemon je exportér NetFlow v5, v7 a v9 z NFDUMP nástrojů. Toky ukládá do souboru ve formátu nfdump a neumožňuje zpracovávat aplikační vrstvy. NFDUMP nástroje jsou součástí projektu NfSen a distribuovány jsou pod BSD licencí.

flow_meter [25] je exportér napsaný v programovacím jazyce C++ z open source projektu NEMEA, jehož rozšíření je předmětem této práce. Lze s ním číst pakety uložené v pcap souboru, neumí získávat provoz ze síťové karty. Základní síťové toky exportuje pouze v interním UniRec formátu projektu.

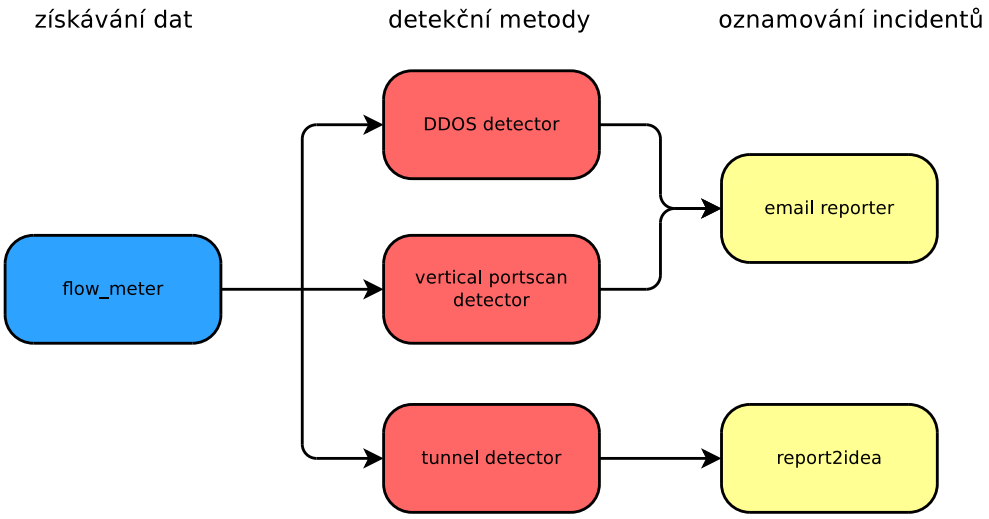
2.5 Projekt NEMEA

Celá práce se bude pohybovat okolo existujícího exportéru flow_meter z projektu NEMEA. Jedná se o open source modulární systém pro proudovou detekci a analýzu síťového provozu. Celý systém se skládá z jednotlivých nezávislých komponent, modulů, které lze mezi sebou propojovat přes komunikační rozhraní. Každý modul má svůj specifický úkol, například se může jednat o:

- Získávání a manipulace s daty, preprocessing, ukládání nebo čtení v různých datových formátech
- Detekce síťových hrozeb a škodlivého provozu
- Oznamování incidentů správcům systému

Toto řešení umožňuje škálovatelnost a velmi snadné přidání nových funkcí skrze moduly do systému. Jednoduchý příklad zapojení modulů znázorňuje obrázek 2.1.

Komunikaci mezi jednotlivými moduly a různé implementace algoritmů zajišťuje NEMEA Framework, což je sada knihoven poskytující modulům společné API. Framework obsahuje knihovnu libtrap (TRAP neboli Traffic Analysis Platform), která implementuje komunikační rozhraní a funkce pro



Obrázek 2.1: Příklad zapojení modulů v systému NEMEA. Šipky znázorňují tok dat.

odesílání a přijímání zpráv mezi nimi. Posílané zprávy jsou ukládány v efektivním datovém formátu UniRec (Unified Record). Ostatní funkce jako například implementace efektivních datových struktur a algoritmů jsou obsaženy v knihovně Common.

Moduly mohou pro komunikaci využívat několik vstupních a výstupních rozhraní. Knihovna libtrap poskytuje možnost vybrat u každého rozhraní i typ. Dostupné typy rozhraní jsou:

Unix domain socket pro komunikaci mezi běžícími procesy. Neumožňuje zprostředkovat komunikaci mezi moduly běžícími na jiných fyzických zařízení.

TCP rozhraní pro komunikaci pomocí TCP soketu. Na rozdíl od Unix soketu lze komunikovat mezi moduly na různých fyzických strojích.

File interface umožňuje ukládat a číst zprávy ze souborů uložených na disku.

2.5.1 UniRec

UniRec je formát pro efektivní přenos zpráv mezi moduly. Obsah zprávy je definován pomocí šablony. Ta se skládá z informačních políček různých typů, které popisuje tabulka 2.1. Na začátku zprávy jsou offsety políček, za nimi se nachází políčka fixní délky a nakonec následují ty s variabilní délkou. Offsety umožňují velmi rychlý přístup k datům uloženým ve zprávě. Důležité je zmínit, že v rámci jednoho spojení je možné používat pouze jednu šablonu.

Tabulka 2.1: Typy políček, které se mohou vyskytovat v UniRec šabloně. Tabulka pochází z [26].

Název typu	Počet bajtů	Popis typu
int8	1	8 bit integer se znaménkem
int16	2	16 bit integer se znaménkem
int32	4	32 bit integer se znaménkem
int64	8	64 bit integer se znaménkem
uint8	1	8 bit integer bez znaménka
uint16	2	16 bit integer bez znaménka
uint32	4	32 bit integer bez znaménka
uint64	8	64 bit integer bez znaménka
char	1	jeden ASCII znak
float	4	číslo s plovoucí řádovou čárkou (IEEE 754)
double	8	číslo s plovoucí řádovou čárkou (IEEE 754)
ipaddr	16	typ obsahující IPv4 nebo IPv6 adresu
time	8	typ obsahující časovou značku
string	-	ASCII string
bytes	-	pole bajtů

2.6 flow_meter

Původně se jednalo o modul určený pro experimenty s různými typy flow cache. Umožňuje exportovat základní záznamy o tocích z paketů přečtených ze souboru ve formátu pcap. K exportu záznamů využívá jedno výstupní TRAP rozhraní. Popis exportovaných políček na tomto rozhraní je obsažen v tabulce 2.2.

Modul získává pakety z pcap souboru vlastním způsobem, bez použití knihovny. V paket parseru je podporováno zpracování následujících protokolů:

- Ethernet II
- 802.1Q vlan
- IPv4
- IPv6
- TCP
- UDP

Strukturu celého exportéru popisuje obrázek 2.2. Na obrázku jsou zobrazeny C++ třídy a jejich propojení. Třída `PcapReader` se stará o získávání paketů ze souboru. Po zparsování je paket vložen do třídy `FlowCache`, která vytvoří záznam o toku nebo aktualizuje stávající. `FlowCache` obsahuje plugin `StatsPlugin` pro vytváření statistik. Toky určené k exportu jsou předány

Tabulka 2.2: Základní informační políčka záznamu o toku exportované modulem flow_meter.

Název UniRec políčka	Datový typ	Popis políčka
DST_IP	ipaddr	cílová IP adresa
SRC_IP	ipaddr	zdrojová IP adresa
BYTES	uint64	počet bajtů v toku
LINK_BIT_FIELD	uint64	identifikace exportéru
TIME_FIRST	time	čas začátku toku
TIME_LAST	time	čas konce toku
PACKETS	uint32	počet paketů v toku
DST_PORT	uint16	cílový port
SRC_PORT	uint16	zdrojový port
DIR_BIT_FIELD	uint8	směr toku (ingress / egress)
PROTOCOL	uint8	protokol transportní vrstvy
TCP_FLAGS	uint8	TCP příznaky
TOS	uint8	IP typ služby
TTL	uint8	IP TTL

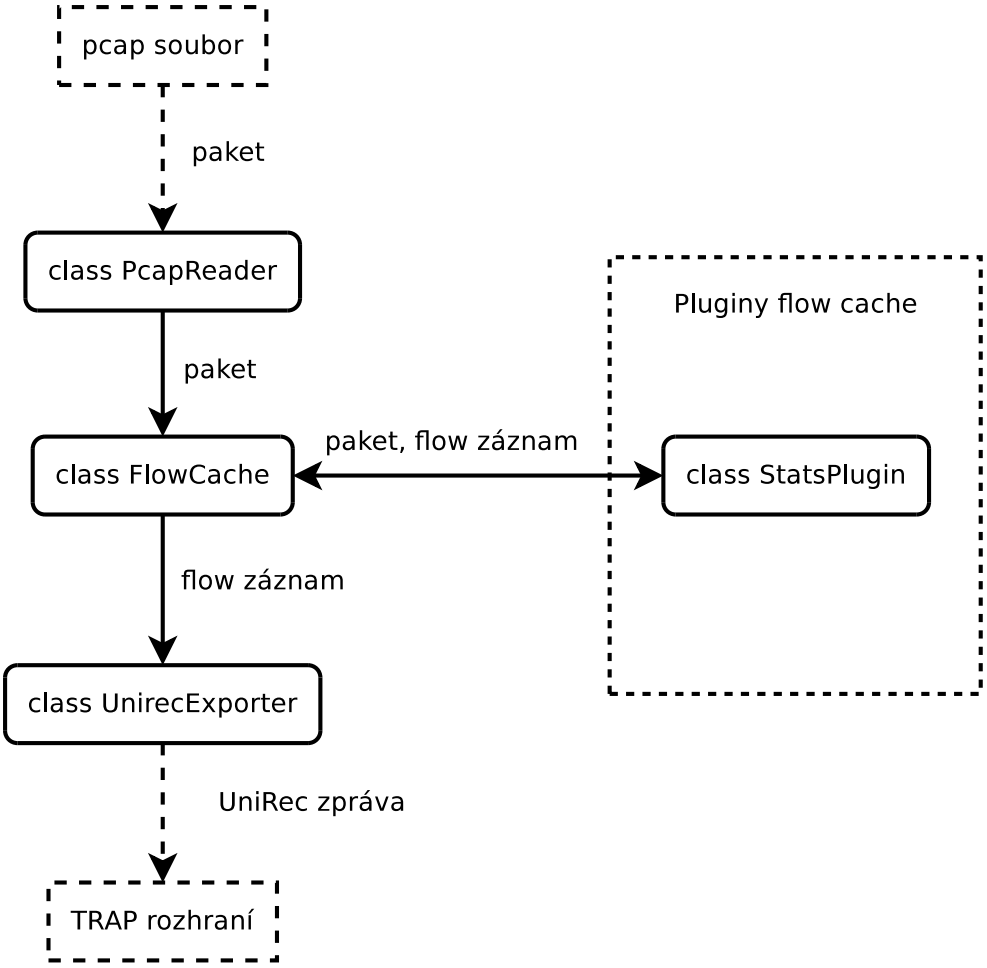
na vstup třídě `UnirecExporter`, která je odešle na výstupní TRAP rozhraní. Pluginy a třída `FlowCache` jsou popsány v následující podsekci.

