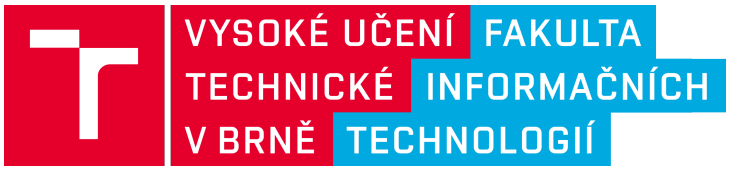


ISA – Projekt, varianta č. 1
Generování NetFlow dat ze zachycené síťové komunikace

Dominik Vágner,
xvagne10

11. listopadu 2022

Obsah

1 Úvod **2**

2 Problematika **2**
2.1 Co je to NetFlow? . . . 2
2.2 Zpracování zachycených paketů . . . 2
2.3 Správa Flows a jejich plnění . . . 3
2.4 Exportování NetFlow paketů . . . 3

3 Implementace **4**
3.1 Části programu . . . 4
3.1.1 Hlavní funkce programu . . . 4
3.1.2 Zpracování argumentů . . . 4
3.1.3 Analýza paketů . . . 4
3.1.4 Správa NetFlow záznamů . . . 4
3.1.5 Exportování NetFlow paketů . . . 4
3.2 Implementační detaily . . . 5
3.2.1 Data ve flows . . . 5
3.2.2 Časy . . . 5
3.3 Funkce a struktury . . . 5
3.4 Použité knihovny . . . 5

4 Testování **6**

5 Návod na použití **6**

6 Závěr **6**

Literatura **7**

1 Úvod

Zadáním projektu bylo navrhnout a implementovat NetFlow exportér, který bude zpracovávat klasické síťové pakety z PCAP souborů.

NetFlow exportér umožňuje exportovat pakety s protokoly komunikace `ICMP`, `TCP` a `UDP`. Vzhledem k tomu že náš exportér nezachycuje pakety ze živé sítě, ale čte je z už zachycených, tak musí také simulovat čas kdy byly zachycené. Exportér taky implicitně agreguje flows a na specifikovaný kolektor je odesílá po co nejvíce najednou. Pro exportování můžeme upřesnit chování expirování flows pomocí argumentů programu.

Výstupem programu jsou pouze NetFlow pakety odesílané na adresu NetFlow kolektoru.

2 Problematika

Problematiku projektu můžeme rozdělit na 4 hlavní části. První je otázka „Co je to NetFlow?“ Druhou je čtení a zpracování zachycených paketů ze souboru a manipulace s nimi. Třetí je správa/plnění jednolivých NetFlow záznamů a jejich expirování. Následnou poslední částí je samotné odesílání NetFlow paketů na kolektor.

2.1 Co je to NetFlow?

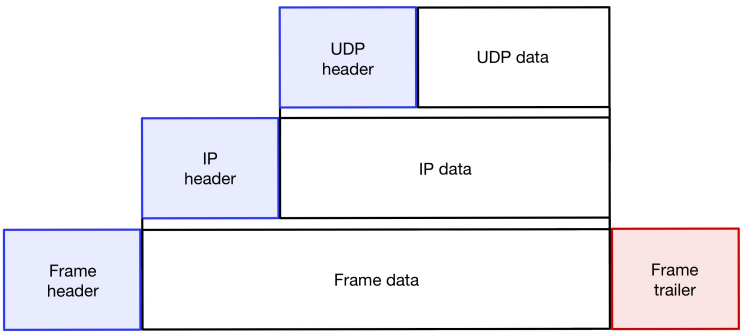
NetFlow je protokol který slouží k monitorování síťového toku pro pochopení patternů sítě a distribuci protokolů. Možnost vidět do sítě je klíčová pro údržbu a bezpečnost. Toho dosahuje tak že balí pakety do takzvaných flows. Flow je skupina paketů, které jsou součástí jedné a té samé konverzace mezi dvěma koncovými body [1].
Netflow komunikace se skládá z různých částí [2]:

- Exportér – sonda/router pro získávání statistik o tocích
- Komunikační protokol NetFlow - např. NetFlow v5, NetFlow v9
- Kolektor – zařízení pro ukládání záznamů o tocích, např. Nfdump/Nfcapd
- Nástroje pro zobrazení dat – grafy, statistiky, apod., např. NfSen

