VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta informačních technologií

ISA - Generovanie NetFlow dát zo zachytenej sieťovej komunikácie

2022/2023

Kristián Kováč
xkovac61

14. novembra 2022

Obsah

1 Zadanie

Zadaním projektu bolo vytvoriť NetFlow exportér, ktorý zo zachytených sieťových dát vo formáte pcap vytvorý NetFlow záznamy, ktoré následne odošle na kolektor.

1.1 Spúštanie programu

```
./flow [-f <file>] [-c <host>[:<port>]] [-a <active_timer>]
       [-i <inactive_timer>] [-m <count>]
```

- `-h | --help` - Výpis nápovedy.
- `-f <file>` - Meno analyzovaného súboru. (`stdin`, keď nie je inak špecifikované)
- `-c <host:port>` - IP adresa alebo hostname a voliteľne aj UDP port NetFlow kolektoru. (127.0.0.1:2055, keď nie je inak špecifikované)
- `-a <active_timeout>` - interval v sekundách, po ktorom sa exportujú aktívne záznamy na kolektor. (60, keď nie je inak špecifikované)
- `-i <inactive_timeout>` - interval v sekundách, po ktorom sa exportujú neaktívne záznamy na kolektor. (10, keď nie je inak špecifikované)
- `-m <count>` - veľkosť flow-cache. Pri dosiahnutí maximálnej veľkosti dôjde k exportu najstaršieho záznamu na kolektor (1024, keď nie je inak špecifikované)

2 NetFlow

NetFlow je protokol vyvinutý soločnosťou Cisco Systems, určený pôvodne ako doplnková služba k Cisco smerovačom. Jeho hlavným účelom je monitorovanie sieťovej prevádzky na základe IP tokov, ktorý v reálnom čase poskytuje administrátorom podrobný prehľad prevádzky na ich sieti.[1]

2.1 NetFlow exportér

NetFlow exportér je pripojený k monitorovanej linke a analyzuje prechádzajúce pakety. Na základe zachytených IP tokov generuje NetFlow štatistiky a tie exportuje na NetFlow kolektor.[1]

2.2 NetFlow kolektor

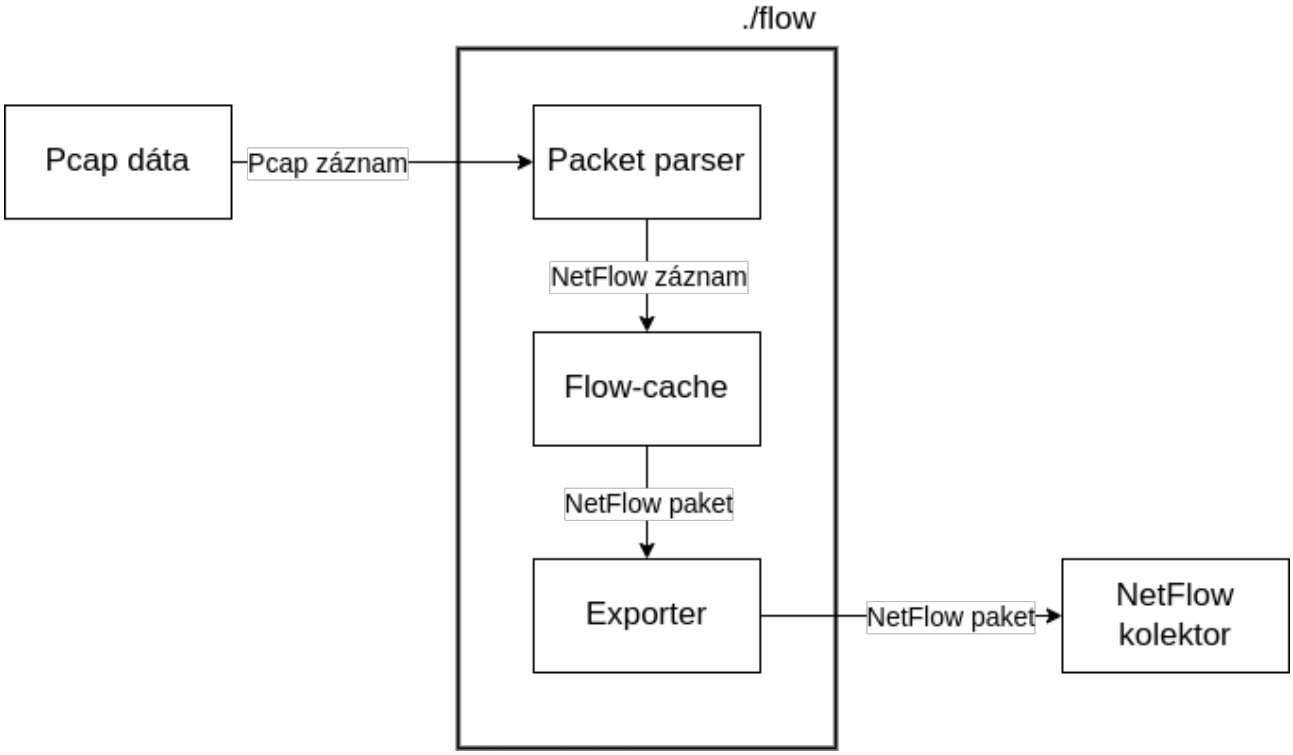
NetFlow kolektor je typicky zariadenie s veľkou úložnou kapacitou, ktoré zbiera štatistiky z väčšieho počtu NetFlow exportérov a ukladá ich do dlhodobej databázy. Tieto dáta sú potom ďalej spracovávané administrátormi na zistenie prípadných závad, úzkych miest, zdrojov vysokej prevádzky alebo sledovanie jednotlivých komunikácií.[1]