2.6.1 Flow cache

Modul flow_meter z projektu NEMEA obsahuje tzv. flow cache. Flow cache je datová struktura umožňující snadno vyhledávat a ukládat záznamy o tocích. Jedná se o hash tabulku s pevným počtem řádků. Řádek obsahuje pevný počet ukazatelů na záznamy a tímto způsobem jsou řešeny kolize. Pro urychlení vyhledávání jsou ukazatele na záznamy o tocích v jednotlivých řádcích řazeny podle doby posledního použití. V paměti jsou jednotlivé řádky umístěny za sebou, jedná se v podstatě o pole. Velikost flow cache lze nastavit parametrem při inicializaci struktury. Ukázková flow cache na obrázku 2.3 popisuje celou strukturu.

Vyhledání řádku probíhá tak, že z pětice *(zdrojový port, cílový port, protokol, zdrojová IP adresa, cílová IP adresa)* identifikující tok je vytvořen klíč a ten je vložen do hash funkce collate [27] z C++ standardní knihovny. Z výstupu hash funkce je poté vypočten index příslušného řádku cache. Jakmile je znám index řádku, vyhledání příslušného záznamu o toku probíhá sekvenčně od začátku řádku. Porovnávají se hodnoty hashe a zároveň hodnoty klíčů na shodu.

Následující C++ kód popisuje, jak se v cache počítá pozice řádku. Do funkce hash() je vložen pole `key` obsahující za sebou IP adresy, porty a protokol délky `keyLength`. Z hodnoty hash funkce je vypočtena hodnota pozice řádku `lineIndex` ve flow cache.

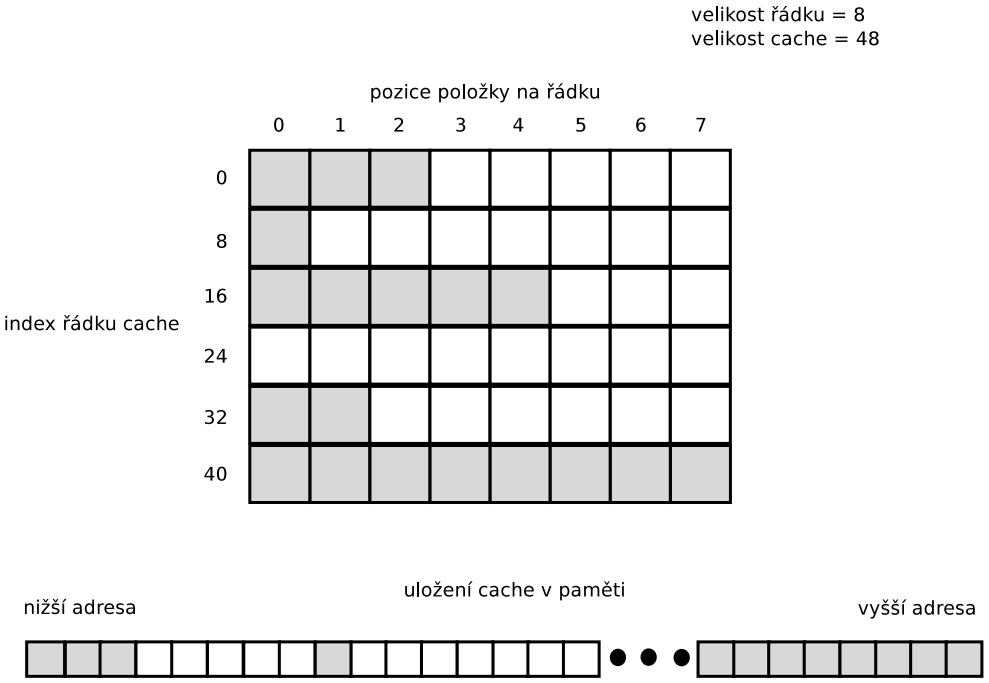


Obrázek 2.2: Vnitřní struktura exportéru

```
uint64_t hashValue = coll.hash(key, key + keyLength);

int tmp = hashValue % cacheSize;
int lineIndex = (tmp / lineSize) * lineSize;
```

Aktualizace nebo vkládání nového toku do flow cache začíná hledáním existující položky. U existujícího nalezeného záznamu jsou upraveny čítače bajtů, paketů, čas a TCP příznaky. Pokud záznam neexistuje, je nově vytvořen a vložen do řádku. V případě, že daný řádek je plný a není možné uložit další záznam, se strategie nahrazování řídí algoritmem LRU. Nejméně používaná položka je odstraněna z cache a nahrazena novou. Nově přidaný a nebo aktualizovaný záznam je poté umístěn na začátek řádku.

Výchozí velikost flow cache je 65536 položek. Položkou je datová struktura `Flow`, která má velikost 240 bajtů. Samotný záznam o toku popisuje datová



Obrázek 2.3: Ukázková flow cache obsahující 48 položek, 6 řádků a 8 položek na řádek.

struktura `FlowRecord`, která v paměti zabírá 112 bajtů.

```
struct FlowRecord {
    uint64_t flowFieldIndicator;
    double   flowStartTimestamp;
    double   flowEndTimestamp;
    uint8_t  ipVersion;
    uint8_t  protocolIdentifier;
    uint8_t  ipClassOfService;
    uint8_t  ipTtl;
    uint32_t sourceIPv4Address;
    uint32_t destinationIPv4Address;
    char     sourceIPv6Address[16];
    char     destinationIPv6Address[16];
    uint16_t sourceTransportPort;
    uint16_t destinationTransportPort;
    uint32_t packetTotalCount;
    uint64_t octetTotalLength;
    uint8_t  tcpControlBits;
    uint64_t flowPayloadStart;
    uint64_t flowPayloadSize;
```

```
};

class Flow
{
    uint64_t hash;
    uint64_t plimit;
    double inactive;
    double active;
    char key[76];
    char *payload;
public:
    bool empty_flow;
    FlowRecord flowrecord;
}
```

Cache používá aktivní a neaktivní časový limit (timeout) pro kontrolu expirovaných záznamů. Kontrola aktivního času se uplatňuje, pokud jsou pakety daného toku stále zachycovány exportérem. Kontrola neaktivního času se uplatňuje, pokud pakety k příslušnému toku se již nevyskytují na síti (komunikace byla ukončena). Pokud jeden z časových limitů vyprší, záznam je exportován a odstraněn z flow cache. Výchozí hodnota aktivního limitu je 300 sekund a neaktivního 30 sekund.

Flow cache obsahuje neefektivní způsob, jak kontrolovat popsané časové limity. Do proměnné se ukládá časová značka posledního vloženého paketu. Při vložení paketu do cache se kontroluje, jestli je rozdíl časových značek nově vkládaného a posledního vloženého paketu větší než konstanta. Pokud ano, celá cache se sekvenčně prochází a kontrolují se aktivní a neaktivní časové limity.

Cache obsahuje API pro vytváření a používání pluginů. Naimplementován je zde jeden plugin pro získávání statistik ohledně flow cache. API pak poskytuje několik callback funkcí umožňující modifikovat záznamy o tocích. Funkce se pro každý aktivní plugin volají v příslušných okamžicích:

init() se volá na začátku při inicializaci flow cache.

post_create() se volá po vytvoření záznam o toku. Vstupem funkce je paket a nově vytvořený záznam.

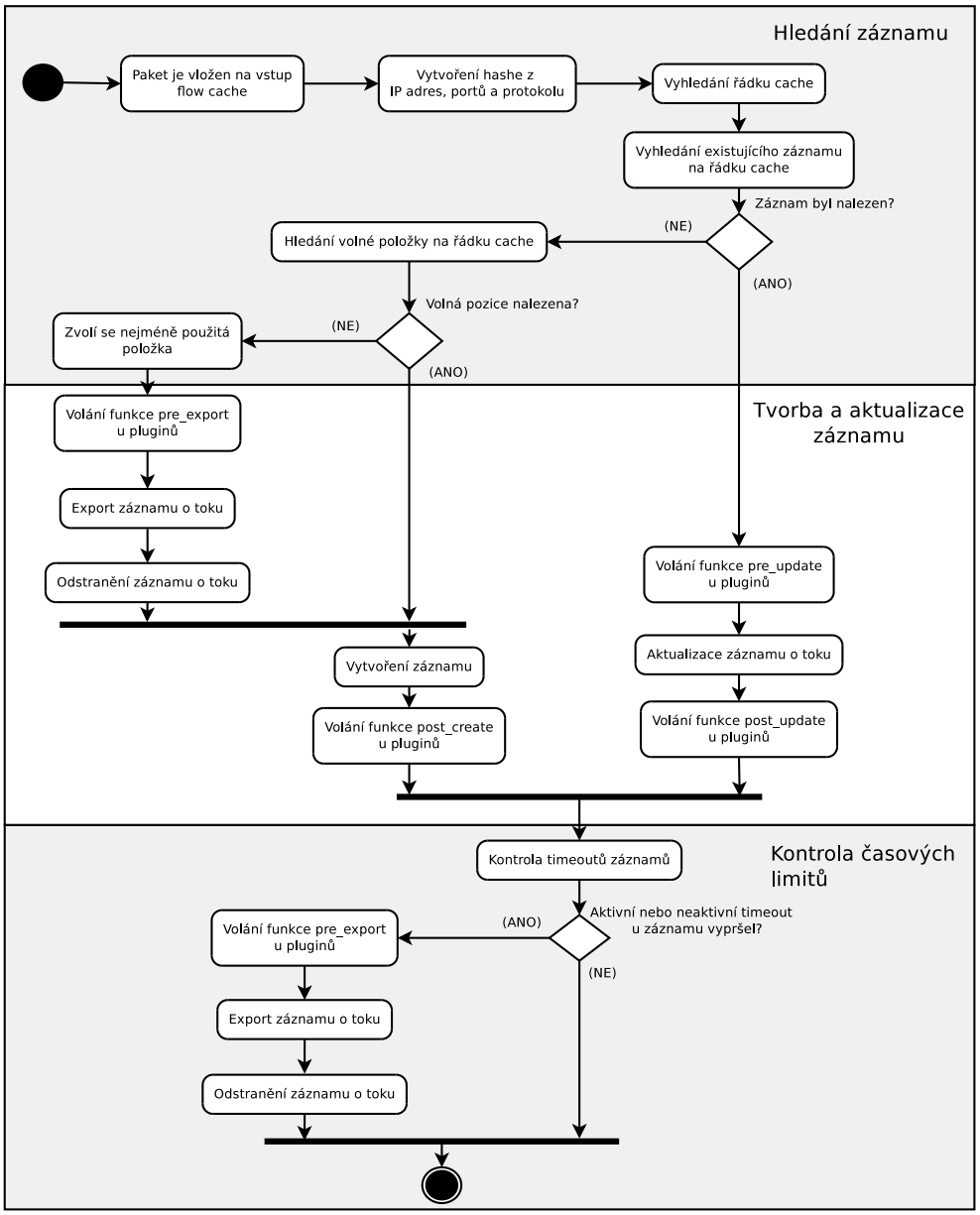
pre_update() se volá před aktualizací existujícího záznamu o toku. Vstupem funkce je aktualizovaný záznam.

post_update() se volá po aktualizaci existujícího záznamu o toku. Vstupem funkce je aktualizovaný záznam.

pre_export() se volá před exportem záznamu o toku, před odstraněním z flow cache. Vstupem je záznam o toku.

finish() se volá při ukončení modulu.