2.2 Zpracování zachycených paketů

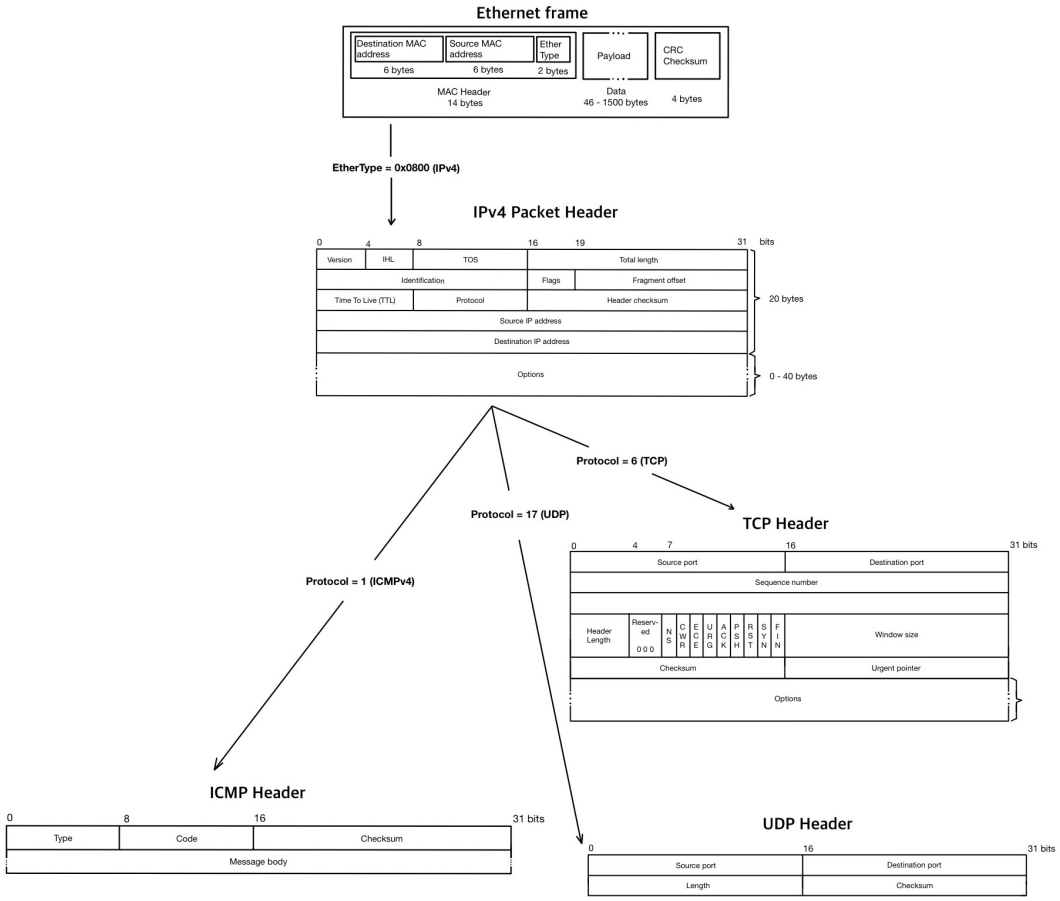
Před zahájením čtení musíme nastavit filtr na námi podporované protokoly komunikace za pomocí `libpcap` knihovny a jejích možností filtrování [3].

Při zpracovávání informací z jednotlivých paketů narážíme na zapouzdřování datagramů. Zachycený ethernetový rámec má za hlavičkou payload s typem dat určeným v hlavičce. Podobně je to tak i s protokoly na síťové a transportní vrstvě.



Obrázek 1: Příklad zapouzdření ukázán na UDP datagramu [4].

Postup pro určení správné hlavičky protokolu (z námi podporovaných) je vyobrazen obrázkem č. 2



Obrázek 2: Diagram postupu určování protokolu [5–10]

.

