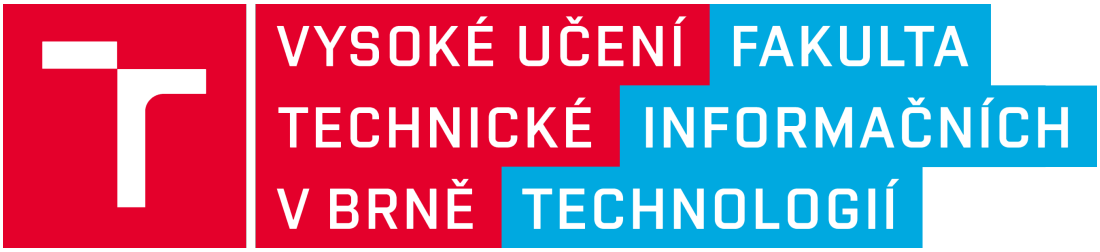


Dokumentace k projektu

PCAP Netflow v5 exportér

Jakub Gryc `xgrycj03`

16. listopadu 2024

Obsah

1 Úvod

Cílem projektu bylo vytvořit aplikaci, která bude schopna zpracovávat soubory ve formátu PCAP a na základě těchto souborů generovat Netflow záznamy ve verzi 5. Tyto záznamy jsou následně posílány pomocí transportního protokolu UDP na kolektor. Aplikace je napsána v jazyce C++ a využívá knihovnu `libpcap` pro zpracování PCAP souborů.

2 Návrh a implementace

Aplikace je rozdělena do několika tříd a souborů, které vzájemně spolupracují. Veškeré zdrojové soubory jsou umístěny v adresáři `src/`. Některé důležité třídy a jejich metody jsou popsány v následujících podkapitolách.

2.1 Vstupní bod programu, main.cpp

Soubor `main.cpp` obsahuje vstupní bod programu. V této části se zpracovávají argumenty a inicializují se hlavní třídy programu. Jedna z těchto tříd je třída Timer, která si po své inicializaci uloží časovou značku, která je nadále využívána pro výpočet relativního času v Netflow záznamech. Dále se zde inicializuje třída UDPConnection, která obstarává posílaní UDP zpráv na kolektor. Poslední třídou, která je zde inicializována, je třída PcapHandler, která slouží jako hlavní jádro celé aplikace, jejíž účel je popsán v další podkapitole.

2.2 Zpracování paketů - PcapHandler

Hlavní jádro programu se nachází ve třídě PcapHandler. Po své inicializaci otevře PCAP soubor a postupně zpracovává jednotlivé pakety. Metoda `PcapHandler::start()` inicializuje instanci třídy FlowCache, která slouží jako rozhraní pro ukládání a manipulování s uloženými Netflow záznamy. Následně se iteruje přes každý paket v PCAP souboru a volá se metoda `PcapHandler::processPacket()`, která zpracovává jednotlivé pakety a relevantní informace z nich ukládá do FlowCache. Po zpracování všech paketů se zavolá metoda `PcapHandler::sendFlows()`, která postupně prochází všechny uložené Netflow záznamy a tyto záznamy odesílá na kolektor pomocí třídy UDPConnection. Pokud vypršel aktivní či neaktivní časový limit pro 30 a více záznamů, tak se tyto záznamy po 30 mohou na kolektor odeslat ještě dříve, než se zpracují všechny pakety v PCAP souboru. Tato funkcionalita je více popsána později viz 2.3.