Celý proces vkládání paketu do flow cache popisuje diagram 2.4. V obrázku je znázorněno, kdy se volají výše uvedené funkce.



Obrázek 2.4: Diagram aktivit flow cache

2.7 HTTP

HTTP neboli Hyper Text Transfer Protocol [28] je textově orientovaný aplikační síťový protokol pro přenos dat. Byl navržen a používá se jako prostředek komunikace mezi klientem a webovým serverem. Dnes je jedním z nejpoužívanějších protokolů na internetu. Typické číslo portu pro komunikaci je TCP 80 a občas se můžeme setkat i s alternativou TCP 8080.

Komunikace mezi klientem a serverem probíhá na principu požadavku a odpovědi. Klient odešle zprávu, ve které specifikuje požadovanou akci, a server reaguje odesláním odpovědi. Požadované akce se v HTTP terminologii nazývají metody a mezi nejčastější patří tyto:

GET je požadavek na obsah identifikovaný pomocí URI v HTTP zprávě.

POST metodou klient žádá, aby server přijal data v požadavku na místo identifikovaným pomocí URI.

Server na požadavky od klienta odpovídá provedením žádané akce a odesláním příslušné odpovědi dle typu dotazu.

Jak již bylo zmíněno, jedná se o textově orientovaný protokol. Veškeré zprávy jsou kódovány v ASCII kódu. Dotaz od klienta se skládá z následujících částí v daném pořadí:

- Řádek s dotazem ve formátu *Metoda URI HTTP-verze*.
- Řádky s informačními hlavičkami ve formátu *Klíč: hodnota*.
- Prázdný řádek.
- Nepovinná část s obsahem.

Standard definuje, že každý řádek je oddělen sekvencí znaků $\backslash r \backslash n$. Odpověď od serveru na požadavek se poté skládá ze stejných uvedených částí s rozdílem, že první řádek je ve formátu *HTTP-verze návratový-kód zpráva*. Poslední prvek *zpráva* obsahuje textovou reprezentaci informace o provedené akci.

Příklad komunikace od klienta je následující:

```
GET / HTTP/1.1
User-Agent: Wget/1.18 (linux-gnu)
Accept: */*
Accept-Encoding: identity
Host: www.test.com
Connection: Keep-Alive
```

Odpověď od serveru:

```
HTTP/1.1 200 OK
Server: Apache
```

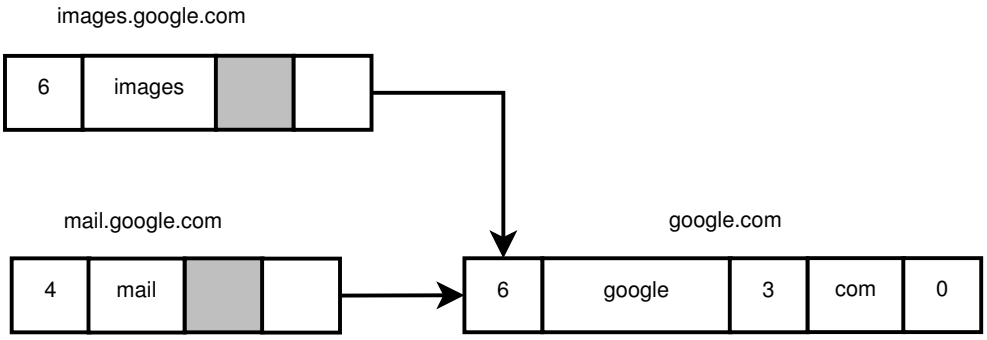
```
Content-Type: text/html
Expires: Fri, 21 Apr 2017 16:28:18
Content-Language: en
Content-Length: 327
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true

<!DOCTYPE html>
<html lang="en">
<head>
<meta http-equiv="content-type"
    content="text/html; charset=utf-8">
<title>Test</title>
</head>
<body>
<p>This is test page</p>
</body>
</html>
```

Nevýhodou protokolu HTTP je jeho bezpečnost. Kdokoliv s přístupem ke komunikačnímu médiu může snadno zachytit a zneužít informace z nezabezpečené komunikace mezi klientem a serverem. Tyto nedostatky poté řeší HTTPS protokol, poskytující vyšší míru bezpečnosti díky šifrování. HTTP provoz je vložen do zašifrovaného tunelu vytvořeného pomocí protokolu SSL nebo TLS. Standardní port pro zabezpečenou HTTPS komunikaci je TCP 443.

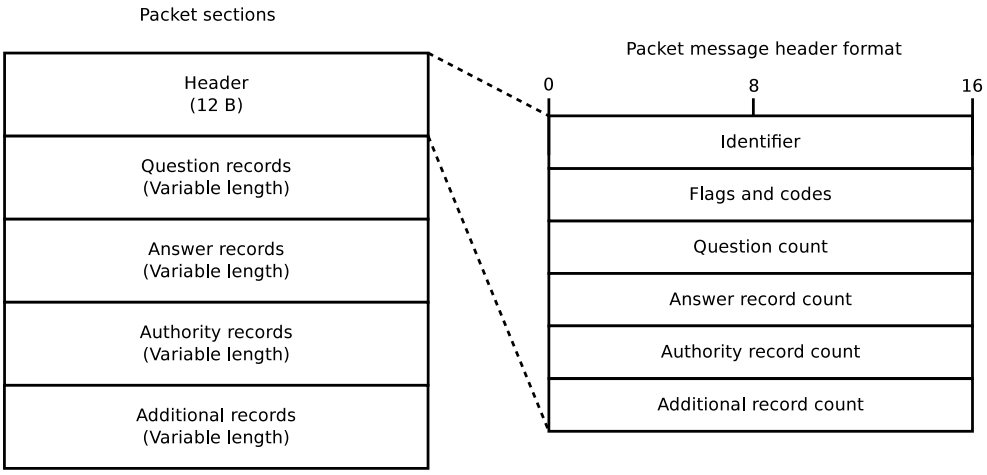
2.8 DNS

Překlad doménových jmen, o které se stará DNS [29] (Domain Name System), je dnes již nedílnou součástí internetu. DNS je hierarchický systém doménových jmen jehož hlavním úkolem je převod mezi doménovými jmény a IP adresami. Ke komunikaci mezi klientem a DNS serverem se využívá stejnojmenný protokol aplikační vrstvy. Spolu s HTTP protokolem tvoří velkou část objemu komunikace na internetu. Komunikace probíhá standardně přes UDP nebo TCP port 53.

Doménové jméno je hierarchicky uspořádáno a obsahuje názvy jednotlivých domén oddělené tečkami. Pro uložení doménového jména v paketu se používá komprese. Opakující se suffix doménových jmen je v paketu uložen pouze jednou. To umožňuje snížit nároky na velikost paketu. V praxi to vypadá tak, že v paketu se před každou část domény (label) vloží buď jeho délka a nebo odkaz. Odkaz obsahuje offset a říká, kde je uložen zbytek doménového jména v paketu. Celou myšlenku ukazuje obrázek 2.5.



Obrázek 2.5: Příklad komprese doménových jmen. Šedé políčko značí příznak, že místo délky labelu následuje odkaz na část doménového jména.



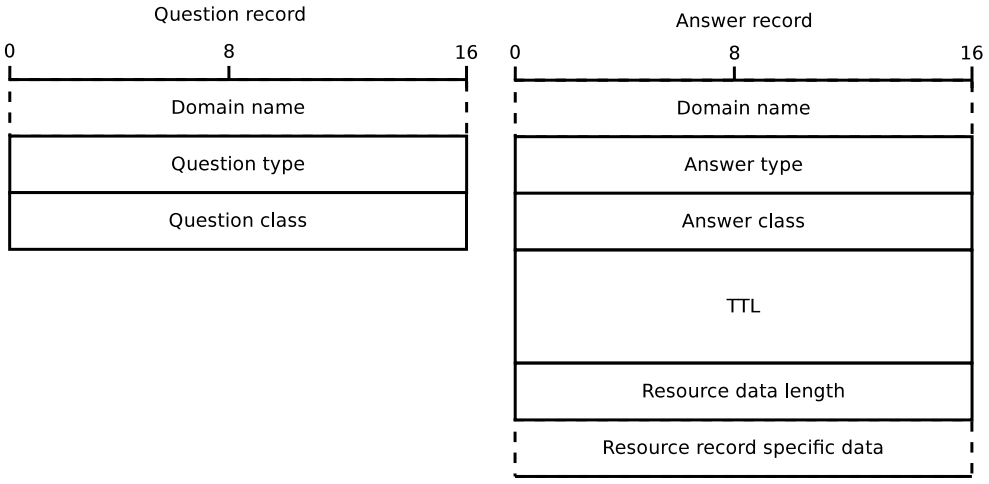
Obrázek 2.6: Struktura DNS paketu. Paket může obsahovat více dotazů a odpovědí, ty jsou v příslušných sekcích řazeny za sebou. Jejich počet je uveden v hlavičce.

Komunikace mezi klientem a DNS serverem probíhá na stejném principu jako u HTTP protokolu. Pro větší efektivitu překladu a ušetření šířky pásma může paket s dotazem obsahovat více požadavku na překlad. DNS server odpovídá paketem s přeloženými záznamy. Struktura paketu je popsána v obrázku 2.6.

Základní dva formáty záznamů, question a answer, popisuje obrázek 2.7. Nejčastější typy záznamů jsou následující:

A je záznam obsahující IPv4 adresu pro dané doménové jméno.

AAAA je záznam obsahující IPv6 adresu pro dané doménové jméno.



Obrázek 2.7: Formát záznamů vyskytujících se v DNS paketu. Záznam s odpovědí obsahuje v poslední položce specifická data dle typu záznamu.

CNAME obsahuje alias pro doménu - jiné doménové jméno k již zavedené doméně.

PTR obsahuje reverzní překlad IP adres na doménové jméno.

MX obsahuje doménová jména dostupných serverů pro elektronickou poštu a jejich priority v dané doméně.

NS obsahuje jméno autoritativního serveru pro danou doménu.

Pro detailnější rozbor formátů a typů záznamů zde není prostor. K orientaci v dané problematice by tato analýza měla stačit, nicméně vše lze dohledat ve specifikaci protokolu.

2.9 OpenWrt

OpenWrt [30] je minimalistická linuxová distribuce pro vestavné zařízení, například běžně dostupné směrovače. Hlavní komponenty tvoří Linux, util-Linux, knihovna uClib a BusyBox. Tyto komponenty jsou optimalizovány vzhledem k velmi malému množství paměti a procesorového výkonu, který je v těchto směrovačích k dispozici.