2.3 Správa Flows a jejich plnění

Extrahované informace ze síťové komunikace ukládáme do NetFlow záznamů (používáme verzi 5) ve specifikovaném formátu [11]. NetFlow záznamy mohou obsahovat více než jeden síťový paket, to zajistíme agregací daných paketů podle 5-tice údajů jenž spojují dané pakety. Pro námi implementovanou verzi NetFlow protokolu je typická pětice `{Zdrojová adresa, Cílová adresa, Zdrojový port, Cílový port, Číslo identifikující protokol [10]}`. Za pomocí tohoto klíče vytvořeného pro každý přečtený paket se rozhodujeme zda vytvořit nový NetFlow záznam nebo zda stačí aktualizovat už vytvořený a to tak, že vyhledáme zda už máme k danému klíči záznam v naší flow cache. Flow cache je místo kde si uchováváme dvojice klíč, záznam.

Před zpracováním každého nového paketu zkontrolujeme zda žádný z našich záznamů ve flow cache, by neměl být expirován a přesunut z flow cache do fronty k odeslání.

2.4 Exportování NetFlow paketů

K odesílání netflow paketů se používá UDP komunikační protokolPo každé kontrole a případném vložení nového NetFlow záznamu, se podíváme zda nemáme nějaké expirované flows ve frontě k odeslání a pokud ano, tak je odešleme na kolektor v NetFlow paketu. Do jednoho NetFlow paketu můžeme vložit maximálně 30 záznamů. Před které musíme také naplnit a vložit NetFlow paket hlavičku.

3 Implementace

Projekt byl implementován v jazyce C++. Je složen ze dvou souborů, zdrojového souboru `flow.c` s implementacemi funkcí ze kterých se program skládá a hlavičkového souboru `flow.h` obsahující seznam použitých knihoven, definice vlastních struktur, deklarace funkcí a komentáře pro dané funkce. Zdroje ze kterých byly získány informace o tom jak implementovat některé části tohoto projektu jsou ozdrojovány v kódu.

3.1 Části programu

3.1.1 Hlavní funkce programu

Ve vstupní/hlavní funkci programu `main()` se nejdříve zavolá funkce `parse_arguments()` pro zpracování argumentů. Poté nastavíme filtr na námi podporované protokoly komunikace a inicializujeme připojení na kolektor přřes socket. Nakonec zahájíme zahájíme čtení paketů ze souboru. Každý zachycený paket pošleme do funkce pro analýzu paketů `callback_handler()`. Před výstupem z funkce/programu nezapomeneme exportovat všechny neexpirované flow z flow cache v průběhu čtení paketů a uzavřeme socket.

3.1.2 Zpracování argumentů

Pro argumenty máme v programu vytvořenou strukturu `arg_values` s příznaky pro všechny argumenty a jejich hodnoty. Ve funkci pro zpracování argumentů `parse_arguments()` tuto strukturu naplníme pomocí knihovny `getopt` [12] a následně ji vrátíme. Tento výsledek si poté uložíme pro budoucí přístup k hodnotám argumentů.

3.1.3 Analýza paketů

Ve funkci pro analýzu paketů postupujeme podle obrázku č. 2. Paket dostaneme pouze jako pole unsigned charakterů a od začátku poté zjišťujeme jaké hlavičky protokolů jsou za sebou. Jako první máme vždy ethernet frame. Poté postupujeme pomocí informací v daných protokolech o následujících headerů. V průběhu parsování hlaviček si nám užitečné informace ukládáme pro budoucí použití. Po projití celého paketu, ale stále ve stejné funkci následuje sekce pro interakcí s flow cache.

3.1.4 Správa NetFlow záznamů

Jednotlivé NetFlow záznamy (flows) si uchováváme ve flow cache, což je struktura obsahující dvojice hodnot klíč (5-tice identifikující flow) a hodnota (flow). Flow cache je implemntována pomocí `std::map`.

Po analýze paketu zavoláme funkci `expire_checks()` která zkontroluje aktivní a neaktivní intervaly po kterou si necháváme flow ve flow cache, je-li už interval překročen, flow expirujeme a přesuneme do fronty k exportu. Dále také v této funkci u flow s komunikačním protokolem `TCP` zda jsem neregistrovali paket s `RST` nebo `FIN` příznakem, pokud ano tak je flow také expirována.

Následně si z extrahovaných informací z analyzovaného paketu složíme klíč který se pokusíme najít ve flow cache a podle výsledku buď voláme funkci pro aktualizování nebo vytvoření nové flow. Při vytváření nové flow také kontrolujeme zda jsme nepřekročili její maximální velikost specifikovanou argumentem programu.

Po vložení nové flow zavoláme znovu funkci pro kontrolu expirace a poté funkci která z fronty flows pro exportování všechny odešle na kolektor.