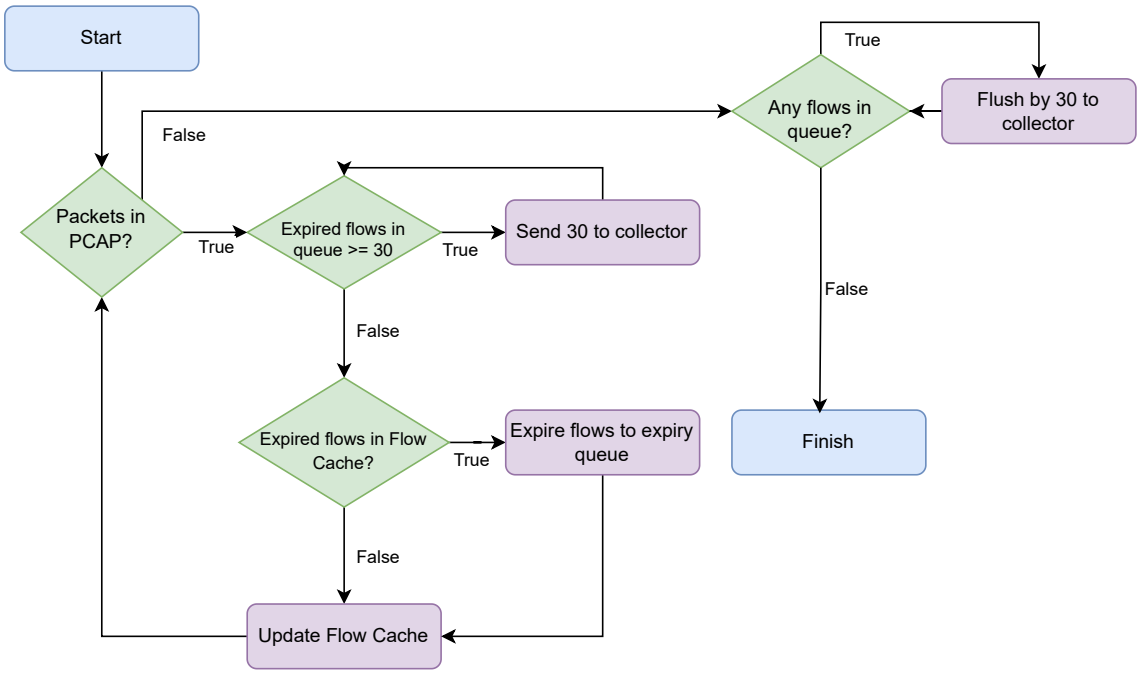
2.3 Zpracování netflow záznamů - FlowCache

Tato třída uchovává jednotlivé záznamy ve formě hash tabulky. Každý záznam je identifikován pomocí klíče, který je složen ze zdrojové IP adresy a portu a cílové IP adresy a portu. Transportní protokol se neukládá, jelikož aplikace řeší jen záznamy o TCP tocích. Hodnoty v této tabulce pak tvoří struktura Flow, která obsahuje informace jako počet bytů, počet paketů, čas prvního a posledního záznamu a další. Třídy FlowCache a Flow obsahují metody pro přidání nového záznamu, aktualizaci stávajícího záznamu či odeslání všech záznamů na kolektor.

Časové limity

Důležitou součástí je hlídání časových limitů pro jednotlivé záznamy. Před každou aktualizací FlowCache se kontroluje, zda jednotlivým záznamům nevypršel časový limit. Veškeré expirované záznamy se odstraní z tabulky a přiřadí do fronty `exportCache`. Do fronty se záznamy přidávají vždy podle toho, jak dlouho je záznam expirovaný. Nejdéle expirované záznamy se do fronty přidávají jako první, jak je popsáno v [2], sekce V-C, aby byly poslány dříve. Funkcionalitu zajišťuje metoda `FlowCache::checkForExpiredFlows()`.

Exportované záznamy se už přímo ukládají do struktury, která se pošle na kolektor. Je zajištěno správné pořadí bytů a tedy jejich konverze do síťového bytového pořadí.



Obrázek 1: Diagram zpracování paketů a odesílání Netflow záznamů mého exportéru.