Distribuce slouží jako náhrada firmwaru, které používají výrobci se svých produktech. Firmware od výrobce často umožňuje pouze omezenou možnost ovládat a konfigurovat zařízení. Horší je, že mnohdy výrobce ani ke svým zařízením nedodává aktualizace. OpenWrt poskytuje plnou kontrolu nad zařízením, umožňuje instalovat vlastní software, aktualizace. Uživatelé mohou

mít ze směrovačů například domácí NAS, HTTP, DNS nebo multimediální server.

V této práci bude na OpenWrt nahlíženo jako na možnost, jak si ze svého domácího směrovače vyrobit monitorovací sondu pomocí exportéru toků.

Kapitola 3

Návrh rozšíření exportéru

3.1 Čtení paketů

Původní verze modulu flow_meter nedokáže číst pakety ze síťové karty, ale pouze ze souboru. Knihovna libpcap je vhodný způsob, jak tyto dva způsoby čtení zajistit. Navíc se jedná o standardizované řešení, které zajistí kompatibilitu na různých platformách.

Ostatní řešení, které byly zmíněny v analýze, jsou většinou dimenzovány na 10 Gb/s sítě. Knihovna libpcap bohatě stačí pro sítě do 1 Gb/s, na které flow_meter cílí. Pokud by bylo potřeba vyššího výkonu, lze libpcap překompilovat s PF_RING Vanilla nebo netmap bez nutnosti zásahu do exportéru.

3.2 Flow cache

Pro implementaci flow cache je potřeba najít vhodný efektivní datový typ. Cache musí být schopna vyhledání, přidání a smazání položky v co nejkratším čase. A pokud možno paměťově efektivní implementace s nízkým počtem výpadku cache procesoru.

Cache bude potřeba upravit a rozšířit tak, aby bylo možné ukládat rozšířené záznamy o tocích. Nejvhodnější způsob, jak ukládat informace navíc, je použití spojového seznamu. K jednotlivým záznamům o tocích může plugin uložit informace navíc. Pokud by záznam již informace navíc obsahoval, lze je jednoduše připojit nakonec spojového seznamu. Struktura záznamu bude navíc obsahovat jeden ukazatel a tedy tento způsob ukládání nebude paměťově náročný.

Exportér je potřeba optimalizovat pro běh na zařízeních s minimem operační paměti, řádově desítek MB. Tím lze docílit snížením paměťových nároků na položky flow cache. Snížit by se mohl například klíč z 76 bajtů na 37 v datové struktuře `Flow`, jelikož součet pětice IP adres, portů a protokolu je maximálně 37. Struktura `FlowRecord` obsahuje samostatně IP adresy verze 4 a verze 6. V záznamu o toku se může nacházet pouze jeden typ IP adres, proto by se mohly tyto položky sjednotit do jedné proměnné. Dále lze odstranit

nevyužité a zbytečné proměnné. Přeuspořádat položky ve struktuře tak, aby nedocházelo k plýtvání místem kvůli vložení výplně. Ze struktury `Flow` odstranit aktivní a neaktivní časový limit, protože jeho hodnota je stejná a nemusí být v programu vícekrát.

V poslední řadě je nutné optimalizovat exportér na procesorový výkon. K tomu by mohlo pomoci přepsání a nebo odstranění nepotřebného kódu ve flow cache. Optimalizace slabých míst profilováním. Změna hash funkce na takovou, která by splňovala následující vlastnosti:

- Rychlá nekryptografická 64 bitová hash funkce.
- Rovnoměrné rozložení hodnot (nízký počet kolizí). Vysoký počet kolizí by způsoboval předčasný export záznamů a s tím spojenou vysokou režii.

3.3 Ukázkové pluginy

Ukázkové pluginy pro HTTP a DNS provoz lze jednoduše realizovat pomocí parseru daného protokolu a využití potřebného API, které bude touto prací přidáno. Parsery protokolů budou vytvořeny tak, aby vyhovovaly standardům, které dané protokoly definují.

Přidané informace pluginy musí být exportér schopen bez problému exportovat. Pokud by flow_meter obsahoval pouze jedno výstupní TRAP rozhraní, posílané záznamy by byly velmi velké a přenášelo by se zbytečně moc informací. Toky s rozšířenými informacemi je proto vhodné exportovat na různá výstupní rozhraní modulu. Tohoto lze docílit tak, že každý plugin bude mít přiřazené vlastní výstupní TRAP rozhraní, kam se budou exportovat rozšířené toky pluginem.

3.4 Portace na OpenWrt

Díky rozdílným architekturám zařízeních by cross kompilace projektů vlastním způsobem byla poměrně složitá. Stávající projekty lze nicméně na OpenWrt portovat velmi snadno. Má totiž vlastní build system [31], umožňující jednoduše nakonfigurovat, zkompilovat a nainstalovat do distribuce vlastní balíček se softwarem. Balíček softwaru má v sobě tyto soubory:

Makefile obsahuje nastavení pro stažení, kompilaci, build a instalaci softwaru.

Config.in obsahuje definici položek v konfiguračním menu.

Patch soubory sloužící pro dodatečné úpravy souborů softwaru, například odstranění nepotřených zdrojových kódů.

Ostatní soubory, nejčastěji konfigurační, které budou instalovány do OpenWrt.

Balíček a nebo kolekce balíčků lze přidat do OpenWrt pomocí tzv. feedů. Instalace je pak velmi jednoduchá, do konfiguračního souboru v build systému se umístí cesta k feedu a pustí se script, který feed s balíčky stáhne. Nastavení OpenWrt a balíčků se dále provádí pomocí shell příkazu *make menuconfig*, který spustí grafické rozhraní s konfiguračním menu. Build a instalaci lze provést pomocí utility *make*. Zkompilovaný software lze vložit přímo do image, který se pak nahraje do zařízení, nebo lze vytvořit ipk soubor, ten lze nainstalovat do již existujícího OpenWrt zařízení.

Systém NEMEA spolu s exportérem lze do OpenWrt přidat pomocí výše uvedených postupů.

Kapitola 4

Realizace

4.1 Implementace čtení paketů

Do nové verze modulu flow_meter byla přidána možnost číst pakety ze síťové karty. Pro získání paketů je zvolena knihovna libpcap. Umožňuje číst jak ze síťové karty, tak i z pcap souboru. Ostatní výhody jsou zmíněny v kapitole s návrhem.

Čtení paketů a paket parser jsou umístěny ve třídě `PcapReader`. Parser je touto prací navíc předělán a rozšířen o možnost parsovat nové hlavičky paketů. flow_meter si nyní poradí i s:

- 802.1AD
- více 802.1Q tagů
- MPLS
- EoMPLS
- PPPoE
- IPv6 hop options
- IPv6 destination options
- IPv6 routing header
- IPv6 authentication header
- ICMP
- ICMPv6

4.2 Implementace flow cache

Flow cache nadále zůstává implementována jako hash tabulka. Tato struktura umožňuje vyhledání, vložení a smazání položky v $\mathcal{O}(1)$ konstantním čase. Kolize jsou řešeny řetězením záznamů za sebou. Na rozdíl od vyhledávacích stromů, je výhoda této implementace v lepším využití prostorové lokality cache v procesoru, jelikož jsou položky v paměti uloženy jako jeden velký blok za sebou. A také umožňuje snadno iterovat jednotlivé položky cache pro kontrolu aktivních a neaktivních časových limitů.

Co se týče podpory aplikačních rozšíření, datová struktura `Flow`, položka flow cache, nově dědí ze struktury `Record` spojový seznam. Prvek spojového seznamu `RecordExt` obsahuje ukazatel na další prvek a typ záznamu. Typ záznamu určuje, o jaké aplikační informace se jedná. Struktura `Record` obsahuje API pro přidání rozšíření na konec spojového seznamu, nalezení rozšíření nebo odstranění.

Struktura `RecordExt` slouží jako vzor, ze které datové struktury definované pluginem dědí. Obsahuje metody pro uložení informací do UniRec (`fillUnirec()`) nebo IPFIX zprávy (`fillIPFIX()`). Každý plugin si uvnitř těchto metod definuje, jak se záznamy které vytváří, mají před exportem vyplnit do daných formátů. V poslední řadě si také definuje nové členské proměnné, do kterých uloží aplikační informace.

```
enum extTypeEnum {
    http_request = 0,
    http_response,
    dns,
    EXTENSION_CNT
};

struct RecordExt {
    RecordExt *next;
    extTypeEnum extType;

    RecordExt(extTypeEnum type);
    virtual void fillUnirec(ur_template_t *tmplt,
                    void *record);
    virtual int fillIPFIX(uint8_t *buffer, int size);
    virtual ~RecordExt();
};

struct Record {
    RecordExt *exts;

    Record();
```

```
    void addExtension(RecordExt* ext);
    RecordExt *getExtension(extTypeEnum extType);
    void removeExtensions();
    ~Record();
};
```

Paměťové optimalizace je docíleno odstraněním zbytečných položek z flow záznamů. Velikost struktury `FlowRecord` je snížena z 112 na 104 bajtů a `Flow` z 240 na 112 bajtů. Ke snížení paměťových nároků přispívá také redukce velikosti výplně ve struktuře vhodným rozmístěním položek.

Vzhledem k 53 % úspoře místa ve flow cache je výchozí počet záznamů zvýšen na 131072. Kdykoliv lze počet záznamů změnit pomocí *-s* parametru modulu.

Výslednou podobu proměnných ve strukturách popisuje následující kód. Unie `ipaddr_t` slouží k uložení IPv4 nebo IPv6 adresy. Ve třídě `Flow` zbyl pouze záznam o toku, `FlowRecord`, a hash nutný k identifikaci ve flow cache. Hodnoty aktivního a neaktivního časového limitu byly přesunuty do flow cache.

```
typedef union ipaddr_u {
    uint8_t  v6[16];
    uint32_t v4;
} ipaddr_t;

struct FlowRecord : public Record {
    struct timeval time_first;
    struct timeval time_last;
    uint64_t octet_total_length;
    uint32_t pkt_total_cnt;
    uint8_t  tcp_control_bits;

    uint8_t  ip_version;
    uint8_t  ip_tos;
    uint8_t  ip_ttl;

    uint8_t  ip_proto;
    uint16_t src_port;
    uint16_t dst_port;
    ipaddr_t src_ip;
    ipaddr_t dst_ip;
};

class Flow {
    uint64_t hash;
public:
```

```
    FlowRecord flow;
}
```

Z hlediska optimalizace na rychlost je činnost flow cache zrychlena pomocí nové hash funkce xxHash [32]. Dále jsou identifikována problémová místa, která způsobují zpomalení a byla zjednána náprava. Například složitý výpočet pozice řádku dělením byl zjednodušen na pouhé maskování proměnné (bitová operace AND). Redukce počtu výpadku cache v procesoru snížením velikosti řádku flow cache z 32 na 16 položek. Nepotřebný kód je odstraněn. Aktivní časový limit je kontrolován při vkládání paketu do flow cache. Měření výkonnosti flow cache je popsáno později.