3.1.5 Exportování NetFlow paketů

Data jsou na kolektor posílána pomocí UDP komunikačního protokolu. Daty jsou myšleny NetFlow pakety skládající se z jedné hlavičky a proměnlivého počtu NetFlow záznamů (1 až 30). Po expiraci si flow přesuneme z flow cache do fronty pro export a tuto frontu pak procházíme ve funkci `send_udp()`

3.2 Implementační detaily

3.2.1 Data ve flows

Flow je definována strukturou podle specifikovaného formátu a naplníme ji všemi námi zjistitelnými daty, ty které zjistit nemůžeme nastavujeme na 0. Při `ICMP` protokolu dáváme namísto cílového portu za sebou složené údaje z `ICMP` hlavičky (`ICMP Type` a `ICMP Code` [6]) respektive. Počet bytů, který vkládáme nebo přičítáme do flow (`dOctets` položka), ze zpracovaných paketů počítáme tak že vezmeme délku z hlavičky paketu a odečteme velikost hlavičky ethernetového paketu.

3.2.2 Časy

Při našem řešení kdy čteme už dříve zachycenou síťovou komunikace tak musíme simulovat čas, což může být poněkud zmatečné, protože většina exportéru odchytává komunikaci v reálném čase ze síťové komunikace. Čas simulujeme tak, že časové razítko z prvního přečteného paketu ze síťové komunikace považujeme za dobu kdy se náš exportér zapl. Od tohoto času poté pomocí funkce `timeval_sub_ms()` počítáme rozdíl v milisekundách od posledně zpracovaného paketu. Tento údaj je velmi důležitý, protože od něj podle specifikovaného formátu hlavičky a záznamu se poté řadí všechny záznamy (ve specifikaci jde o položku „SysUptime“).

3.3 Funkce a struktury

- `arg_values` – Struktura obsahující všechny detaily o argumentech programu.
- `netflow_header` – Struktura reprezentující hlavičku NetFlow paketu, vytvořena podle specifikace [11].
- `netflow_record` – Struktura reprezentující NetFlow záznam (flow), vytvořena podle specifikace [11].
- `comp_oldest` – Pomocná struktura se vzorovým typem pro řazení flow cache podle času v záznamech.

- `parse_arguments()` – Funkce pro zpracování argumentů programu za pomocí `getopt` knihovny.
- `callback_handler()` – Funkce pro analýzu a zpracování přečteného paketu.
- `insert_flow()` – Funkce pro vložení nového záznamu do flow cache.
- `update_flow()` – Funkce pro aktualizování už existující flow ve flow cache.
- `expire_checks()` – Funkce pro kontrolu zda nějaká flow ve flow cache může být označena za expirovanou a přesunuta do fronty flows k exportu.
- `init_udp()` – Funkce pro inicializaci spojení k adrese NetFlow kolektoru..
- `send_udp()` – Funkce pro export NetFlow paketů na kolektor z fronty expirovaných NetFlow záznamů.
- `timeval_sub_ms()` – Funkce pro vypočítání rozdílu v milisekundách mezi dvěma `timeval` strukturami.
- `comp_oldest_vec()` – Pomocná funkce pro řazení vektoru NetFlow záznamů podle času v záznamech.

3.4 Použité knihovny

- `pcap.h` – Hlavní knihovna pro čtení paketů ze souboru a jejich filtrování.
- `arpa/inet.h` – Funkce pro obracení endianness unsigned short a long proměnných.
- `netinet/if_ether.h` – Struktura pro ethernet header a makra k ní vázané.
- `netinet/tcp.h` – Struktura pro TCP header a makra k ní vázané.
- `netinet/udp.h` – Struktura pro UDP header a makra k ní vázané.
- `netinet/ip_icmp.h` – Struktura pro ICMPv4 header a makra k ní vázané.
- `netinet/ip.h` – Struktura pro IPv4 header a makra k ní vázané.
- `iostream`, `iomanip` – C++ knihovny pro výpis pomocí streamů a jejich formátování.
- `getopt.h` – Funkce k zpracování argumentů programu.