Díky tomuto přístupu jsou záznamy odesílány na kolektor seřazené vzestupně podle časové značku prvního paketu daného flowu, jak lze vidět na obrázku 2.

```
❯ nfdump -r nfcapd.20241116003630
Date first seen          Event  XEvent Proto      Src IP Addr:Port          Dst IP Addr:Port     X-Src IP Addr:Port       X-Dst IP Addr:Port   In Byte Out Byte
2024-10-13 23:30:21.430 INVALID  Ignore TCP     147.229.192.94:43634 ->   100.64.200.182:1716          0.0.0.0:0     ->          0.0.0.0:0         104        0
2024-10-13 23:30:21.432 INVALID  Ignore TCP     100.64.200.182:1716  ->   147.229.192.94:43634          0.0.0.0:0     ->          0.0.0.0:0         104        0
2024-10-13 23:30:21.841 INVALID  Ignore TCP     100.64.221.182:1716  ->   147.229.192.94:57796          0.0.0.0:0     ->          0.0.0.0:0         156        0
2024-10-13 23:30:21.841 INVALID  Ignore TCP     147.229.192.94:57796 ->   100.64.221.182:1716           0.0.0.0:0     ->          0.0.0.0:0         156        0
2024-10-13 23:30:21.941 INVALID  Ignore TCP     147.229.192.94:55652 ->    100.64.207.70:1716           0.0.0.0:0     ->          0.0.0.0:0         104        0
2024-10-13 23:30:22.124 INVALID  Ignore TCP      100.64.207.70:1716  ->   147.229.192.94:55652          0.0.0.0:0     ->          0.0.0.0:0         104        0
2024-10-13 23:30:22.601 INVALID  Ignore TCP     147.229.192.94:46704 ->  162.159.130.234:443            0.0.0.0:0     ->          0.0.0.0:0        1716        0
2024-10-13 23:30:22.601 INVALID  Ignore TCP    162.159.130.234:443   ->   147.229.192.94:46704          0.0.0.0:0     ->          0.0.0.0:0        6206        0
2024-10-13 23:30:22.966 INVALID  Ignore TCP      100.64.212.21:1716  ->    100.64.201.28:33806          0.0.0.0:0     ->          0.0.0.0:0         156        0
2024-10-13 23:30:26.548 INVALID  Ignore TCP     147.229.192.94:36366 ->   100.64.208.240:1716           0.0.0.0:0     ->          0.0.0.0:0          52        0
2024-10-13 23:30:26.560 INVALID  Ignore TCP     100.64.208.240:1716  ->   147.229.192.94:36366          0.0.0.0:0     ->          0.0.0.0:0          52        0
2024-10-13 23:30:27.061 INVALID  Ignore TCP     147.229.192.94:42650 ->    100.64.199.83:1716           0.0.0.0:0     ->          0.0.0.0:0          52        0
2024-10-13 23:30:27.151 INVALID  Ignore TCP      100.64.199.83:1716  ->   147.229.192.94:42650          0.0.0.0:0     ->          0.0.0.0:0          52        0
2024-10-13 23:30:27.911 INVALID  Ignore TCP      100.64.212.21:1716  ->   100.64.193.202:59896          0.0.0.0:0     ->          0.0.0.0:0         104        0
2024-10-13 23:30:30.645 INVALID  Ignore TCP     147.229.192.94:33620 ->   100.64.198.247:1716           0.0.0.0:0     ->          0.0.0.0:0          52        0
2024-10-13 23:30:30.645 INVALID  Ignore TCP     147.229.192.94:38472 ->   100.64.201.132:1716           0.0.0.0:0     ->          0.0.0.0:0          52        0
2024-10-13 23:30:30.647 INVALID  Ignore TCP     100.64.201.132:1716  ->   147.229.192.94:38472          0.0.0.0:0     ->          0.0.0.0:0          52        0
2024-10-13 23:30:30.688 INVALID  Ignore TCP     100.64.198.247:1716  ->   147.229.192.94:33620          0.0.0.0:0     ->          0.0.0.0:0          52        0
2024-10-13 23:30:31.157 INVALID  Ignore TCP     147.229.192.94:44804 ->   100.64.192.185:1716           0.0.0.0:0     ->          0.0.0.0:0          52        0
2024-10-13 23:30:31.268 INVALID  Ignore TCP     100.64.192.185:1716  ->   147.229.192.94:44804          0.0.0.0:0     ->          0.0.0.0:0          52        0
2024-10-13 23:30:31.287 INVALID  Ignore TCP     147.229.192.94:50706 ->       52.1.53.78:443            0.0.0.0:0     ->          0.0.0.0:0         435        0
2024-10-13 23:30:31.369 INVALID  Ignore TCP         52.1.53.78:443   ->   147.229.192.94:50706          0.0.0.0:0     ->          0.0.0.0:0         509        0
2024-10-13 23:30:33.501 INVALID  Ignore TCP     147.229.192.94:38262 ->      52.26.161.5:443            0.0.0.0:0     ->          0.0.0.0:0        2482        0
2024-10-13 23:30:33.689 INVALID  Ignore TCP        52.26.161.5:443   ->   147.229.192.94:38262          0.0.0.0:0     ->          0.0.0.0:0        3741        0
Summary: total flows: 24, total bytes: 16597, total packets: 127, avg bps: 8818, avg pps: 8, avg bpp: 130
Time window: 2024-10-13 23:30:21 - 2024-10-13 23:30:36
Total flows processed: 24, passed: 24, Blocks skipped: 0, Bytes read: 2080
Sys: 0.0048s User: 0.0010s Wall: 0.0016s flows/second: 15055.8 Runtime: 0.0016s
```

Obrázek 2: Ukázka seřazených záznamů pomocí nástroje nfdump. Použit byl soubor `tests/pcaps/test8.pcap`.

2.4 UDP spojení

Třída UDPConnection slouží k odesílání Netflow záznamů na kolektor. Po své inicializaci se vytvoří socket, který je následně využíván pro odesílání záznamů. Metoda `UDPConnection::sendFlows()` odesílá po-

stupně jednotlivé záznamy na kolektor v Netflow v5 formátu [1]. Pokud ještě nedošlo ke zpracování všech paketů ze souboru, tak se záznamy odesílají jen po 30 záznamech, které byly již exportovány do `exportCache`. Pokud jich je méně, tak se záznamy neposílají a čeká se na naplnění fronty na alespoň 30 záznamů. Pokud se zpracovaly všechny pakety, tak se záznamy posílají všechny, i když jich je méně než 30, postupně po 30.