Následující kód popisuje zjednodušený výpočet pozice řádku. Proměnná `line_size_mask` obsahuje bity, kterými se bitovou operací AND získá pozice řádku. Podmínkou je, že velikost cache a velikost řádku musí být mocnina dvou.

```
uint64_t hashval = XXH64(key, key_len, 0);

uint32_t line_index = hashval & line_size_mask;
```

xxHash je nekryptografická 64 bitová hash funkce. Zvolení právě této funkce bylo učiněno na základě benchmarku popsaném v tabulce 4.1. Benchmark používá nástroj SMHasher a testuje hash funkce a kontrolní součty na distribuci hodnot, množství kolizí a výkonnost. Sloupce s kvalitou určuje bodové hodnocení v prvních dvou testech. Hodnota 10 je nejlepší a funkce s hodnotu menší než 5 nejsou v tabulce zobrazeny. Srovnání výkonnosti původní funkce collate a nové xxHash je popsáno v kapitole s měřením výkonnosti.

Tabulka 4.1: Výsledky testů funkcí v SMHasher nástroji. Tabulka je převzatá z [33].

Jméno	Rychlost	Kvalita	Autor
xxHash	5,4 GB/s	10	Y. C.
MurmurHash 3a	2,7 GB/s	10	Austin Appleby
SBox	1,4 GB/s	9	Bret Mulvey
Lookup3	1,2 GB/s	9	Bob Jenkins
CityHash64	1,05 GB/s	10	Pike & Alakuijala
FNV	0,55 GB/s	5	Fowler, Noll, Vo
CRC32	0,43 GB/s	9	
MD5-32	0,33 GB/s	10	Ronald L. Rivest
SHA1-32	0,28 GB/s	10	

4.3 Implementace pluginů

Na ukázkových implementacích HTTP a DNS pluginů jsou demonstrovány principy a postup při tvorbě nového pluginu.

4.3.1 HTTP plugin

Plugin pro parsování a export HTTP provozu je implementován, aby vyhovolal specifikaci standardu RFC 7230. Plugin ve funkci `post_create()` zparsuje nově příchozí paket a aplikační položky vloží k záznamu toku.

V HTTP paketu může nacházet hlavička *Connection: keep-alive*, která indikuje, že TCP spojení se po odpovědi serveru neukončuje. V navázaném spojení je potřeba rozlišit jednotlivé požadavky a odpovědi ve spojení. Toho je docíleno ve funkci `pre_update()`, která vyhledá existující aplikační rozšíření v toku a zparsuje paket. Pokud se parsování nepovede, paket obsahuje data, pokud se povede, paket obsahuje nový požadavek nebo odpověď a je potřeba záznam exportovat. Plugin návratovým kódem vynutí exportování toku a celý proces vkládání do flow cache se děje znovu. Tímto jsou i v rámci jednoho spojení všechny požadavky a odpovědi zachyceny a samostatně exportovány.

Pro moduly by bylo poněkud nepraktické mít jedno výstupní rozhraní pro export požadavků a druhý pro export odpovědí. Položky jsou exportovány na stejné TRAP rozhraní. UniRec políčka na výstupním rozhraní popisuje tabulka 4.2.

Tabulka 4.2: Šablona UniRec políček na výstupním rozhraní pro HTTP provoz.

Název UniRec políčka	Typ	Popis políčka
HTTP_METHOD	string	HTTP metoda dotazu
HTTP_HOST	string	položka Host v dotazu
HTTP_URL	string	URI v dotazu
HTTP_USER_AGENT	string	položka User-Agent v dotazu
HTTP_REFERER	string	položka Referer v dotazu
HTTP_RESPONSE_CODE	uint16	návratový kód odpovědi
HTTP_CONTENT_TYPE	string	typ obsahu v odpovědi

Z HTTP požadavku jsou parsovány a exportovány následující položky:

- Metoda
- URI
- Host
- UserAgent
- Referer

Z HTTP odpovědi jsou zpracovávány a exportovány tyto položky:

- Návratový kód odpovědi
- ContentType

Datové struktury HTTP pluginu popisuje následující kód. Třída `HTTPPlugin` se stará o parsování paketů a vkládání k existujícím záznamům o tocích. Plugin má definované dvě struktury, `RecordExtHTTPReq` pro HTTP dotaz a `RecordExtHTTPResp` pro HTTP odpověď.

```
struct RecordExtHTTPReq : RecordExt {
    char method[10];
    char host[64];
    char uri[128];
    char user_agent[128];
    char referer[128];

    RecordExtHTTPReq();
    virtual void fillUnirec(ur_template_t *tmplt,
            void *record);
    virtual int fillIPFIX(uint8_t *buffer, int size);
};

struct RecordExtHTTPResp : RecordExt {
    uint16_t code;
    char content_type[32];

    RecordExtHTTPResp();
    virtual void fillUnirec(ur_template_t *tmplt,
            void *record);
    virtual int fillIPFIX(uint8_t *buffer, int size);
};
```

4.3.2 DNS plugin

Plugin pro export aplikačních políček ze služby DNS je implementován, aby vyhovoval specifikaci RFC 1035, 4034 [34] a 7766 [35]. Podporuje jak UDP, tak i dotazy přes TCP protokol.

DNS plugin využívá ke své činnosti pouze funkci `post_create()`. Vstupní paket je zparsován a uložen k záznamu toku. DNS požadavek nebo odpověď je obsažena v jednom jediném paketu, proto plugin návratovým kódem indikuje, že se rozšířený záznam s tokem má okamžitě exportovat. Šablonu na výstupním rozhraní pro DNS provoz popisuje tabulka 4.3.

Struktury definované DNS pluginem popisuje následující kód. K uložení informací postačí jediná struktura, `RecordExtDNS`, kterou využívá třída `DNSPlugin`.

Tabulka 4.3: Šablona UniRec políček na výstupním rozhraní pro DNS provoz.

Název UniRec políčka	Datový typ	Popis políčka
DNS_ID	uint16	ID transakce
DNS_ANSWERS	uint16	počet záznamů s odpovědí
DNS_RCODE	uint8	návratový kód
DNS_NAME	string	doménové jméno v první otázce
DNS_QTYPE	uint16	typ první otázky
DNS_CLASS	uint16	třída první otázky
DNS_RR_TTL	uint32	TTL políčko záznamu otázky
DNS_RLENGTH	uint16	délka položky DNS_RDATA
DNS_RDATA	bytes	specifická data záznamu
DNS_PSIZE	uint16	maximální velikost paketu žadatele
DNS_DO	uint8	DNSSEC OK bit

```
struct RecordExtDNS : RecordExt {
   uint16_t id;
   uint16_t answers;
   uint8_t rcode;
   char qname[128];
   uint16_t qtype;
   uint16_t qclass;
   uint32_t rr_ttl;
   uint16_t rlength;
   char data[160];
   uint16_t psize;
   uint8_t dns_do;

   RecordExtDNS();
   virtual void fillUnirec(ur_template_t *tmplt,
                           void *record);
   virtual int fillIPFIX(uint8_t *buffer, int size);
};
```

Parser zpracovává tyto typy záznamů: A, AAAA, NS, CNAME, DNAME, PTR, SOA, SRV, MX, TXT, MINFO, HINFO, ISDN, DS, RRSIG a DNSKEY. Ze záznamů v paketu jsou extrahovány a exportovány tyto položky:

- ID transakce
- Doménové jméno v otázce
- Typ otázky
- Třída otázky

- TTL políčko záznamu
- DNSSEC OK bit
- Specifická data záznamu
- Počet DNS odpovědí
- Návratový kód odpovědi

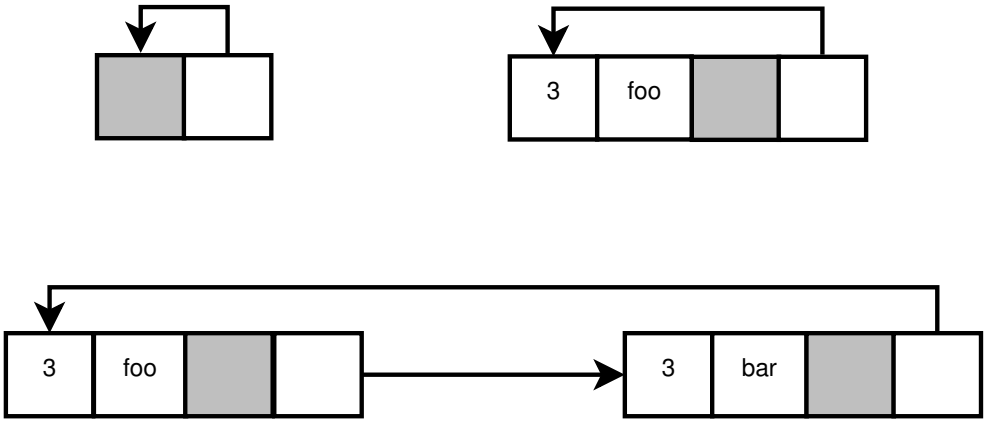
Dekódování doménového jména může skrývat spoustu problémů. DNS parser obsahuje ochranu proti následujícím exploitům při dekódování doménového jména:

Label pointer ukazující sám na sebe způsobí zacyklení exportéru při iterativním dekódování. Při rekurzivním dekódování doménového jména způsobí přetečení zásobníku programu.

Smyčka přes více pointerů je podobný exploit jako v předchozím případě. Rozdíl je, že dva a více ukazatelů nyní ukazují navzájem na sebe.

Dlouhé doménové jméno by mohlo způsobit vyčerpání dostupné paměti.

Nemůže se proto stát, že by útočník poslal modifikovaný DNS paket a exportér by kvůli tomu havaroval. Ukázku problémových případů popisuje obrázek 4.1.



Obrázek 4.1: Příklad problémových kódování doménových jmen

4.4 Výstupy exportéru

4.4.1 Implementace TRAP rozhraní

Podpora více TRAP rozhraní je implementována ve třídě `UnirecExporter`. Při inicializaci třídy je předán seznam pluginů spolu s konfigurací, na jaké

rozhraní bude který plugin exportovat. Třída poté provede potřebnou inicializaci pomocí API definovaného v knihovně libtrap.

Při exportu záznamu se rozhoduje, na které rozhraní se jaký tok pošle. Toho je docíleno pomocí identifikátoru obsaženém v rozšířeních u záznamů. Ještě před exportem se na rozšířený záznam zavolá funkce `fillUnirec`, která do zprávy vyplní informace ze struktury. Zpráva je poté odeslána přes příslušné výstupní rozhraní.

4.4.2 Export v IPFIX

Tato práce přidává možnost exportovat síťové toky ve více flexibilním IPFIX formátu. Tím se flow_meter stává mnohem využitelnějším nástrojem a umožňuje zprostředkovat zachycené toky ostatním systémům, které nepoužívají knihovnu libtrap.

K exportu byla využita již existující implementace exportního pluginu z Flowmon exportéru [36] napsanou v jazyce C od Petra Velana. Použita je ve třídě `IPFIXExporter`. Vzhledem k architektuře modulu flow_meter bylo nutné přepsat tuto implementaci do C++ a odstranit část nepotřebného kódu. Definice IPFIX šablon je pevně daná v kódu a nelze ji měnit ani pomocí konfiguračního souboru. Na IPFIX kolektor lze vytvořit spojení přes TCP nebo UDP protokol.