2.3 IP tok

Na základe IP tokov sú generované NetFlow štatistiky. Tok je definovaný ako sekvencia paketov so zhodnou päticou údajov: zdrojová/cieľová IP adresa, zdrojový/cieľový port a číslo protokolu. Pre každý tok je zaznamenávaná doba jeho vzniku, dĺžka trvania, počet prenesených paketov a ďašie údaje.[1]

3 Architektúra

Program sa skladá z 3 hlavných častí: `PacketParser`, `FlowCache` a `Exporter` cez ktoré postupne prechádzajú všetky dáta. Zachytené dáta sú načítavané v module `PacketParser`, ktorý ich následne predáva modulu `FlowCache`. Z `FlowCache` sú potom postupne, podľa definovaných pravidiel, záznamy exportované na NetFlow kolektor pomocou UDP paketov.



4 Implementácia

Aplikácia je implementovná pre NetFlow v5 v programovacom jazyku C++ s použitím knižnice libpcap.[2] Program je rozdelený do 4 logických modulov (`ArgumentParser`, `PacketParser`, `FlowCache` a `Exporter`). Všetky tieto moduly sú inicializované v `main.cpp`.

4.1 Spracovanie argumentov(`ArgumentParser.cpp`)

Spracovanie a validáciu argumentov zaisťuje trieda `ArgumentParser`, ktorá je implementovaná s použitím knižnice getopt[3]. Pre spracovanie dlhých argumentov sa využíva funkcia `getopt_long()`.

4.2 Filtrovanie paketov

Na filtrovanie paketov sa využíva `pcap-filter`[4], ktorý je súčasťou knižnice libpcap. Ten definuje syntax filtrovacieho reťazca, ktorý sa následne skompiluje a aplikuje pri načítavaní paketov. V našom prípade sa jedná o filter, ktorý odfiltruje všetky pakety, ktoré sú iného protokolu než TCP, UDP alebo ICMP.

4.3 Spracovanie paketov z pcap dát(`PacketParser.cpp`)

Po otvorení nášho pcap súboru a nastavení správnych filtrov sa začnú spracovávať jednotlivé pakety. Pre každý paket sa najprv zistí, akého je protokolu a následne sa pridáva do flow-cache. Po načítaní

všetkých paketov sa dodatočne exportujú pakety, ktoré zostali vo flow-cache.

4.4 Flow-cache(`FlowCache.cpp`)

Flow-cache je implementovaý pomocou štruktúry `std::vector<NF5Record>`. Pred vložením každého záznamu sa najprv overí, či sa záznam ešte zmestí do flow-cache. Keď nie, tak sa exportuje najstarší záznam vo flow-cache. V ďalšom kroku sa kontrolujú časové limity jednotlivých záznamov. Ak je nejaký záznam dlhšie neaktívny alebo už dostatočne dlho aktívny, tak sa exportuje na kolektor. Keď túto podmienku počas jednej kontroly spĺňa viac záznamov, tak sú združené do paketov po maximálne tridsiatich záznamoch.