- `stdio.h` – Pro `printf()` funkci.
- `string.h` – Přidání string typu.
- `signal.h` – Zpracování interrupt signalů pro korektní ukončení.
- `unistd.h` – Funkce pro práci se sockety.
- `err.h` – Pro makra ukončující program se správnou hláškou.
- `map` – Třída reprezentující strukturu podobnou hashmapě která je použita na implementaci flow cache.
- `tuple` – C++ implementace pro n-tici, pomcí které je realizován klíč flow cache.
- `set`, `algorithm` – Knihovny používané pro správné řazení (od nejstaršího) flow cache a fronty pro export.
- `vector` – Použito pro realizování fronty/pole flows k exportu.

4 Testování

Testování našeho exportéru a porovnávat výsledky dalšími je dosti problematické, jelikož každý implementuje některé části odlišně a exportované údaje se tedy mohou lišit. Výsledný výstup mého exportéru jsem se snažil co nejvíce přiblížit NetFlow analyzátoru a exportéru `softflowd`[1].

Při testování byly hojně využívány nástroje ze sady nástrojů pro práci s NetFlow `nfdump`[2], hlavně z ní byl použit referenční NetFlow kolektor a nástroj pro zobrazení zachycených dat.

5 Návod na použití

Před spuštěním našeho exportéru musíme spustit nějaký NetFlow kolektor na jehož adresu budeme odesílat NetFlow pakety. Exportér můžeme spustit s těmito argumenty:
`./flow [-f <file>] [-c <netflow_collector>[:<port>]] [-a <active_timer>] [-i <inactive_timer>] [-m <count>]`
Například tedy: `./flow -f capture.pcap -i 15 -m 256`
Pro více informací o rozběhnutí projektu, jeho spuštění a argumentech si můžeme zobrazit nápovědu programu nebo manuál pomocí následujících příkazů: „`./flow -h`“ nebo „`man -l flow.1`“

6 Závěr

V projektu se mi povedlo implementovat vše požadované, a hodně jsem se při něm naučil, zejména jak funguje celý NetFlow ekosystém. Vyzkoušet si tvorbu linuxového manuálu jsem také shledal dobrou zkušeností. Projekt mi přišel jako jeden z těch zajímavějších tento semestr, i když to byl jeden z těch náročnějších, kvůli různým odlišnostem mezi existujícími NetFlow exportéry.

[1] `softflowd`: `https://github.com/irino/softflowd`
[2] `nfdump`: `https://github.com/phaag/nfdump`

Literatura

[1] Wikipedia. User Datagram Protocol — Wikipedia, the free encyclopedia. https://en.wikipedia.org/wiki/NetFlow, 2022. [Online; accessed 06-November-2022].

[2] Petr. Matoušek. ISA: Monitorování toků NetFlow. *[Univerzitní přednáška]*, 2021.

[3] Tim Carstens. Programming with pcap. https://www.tcpdump.org/pcap.html, 2002. [Online; accessed 06-November-2022].

[4] V. Veselý. Transportní vrstva. *[Univerzitní přednáška]*, 2022.

[5] Wikipedia. Ethernet frame — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Ethernet%20frame&oldid=1080405638, 2022. [Online; accessed 06-November-2022].

[6] Wikipedia. Internet Control Message Protocol — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Internet%20Control%20Message%20Protocol&oldid=1081447774, 2022. [Online; accessed 06-November-2022].

[7] Wikipedia. IPv4 — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=IPv4&oldid=1081194704, 2022. [Online; accessed 06-November-2022].

[8] Wikipedia. Transmission Control Protocol — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=Transmission%20Control%20Protocol&oldid=1083491738, 2022. [Online; accessed 06-November-2022].

[9] Wikipedia. User Datagram Protocol — Wikipedia, the free encyclopedia. http://en.wikipedia.org/w/index.php?title=User%20Datagram%20Protocol&oldid=1079061202, 2022. [Online; accessed 06-November-2022].

[10] Internet Assigned Numbers Authority. Protocol numbers. https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml, Apr 2021. [Online; accessed 06-November-2022].

[11] Cisco. Netflow export datagram format. https://www.cisco.com/c/en/us/td/docs/net_mgmt/netflow_collection_engine/3-6/user/guide/format.html#wp1003394, Mar 2015. [Online; accessed 06-November-2022].

[12] Lei Mao. Parsing argument using getopt in c/c++. https://leimao.github.io/blog/Argument-Parser-Getopt-C/, 2019. [Online; accessed 06-November-2022].