4.5 Portace na OpenWrt

NEMEA projekt byl přidán do OpenWrt pomocí feedu obsahující dva balíčky, jeden s modulem flow_meter a druhý s knihovnami z NEMEA framework. Feed si může kdokoliv přidat do svého OpenWrt build systému a vytvořit balíky. Více se lze dočíst v návodu obsaženém v příloze C.2.

V konfiguračním menu build systému lze nastavit velikost flow cache pro flow_meter a také velikost bufferu u rozhraní v knihovně libtrap. Výchozí nastavení velikosti flow cache je snížena z 131072 na 8192 položek a velikost bufferu ze 100000 na 5000 bajtů. Díky této konfiguraci lze bez problému spustit flow_meter na zařízeních s alespoň 32 MB paměti. Pokud by velikost flow cache nestačila, lze ji dodatečně změnit pomocí parametru modulu.

Pro snadnější správu modulu flow_meter na OpenWrt byl vytvořen init skript umožňující jej spustit na pozadí jako službu. Přidána byla i možnost vytvoření vlastních profilů pro spouštění více instancí exportéru. Skript se po instalaci nachází v */etc/init.d/flow_meter* a lze jím jednotlivé profily ovládat. Mezi základní operace patří spuštění, vypnutí, restartování služby a zapínání instancí profilů při startu systému. Profily a k nim přiřazené instance exportérů lze konfigurovat pomocí souboru */etc/config/flow_meter*. Ukázková konfigurace profilu *wan* pak vypadá následovně:

```
config profile wan
```

```
option interface eth0
option plugins basic,dns,http
option trap_ifc_spec t:10000,t:10001,t:10002
option ipfix_collector 192.168.0.2:4739
option ipfix_enable 0
option ipfix_udp 0
option cache_size default
option timeouts default
option link 0x1
option dir 0
option respawn 1
option respawn_threshold 3600
option respawn_timeout 5
option respawn_retry 5
option core 1
option enabled 1
```

Pomocí této konfigurace bude běžící instance exportéru číst provoz ze síťového rozhraní *eth0*. Základní toky se posílají přes libtrap TCP rozhraní na portu *10000*, toky rozšířené o HTTP provoz na rozhraní *10001* a toky rozšířené o DNS na *10002*. Velikost cache a časové limity mají výchozí hodnoty, identifikátor exportéru je hodnota *1* a je zapnutý respawn profilu, který umožní aplikaci restartovat v případě havárie. Volba *core* nastavena na *1* vytvoří při pádu aplikace core dump soubor. Položka *enabled*, nastavená na *1*, způsobí, že při startu systému bude instance profilu spuštěna.

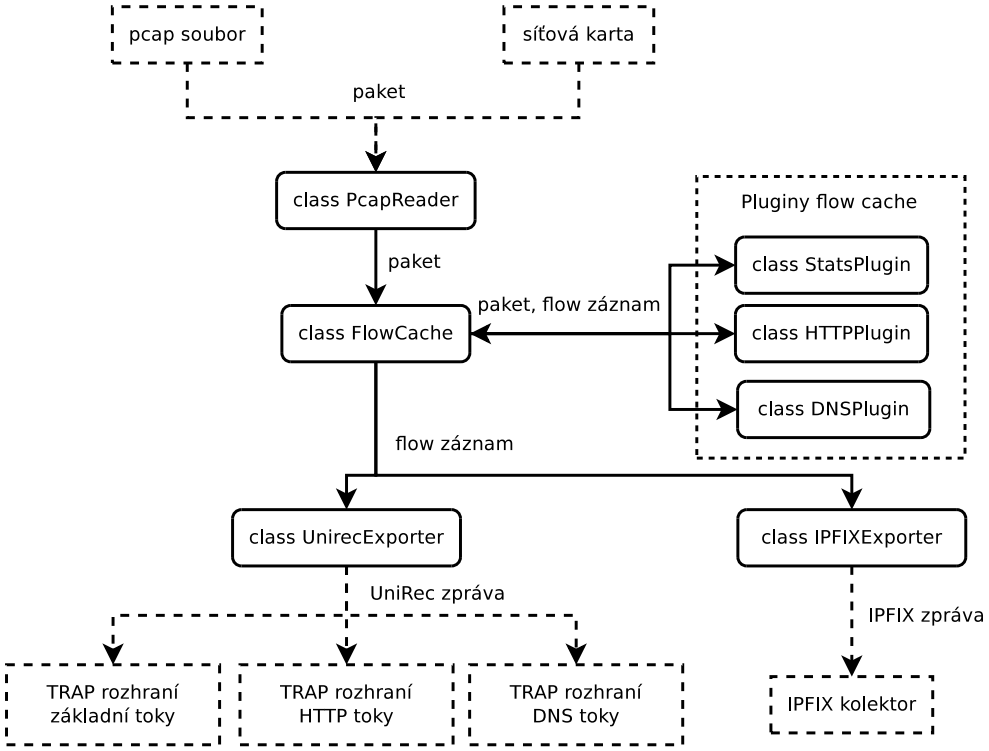
Z důvodu rozdílných architektur směrovačů a běžných počítačů se mohou vyskytnout problémy související s pořadím bajtů v paměti. UniRec byl navržen pro x86 systémy s little endian pořadím bytů. Z výkonnostních důvodů se při odesílání zprávy nepřevádí pořadí bajtů ve zprávě. Při exportu toků v UniRec formátu ze směrovače s big endian pořadím bajtů by se zprávy špatně interpretovaly. Z tohoto důvodu je do NEMEA systému přidán nový modul, endiverter, pro převod pořadí bajtů v UniRec zprávě.

4.6 Výsledná struktura exportéru

Rozšíření a výsledná struktura exportéru je zobrazena na obrázku 4.2.

Vylepšení přidává možnost číst pakety ze síťové karty pomocí knihovny libpcap ve třídě `PcapReader`. Do flow cache byly přidány dva nové pluginy, `HTTPPlugin` a `DNSPlugin` pro parsování provozu. Do třídy `UnirecExporter`, která se stará o export v UniRec formátu, je přidána možnost exportovat rozšířené toky na různá TRAP rozhraní. Přidána je také nová třída pro export `IPFIXExporter`, která umožňuje exportovat toky na kolektor ve formátu IPFIX.

Uživatel si může zvolit, která exportní třída bude toky exportovat a které pluginy budou ve flow cache aktivní. Počet výstupních TRAP rozhraní modulu je odvozeno od počtu aktivních parsovacích pluginů ve flow cache.



Obrázek 4.2: Vnitřní struktura vylepšeného exportéru

Kapitola 5

Ukázka

5.0.1 UniRec

Jedním z možných využití nového exportéru flow_meter je dodávat zachycené toky do systému NEMEA. Toky lze jednorázově přečíst ze souboru, nebo kontinuálně číst ze síťového rozhraní. V této sekci je popsána ukázka spuštění nainstalovaného NEMEA systému na dvou zařízení *A* a *B* s linux distribucí. Návod na instalaci systému NEMEA je obsažen v příloze C.1.

Následující shellový příkaz spustí flow_meter na síťovém rozhraní *eth0* na zařízení A. Přepínač *-p* slouží ke specifikaci pluginů ve flow cache (*basic* je označení pro pseudo plugin pro export základních toků). Přepínačem *-i* se nastaví vstupní a výstupní TRAP rozhraní, pořadí rozhraní v modulu flow_meter koresponduje s pořadím pluginů v parametru *-p*. HTTP toky se ukládají do souboru */data/http*, DNS na *dns* Unix rozhraní a základní toky na TCP rozhraní na portu *7000*.

 # A:
```
$ flow_meter -i f:/data/http,u:dns,t:7000 -I eth0 \
        -p http,dns,basic
```

Na spuštěný flow_meter lze napojit další moduly, které z exportéru budou přijímat UniRec zprávy s toky. Modul *dnstunnel_detection* bude přijímat na stejném fyzickém zařízení DNS toky z Unix rozhraní *dns* a provádět nad nimi detekci DNS tunelů. Informace o detekovaných hrozbách odesílá přes Unix rozhraní *dnstun_out* do modulu *email_reporter*, který je přeposílá na vybranný email. Ze zařízení B se přes TCP rozhraní připojí modul pro detekci horizontálních skenů portů na síti, *haddrscan_detector*. Detekované skeny odesílá do modulu *haddrscan_detector*, který je shromažďuje a jednou za 30 minut odešle modulu *email_reporter*. Na obrázku 5.1 je vizualizace popsané konfigurace NEMEA systému.

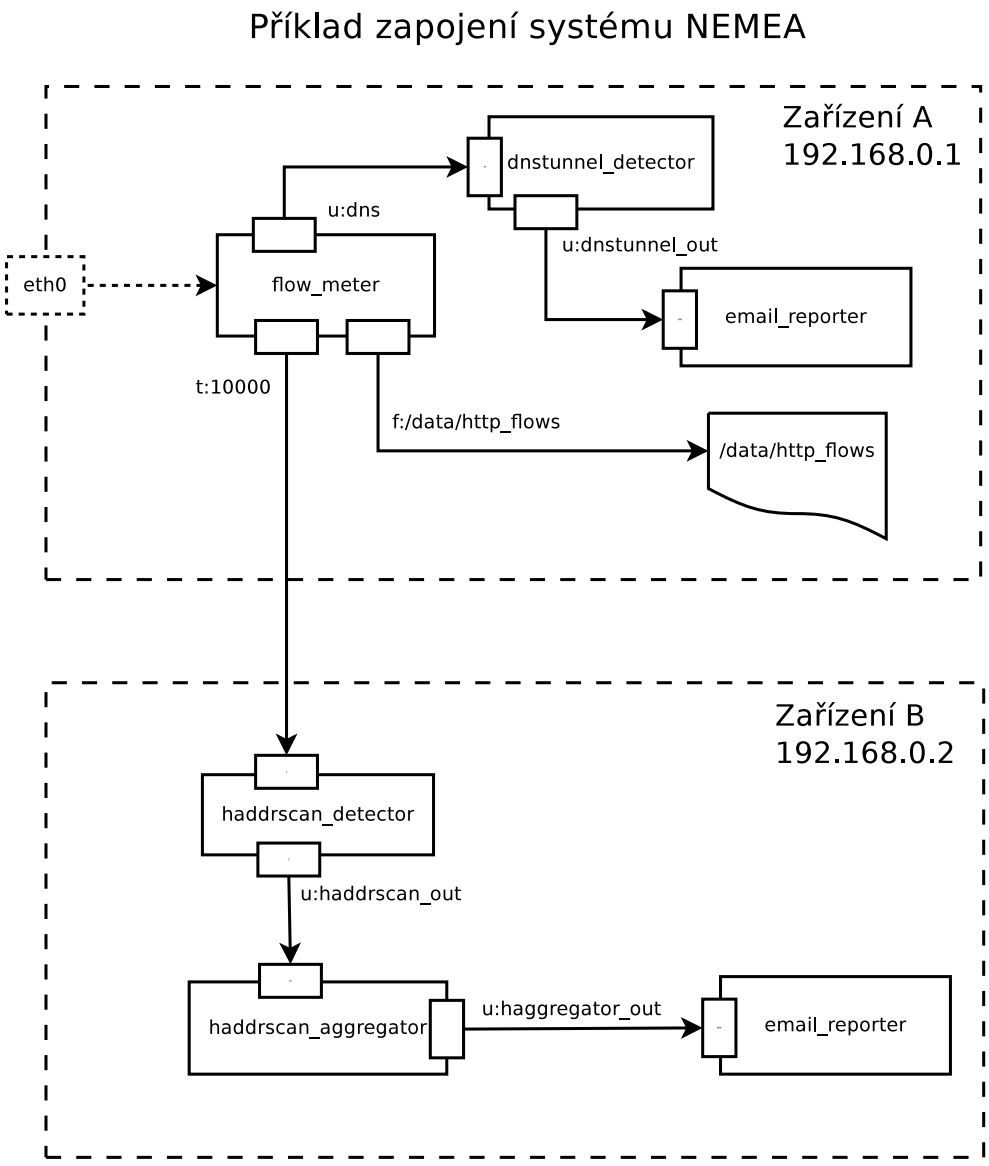
 # A:
```
$ dnstunnel_detection -i u:dns,u:dnstun_out
```

```
$ python email_reporter.py -i u:dnstun_out \
                /etc/nemea/smtp.cfg
# B:
$ haddrscan_detector -i t:192.168.0.1:7000,u:hscan_out
$ python haddrscan_aggregator.py \
                -i u:hscan_out,u:hagg_out -t 30
$ python email_reporter.py -i u:hagg_out \
                /etc/nemea/smtp.cfg
```