3 Návod k použití

Program je možné sestavit pomocí příkazu `make` v domovském adresáři projektu. Po sestavení je možné program spustit s následujícími argumenty:

```
./p2nprobe <host>:<port> <pcap_file> [-a <active_timeout>] [-i
<inactive_timeout>]
```

- `<host>:<port>` - povinný argument, který určuje IP adresu a port, na kterých kolektor běží.
- `<pcap_file>` - povinný argument, který určuje cestu k PCAP souboru, který se analyzuje.
- `-a <active_timeout>` - nepovinný argument, který určuje časový limit pro aktivní timeout. Výchozí hodnota je 60 sekund.
- `-i <incative_timeout>` - nepovinný argument, který určuje časový limit pro neaktivní timeout. Výchozí hodnota je 60 sekund.

4 Testování

Pro korektní testování bylo nutné obstarat si několik PCAP souborů. Pro testování základní funkčnosti exportéru jsem si vytvořil dva jednoduché programy v Pythonu, jeden sloužil jako TCP klient a druhý jako TCP server. Tyto programy jsem spustil a zachytil jejich komunikaci pomocí nástroje Wireshark, kterou jsem ukládal do PCAP souborů. Jejich implementace je dostupná v adresáři `tests/`, PCAP soubory v `tests/pcaps`. Implementoval jsem taktéž jednoduchý testovací skript `tests/test.py` v Pythonu, který obsahuje mnou implemntovaný kolektor. Tento skript pouští můj exportér a kolektor, který následně zpracovává přijaté záznamy a ukládá je do logů v adresáří `tests/logs`. Takto jsem byl schopen jednoduše ověřovat obsah záznamů, které exportér posílá.

Logy z kolektoru jsou ve formátu JSON a obsahují dvě pole, `headers` a `records`. Pole `received` obsahuje hlavičky záznamů, které kolektor přijal, a pole `records` obsahuje samotné záznamy. Záznamy rovněž obsahují zpracovanou časovou značku, kdy byl záznam odeslán a délku trvání flowu, tyto hodnoty exportér neodesílá.

Listing 1: Ukázka logu z kolektoru

```
{
    "headers": {
        "header_0": {
            "Version": 5,
            "Count": 2,
            "SysUptime": 0,
            "UnixSecs": 1731704591,
            "UnixNsecs": 808015000,
            "FlowSequence": 0,
            "EngineType": 0,
            "EngineID": 0,
            "SamplingInterval": 0
```

```
        }
    },
    "records": {
        "0184c1408f81a885a9bbdb50fc88065a": {
            "Timestamp": "2024-10-14 00:38:17.370",
            "SrcAddr": "127.0.0.1",
            "DstAddr": "127.0.0.1",
            "NextHop": "0.0.0.0",
            "Input": 0,
            "Output": 0,
            "Packets": 4,
            "Octets": 216,
            "First": 1449472857,
            "Last": 1449491651,
            "Duration": 18794,
            "SrcPort": 9119,
            "DstPort": 14002,
            "Padding": 0,
            "TCPFlags": 19,
            "Protocol": 6,
            "Tos": 0,
            "SrcAS": 0,
            "DstAS": 0,
            "SrcMask": 0,
            "DstMask": 0,
            "Padding2": 0
        },
        ...
    }
}
```


Program rovněž obsahuje několik testů, které pouští jak můj exportér, tak referenční exportér `softflowd` a porovnává jejich výstupy. Tyto testy lze spustit pomocí příkazu `python test.py -c -r`. Vzhledem k tomu, že referenční exportér má jinou implementaci, není možné jej plnohodně porovnávat s naším zadáním. Softflowd například nedokáže měnit časové limity, když načítá pakety ze souboru, taktéž řeší TCP fin a reset flagy. To v našem našem exportéru není zadáno. Testy tedy spíš slouží k porovnání jen se soubory, které obsahují malé množství záznamů, kdy se výstupy téměř vždy shodují.

Reference

[1] IBM: NetFlow V5 formats. [online], 2022, [cit. 2024-11-05].
URL `https://www.ibm.com/docs/en/npi/1.3.1?topic=versions-netflow-v5-formats`

[2] Vormayr, G.; Fabini, J.; Zseby, T.: Why are My Flows Different? A Tutorial on Flow Exporters. *IEEE Communications Surveys & Tutorials*, ročník 22, č. 3, 2020: s. 2064–2103, doi:10.1109/COMST.2020.2989695.