5.0.2 IPFIX

Druhým způsobem využití je exportovat toky na IPFIX kolektor. Spuštění exportéru je podobné jako v při exportu do systému NEMEA. Místo přepínače *-i* je použit *-x*, přes který se dá specifikovat adresa vzdáleného IPFIX kolektoru. Přepínačem *-u* se jako transportní protokol pro IPFIX zprávy zvolí UDP, jinak je výchozí TCP protokol.

```
$ flow_meter -x 10.0.0.2:4739 -u -I eth0 -p http,dns,basic
```



Obrázek 5.1: Ukázkové zapojení systému NEMEA s novým exportérem. Menší krabičky se šipkami znázorňují TRAP rozhraní a jejich propojení.

Kapitola 6

Měření výkonu

6.1 Flow cache

Tato sekce si klade za cíl otestovat výkonnost vylepšených a optimalizovaných datových struktur a algoritmů flow cache. Celý test probíhal bez účasti čtení paketů ze souboru, sítě, použití parsovacích pluginů nebo odesílání toků na TRAP rozhraní. Výsledky testu propustnosti flow cache jsou uvedeny v tabulce 6.1.

Měření probíhalo na stroji s procesorem *Intel(R) Core(TM) i7-4785T CPU 2,20 GHz* a *16 GB RAM*. Program byl zkompilován pomocí *g++* s parametry *-Wall -O3 -std=c++98*.

První případ testu ukazuje, jak si flow cache dokáže poradit s jedním velkým IPv4 nebo IPv6 tokem. Na začátku testu byl náhodně vygenerován jeden paket. Ten byl poté vkládán do flow cache po dobu jedné minuty. Bylo provedeno 50 měření a výsledek byl zprůměrován.

Druhý případ testuje zachycování jedno-paketových IPv4 a IPv6 toků. Před každým vkládáním do flow cache je paket nově vytvořen. Tvorba paketu neprobíhá náhodně jako v prvním testu, ale pomocí čítače a násobení prvočísly. Před vytvářením paketu je hodnota čítače vždy zvýšena o 1 a každému prvku z pětice identifikující tok je přiřazena nová hodnota. Hodnota prvku se vypočítá jako násobek čítače a prvočísla, kde prvočíslo je pro každý prvek jiné. Tento způsob zajistí, že test nebude zkreslen náhodou a podmínky pro testy

Tabulka 6.1: Propustnost flow cache v počtu milionu zpracovaných paketů za sekundu.

flow cache	jeden veliký tok		jedno-paketové toky	
	IPv4	IPv6	IPv4	IPv6
nová s xxHash	19,0	16,7	3,4	3,3
nová s collate	14,7	12,0	3,1	2,9
stará s collate	5,5	5,5	1,4	1,3

obou verzí cache budou stejné.

6.2 Výkon bez pluginů

Měření probíhalo pomocí programu *tcpreplay-edit*. Na směrovač *TP-LINK Archer C7 AC1750* (*720 Mhz CPU, 128 MB RAM, 1 Gb/s ethernet*) s OpenWrt Chaos Calmer 15.05.1 distribucí byla ve smyčce posílána zachycená komunikace v pcap souboru. Spuštěný flow_meter s jedním zapnutým TCP TRAP rozhraním se tuto komunikaci pokoušel zachytit. Pcap soubor obsahoval 5655 IPv4 a IPv6 paketů s HTTP, HTTPS, DNS, ICMP a DHCP provozem.

Testování probíhalo vícekrát. Snahou bylo najít limit, kdy exportér nezahazuje žádné pakety nebo je procento zahozených paketů menší než 1. Výsledkem je zachycených 73,2 tisíc paketů za sekundu při rychlosti 415 Mb/s.

6.3 Výkon HTTP plugin

Testování probíhalo jako v předchozím případě, na směrovač byly odesílány pakety ze souboru. flow_meter měl zapnutý pouze HTTP plugin a obsahoval jedno TCP rozhraní. Pcap soubor s HTTP komunikací obsahoval 2035 dotazů, 2070 odpovědí a 52444 HTTP paketů s daty. Průměrná velikost paketu je 1458 bajtů.

Výsledek testu je, že flow_meter stíhal zachytit 20 tisíc HTTP paketů za sekundu při rychlosti 229 Mb/s.

6.4 Výkon DNS plugin

Stejný scénář testování. Zapnutý byl pouze DNS plugin s TCP rozhraním. DNS pcap soubor obsahoval 8826 dotazů a 8293 odpovědí, celkem 17119 paketů. Průměrná velikost paketu je 170 bajtů.

DNS paket je mnohem strukturovanější a tím pádem složitější na zpracování než HTTP paket. V tomto testu flow_meter dokázal zachytit 6,8 tisíc DNS paketů za sekundu při rychlosti 8,85 Mb/s.

Závěr

Zadáním této bakalářské práce bylo prostudovat a rozšířit existující exportér síťových toků ze systému NEMEA o možnost exportu informací z aplikačních protokolů. Analyzovat existující struktury a algoritmy ve flow cache a navrhnout jejich optimalizaci. Flow cache rozšířit o možnost ukládání aplikačních protokolů. Jako ukázku použitých datových struktur a algoritmů implementovat export HTTP nebo DNS protokolu a změřit výkonnost exportéru.

V kapitole s analýzou je popsáno monitorování síťových toků a systém NEMEA. Jsou zde zmíněny existující metody pro získávání paketů, formáty pro export toků, existující exportéry toků a protokoly HTTP a DNS. Popsán je i systém NEMEA a jednotlivé komponenty, zejména existující exportér flow_meter spolu s jeho použitými strukturami a algoritmy.

V kapitole s návrhem jsou zmíněna možná rozšíření a optimalizace exportéru. Diskuze je vedena nad důležitými částmi exportéru, které je potřeba přidat, předělat nebo rozšířit.

Kapitola s realizací obsahuje popis nových datových struktur a algoritmů. Popsána je volba knihovny libpcap pro čtení paketů a rozšíření parseru protokolů. Pozornost je věnována implementaci a optimalizaci flow cache, snížení její velikosti v paměti o 53% a rozšíření o možnost ukládat aplikační informací parsovacími pluginy. Jako ukázka použití nových datových struktur jsou implementovány pluginy pro parsování a export obou protokolů, HTTP a DNS. Pro export síťových toků s aplikačními informacemi je přidána možnost odesílat záznamy o tocích na různá výstupní TRAP rozhraní exportéru. Navíc je přidána možnost exportovat záznamy o tocích ve formátu IPFIX. Dále byl exportér úspěšně portován na linuxovou distribuci OpenWrt.

Kapitola s ukázkou popisuje příklad použití nového exportéru. Popsán je zde scénář využití exportéru jako zdroj dat pro systém NEMEA. Zmíněno je i použití pro export na IPFIX kolektory pro ostatní systémy.

Měření výkonu se skládá z testů propustnosti flow cache a testů výkonnosti bez a s HTTP a DNS pluginy. Vylepšená flow cache má v testu s jedním velkým tokem 3,5krát větší propustnost paketů za sekundu a v testu s jedno-

paketovými toky 2,4krát. V testu výkonnosti dokázal flow_meter bez pluginů zachytit 73,2 tisíc paketů za sekunu (pps) při rychlosti 415 Mb/s. HTTP plugin zvládl zachytit 20 tisíc pps při 229 Mb/s. DNS plugin zvládl 6,8 tisíc pps při 8,85 Mb/s. Testování probíhalo na směrovači *TP-LINK Archer C7 AC1750*.

Výsledky této práce využili dva studenti pro své úspěšně obhájené závěrečné práce. Tomáš Jánský v bakalářské práci „Aplikace pro analýzu síťových toků technologie VoIP/SIP“ a Alejandro Robledo v diplomové práci „Network Time Protocol attacks detection“. flow_meter díky nim obsahuje dva nové pluginy pro export SIP a NTP provozu. Exportér z této práce se navíc povedlo přidat do EduroamAP směrovačů, kde nahradil stávající softflowd exportér.

Ve vývoji exportéru se bude i nadále pokračovat. Potenciálním zlepšením může být vylepšení čtecího mechanismu paketů nebo vylepšení kontroly neaktivních časových limitů ve flow cache.

Literatura

[1] Čejka, T.; Rosa, Z.; Kubátová, H.: Stream-wise Detection of Surreptitious Traffic over DNS. In *IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*, Athens, 2014.

[2] Čejka, T.; Bartoš, V.; Truxa, L.; aj.: Using Application-Aware Flow Monitoring for SIP Fraud Detection. In *Proc. of 9th International Conference on Autonomous Infrastructure, Management and Security (AIMS15)*, 2015.

[3] Jánský, T.; Čejka, T.; Bartoš, V.: Hunting SIP Authentication Attacks Efficiently. In *AIMS 2017*, 2017.

[4] Čejka, T.; Bodó, R.; Kubátová, H.: Nemea: Searching for Botnet Footprints. In *Proceedings of the 3rd Prague Embedded Systems Workshop*, Prague, 2015.

[5] 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)*, Canada, 2016.

[6] Linux programmer's manual: PACKET(7). Dostupné z: `http://man7.org/linux/man-pages/man7/packet.7.html`

[7] Asynchronous packet socket reading with PACKET_RX_RING. Dostupné z: `https://codemonkeytips.blogspot.cz/2011/07/asynchronous-packet-socket-reading-with.html`

[8] Knihovna libpcap. Dostupné z: `http://www.tcpdump.org/`

[9] Framework PF_RING Vanilla. Dostupné z: `http://www.ntop.org/products/packet-capture/pf_ring/`

[10] Framework PF_RING Zero Copy. Dostupné z: http://www.ntop.org/products/packet-capture/pf_ring/pf_ring-zc-zero-copy/

[11] Rizzo, L.: netmap: a novel framework for fast packet I/O. In *USENIX annual technical conference*, Boston, 2012. Dostupné z: https://www.usenix.org/system/files/conference/atc12/atc12-final186.pdf

[12] Rizzo, L.: netmap info. Dostupné z: http://info.iet.unipi.it/~luigi/netmap/

[13] Intel DPDK framework. Dostupné z: http://dpdk.org/

[14] Myricom Sniffer 10G. Dostupné z: http://brookstonegroup.com/group/down/Sniffer%20Packet%20Capture%20Software.pdf

[15] NetFlow v5. Dostupné z: https://www.cisco.com/c/en/us/td/docs/net_mgmt/netflow_collection_engine/3-6/user/guide/format.html

[16] RFC 3954: Cisco Systems NetFlow Services Export Version 9. Dostupné z: https://www.ietf.org/rfc/rfc3954.txt

[17] RFC 7011: Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. Dostupné z: https://tools.ietf.org/html/rfc7011

[18] NEMEA UniRec. Dostupné z: https://www.cesnet.cz/wp-content/uploads/2014/02/trapnemea.pdf

[19] FlowMon probe. Dostupné z: https://www.flowmon.com/en/products/flowmon/probe

[20] nProbe. Dostupné z: http://www.ntop.org/products/netflow/nprobe/

[21] Introduction to Cisco IOS NetFlow - A Technical Overview. Dostupné z: https://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/ios-netflow/prod_white_paper0900aecd80406232.html

[22] fprobe. Dostupné z: https://sourceforge.net/projects/fprobe/

[23] softflowd. Dostupné z: http://www.mindrot.org/projects/softflowd/

[24] NFDUMP. Dostupné z: http://nfdump.sourceforge.net/

[25] flow_meter exportér. Dostupné z: https://github.com/CESNET/Nemea-Modules/tree/master/flow_meter

[26] NEMEA UniRec políčka. Dostupné z: https://github.com/CESNET/Nemea-Framework/blob/78672a5d9391e24421960d7f7d273dc881127699/unirec/README

[27] collate hash. Dostupné z: http://en.cppreference.com/w/cpp/locale/collate/hash

[28] RFC 7230: Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing. Dostupné z: https://tools.ietf.org/html/rfc7230

[29] RFC 1035: DOMAIN NAMES - IMPLEMENTATION AND SPECIFICATION. Dostupné z: https://www.ietf.org/rfc/rfc1035.txt

[30] OpenWrt. Dostupné z: https://wiki.openwrt.org/doc/start

[31] OpenWrt build system. Dostupné z: https://github.com/openwrt/openwrt

[32] xxHash. Dostupné z: https://github.com/Cyan4973/xxHash

[33] xxHash benchmark. Dostupné z: https://cyan4973.github.io/xxHash/

[34] RFC 4034: Resource Records for the DNS Security Extensions. Dostupné z: https://tools.ietf.org/html/rfc4034

[35] DNS Transport over TCP - Implementation Requirements. Dostupné z: https://tools.ietf.org/html/rfc7766

[36] Velan, P.: Flowmon export IPFIX. Dostupné z: https://dior.ics.muni.cz/~velan/flowmon-export-ipfix/

Příloha A

Seznam použitých zkratek

NEMEA Network Measurements Analysis

TRAP Traffic Analysis Platform

UniRec Unified Record

IPFIX Internet Protocol Flow Information Export

IP Internet Protocol

HTTP Hypertext Transfer Protocol

DNS Domain Name System

SIP Session Initiation Protocol

NTP Network Time Protocol

Příloha B

Deklarace tříd pluginů

B.1 HTTP

```
class HTTPPlugin : public FlowCachePlugin
{
public:
   HTTPPlugin(const options_t &module_options);
   int post_create(Flow &rec, const Packet &pkt);
   int pre_update(Flow &rec, Packet &pkt);
   void finish();
   string get_unirec_field_string();
   const char **get_ipfix_string();

private:
   bool parse_http_request(const char *data,
         int payload_len, RecordExtHTTPReq *rec,
         bool create);
   bool parse_http_response(const char *data,
         int payload_len, RecordExtHTTPResp *rec,
         bool create);
   int add_ext_http_request(const char *data,
         int payload_len, Flow &rec);
   int add_ext_http_response(const char *data,
         int payload_len, Flow &rec);
   bool valid_http_method(const char *method) const;
}
```

B.2 DNS

```
class DNSPlugin : public FlowCachePlugin
{
```

```
public:
   DNSPlugin(const options_t &module_options);
   int post_create(Flow &rec, const Packet &pkt);
   void finish();
   string get_unirec_field_string();
   const char **get_ipfix_string();

private:
   bool parse_dns(const char *data,
         unsigned int payload_len, bool tcp,
         RecordExtDNS *rec);
   int  add_ext_dns(const char *data,
         unsigned int payload_len, bool tcp, Flow &rec);
   void process_srv(string &str) const;
   void process_rdata(const char *record_begin,
      const char *data, ostringstream &rdata,
      uint16_t type, size_t length) const;

   string get_name(const char *data) const;
   size_t get_name_length(const char *data) const;
}
```

Příloha C

Instalace systému NEMEA

Návody popisují, jak pomocí příkazů v terminálu zkompilovat a zprovoznit nový exportér na systému s linux distribucí.

C.1 Ze zdrojových kódů

Zdrojové kódy systému NEMEA lze získat buď z archivu na CD médiu, nebo z github repozitáře:

```
$ git clone --recursive https://github.com/CESNET/nemea
```

Pro kompilaci je potřeba mít nainstalované tyto balíky: `gcc`, `gcc-c++`, `make`, `autoconf`, `automake`, `libtool`, `pkg-config`, `libxml2-dev` a `libpcap-dev`. Systém lze poté zkompilovat pomocí následujících příkazů:

```
$ cd nemea
$ ./bootstrap.sh
$ ./configure
$ make -j 4
```

Exportér flow_meter se po úspěšném dokončení kompilace nachází ve složce *nemea/modules/flow_meter*. Pro instalaci systému stačí pustit následující příkaz:

```
$ sudo make install
```

C.2 OpenWrt

Systém NEMEA lze do OpenWrt distribuce dostat pomocí ipk balíku nebo přidáním do image s OpenWrt. Tento návod popisuje, jak vytvořit zmíněné ipk balíky a nainstalovat je na existující OpenWrt systém. Pro vytváření balíku je potřeba mít stažený OpenWrt build systém:

```
$ git clone https://git.openwrt.org/15.05/openwrt.git
$ cd openwrt
```

Dále je potřeba stáhnout a nainstalovat NEMEA feed, který obsahuje modul flow_meter a potřebné knihovny. Pro stažení je nutné vložit následující řádek do souboru *feeds.conf*.

```
src-git nemea https://github.com/CESNET/Nemea-OpenWRT
```

Feed lze nainstalovat pomocí:

```
$ ./scripts/feeds update nemea
$ ./scripts/feeds install -a -p nemea
```

S využitím utility *make menuconfig* je dále potřeba nastavit architekturu (menu *Target System/*) a cílové zařízení (menu *Target Profile/*). Na závěr v menu *Network/NEMEA/* nastavit `nemea-flow_meter` a `nemea-framework` na volbu *M*. Po nastavení je možné vytvořit balíky a zkopírovat je na zařízení s OpenWrt. Vytvořené ipk balíky se nachází ve složce *bin/<ZVOLENÁ-ARCHITEKTURA>/packages/nemea/*.

```
$ make menuconfig
$ make package/nemea-modules/{clean,compile} -j 4
$ scp bin/<ARCH>/packages/nemea/*.ipk root@<IP>:
```

Na cílovém zařízení je možné balíky nainstalovat pomocí utility *opkg*. Pokud by instalace selhala, je pravděpodobné, že chybí závislosti. Ty je možné doinstalovat pomocí *opkg update*.

```
$ opkg install nemea-*
$ opkg update
```

Nainstalovaný flow_meter se bude nacházet ve složce */usr/bin/nemea/*. Init script se nachází v */etc/init.d/flow_meter* a jeho konfigurační soubor v */etc/config/flow_meter*.

Příloha D

Použití modulu flow_meter

Tato kapitola popisuje jak spustit zkompilovaný exportér. Nejprve je potřeba specifikovat jeden z možných vstupů:

-I ROZHRANÍ čtení ze síťového rozhraní. Zadává se název rozhraní, například *eth0*. Vyžaduje práva root uživatele.

-r SOUBOR čtení ze souboru ve formátu pcap.

Druhým povinným parametrem je specifikace jednoho z výstupu exportéru.

-i TRAPSPEC export přes TRAP rozhraní do systému NEMEA. Rozhraní jsou ve formátu TYP:PARAMETRY. Typ může být buď UNIX rozhraní (u:NÁZEV), TCP rozhraní (t:PORT) nebo soubor (f:CESTA). Počet rozhraní je odvozen od počtu aktivních pluginů (`-p` přepínač) a jsou oddělena čárkou, například *u:muj_socket,f:/data/flows,t:10200*.

-x IP:PORT export na vzdálený IPFIX kolektor toků. Výchozí transportní protokol je TCP. Pomocí přepínače `-u` lze exportovat přes protokol UDP.

Nejdůležitější nepovinné přepínače:

-p PLUGINY aktivuje dané pluginy ve flow cache. Zadavají se názvy oddělené čárkou. Možné názvy jsou *basic,http,dns,sip,ntp,arp*. Pořadí pluginů koresponduje s pořadím výstupních rozhraní v `-i` parametru (pokud je specifikován).

-c ČÍSLO způsobí, že exportér se po přečtení daného počtu paketů ukončí.

-u způsobí použití UDP jako transportního protokolu pro export v IPFIX formátu.

-s POČET nastaví počet záznamů ve flow cache.

-t AKTIVNÍ:NEAKTIVNÍ nastaví aktivní a neaktivní časové limity.

-F FILTER nastaví BPF filter při čtení paketů. Provoz, který neprojde filtrem, je zahozen.

-l ČÍSLO nastaví maximální velikost zachyceného paketu.

-h zobrazí nápovědu.

Nápovědu k ostatním přepínačům lze zobrazit pomocí parametru `-h`.

Příloha E

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `src`
  - `nemea.tar.gz` ..... zdrojové kódy NEMEA s exportérem flow_meter
  - `thesis.tex` ..... zdrojová forma práce ve formátu LaTeX
  - `thesis.bib` ..... soubor s citacemi
- `text` ..... text práce
  - `thesis.pdf` ..... text práce ve formátu PDF
  - `thesis.ps` ..... text práce ve formátu PS