Po všetkých kontrolách sa podľa definovanej pätice overí, či vkladaný záznam už nie je vo flow-cache. Keď je, tak sa iba aktualizuje už existujúci záznam, keď ešte nie je, tak sa vo flow-cache vytvorí nový záznam.

Pri aktualizácii je tiež nutné overiť, či spracovávaný paket nie je protokolu TCP s príznakom RST alebo FIN. V takom prípade môžeme predpokladať, že sa komunikácia skončila a záznam sa ihneď exportuje. Kľúčovým krokom je správna konverzia medzi rôznymi endianitami dát pri načítavaní a exportovaní záznamov.

4.5 Exportér(`Exporter.cpp`)

Exportér na začiatku behu programu vytvorí schránku, pomocou ktorej bude následne posielať NetFlow pakety na kolektor. Jednotlivé pakety sú posielané pomocou metódy `Exporter::send()`, ktorá iba vypočíta ich veľkosť a odošle UDP paket.

5 Príklad použitia

K tomu, aby sme si vedeli prehliadnuť exportované záznamy, si musíme spustiť lokálny NetFlow kolektor, ktorý bude zaznamenávať zachytené záznamy do súboru. Ako príklad použijeme program `nfcapd`, ktorý spustíme, aby počúval na porte 2054.

```
nfcapd -T all -l . -I any -p 2054
```

Následne môžeme spustiť nami implementovaný exportér, ktorý bude exportovať NetFlow záznamy na adresu localhost:2054, kde počúva náš kolektor. Pri spúštaní by sme mohli definovať aj ľubovolné ďalšie argumenty pre nastavenie parametrov exportéru.

```
./flow -f test.pcap -c localhost:2054
```

Pre zobrazenie kolektorom zachytených záznamov môžeme použiť program `nfdump`, ktorý spustíme nasledovne:

```
nfdump -r <názov súboru>
```

Výsledok môže vyzerať napríklad takto:

```
2022-10-23 00:43:42.233 INVALID  Ignore TCP      192.168.0.106:56926 ->     18.64.119.11:443           0.0.0.0:0      ->          0.0.0.0:0           52        0
2022-10-23 00:43:42.233 INVALID  Ignore TCP      192.168.0.106:56906 ->     18.64.119.11:443           0.0.0.0:0      ->          0.0.0.0:0           52        0
2022-10-23 00:43:42.233 INVALID  Ignore TCP      192.168.0.106:56916 ->     18.64.119.11:443           0.0.0.0:0      ->          0.0.0.0:0           52        0
2022-10-23 00:43:42.233 INVALID  Ignore TCP      192.168.0.106:56882 ->     18.64.119.11:443           0.0.0.0:0      ->          0.0.0.0:0           52        0
2022-10-23 00:43:42.233 INVALID  Ignore TCP      192.168.0.106:56892 ->     18.64.119.11:443           0.0.0.0:0      ->          0.0.0.0:0           52        0
2022-10-23 00:43:42.264 INVALID  Ignore TCP       18.64.119.11:443   ->    192.168.0.106:56906         0.0.0.0:0      ->          0.0.0.0:0           52        0
2022-10-23 00:43:42.264 INVALID  Ignore TCP       18.64.119.11:443   ->    192.168.0.106:56926         0.0.0.0:0      ->          0.0.0.0:0           52        0
2022-10-23 00:43:42.276 INVALID  Ignore TCP       18.64.119.11:443   ->    192.168.0.106:56892         0.0.0.0:0      ->          0.0.0.0:0           52        0
2022-10-23 00:43:42.276 INVALID  Ignore TCP       18.64.119.11:443   ->    192.168.0.106:56916         0.0.0.0:0      ->          0.0.0.0:0           52        0
2022-10-23 00:43:42.276 INVALID  Ignore TCP       18.64.119.11:443   ->    192.168.0.106:56882         0.0.0.0:0      ->          0.0.0.0:0           52        0
2022-10-23 00:43:42.930 INVALID  Ignore ICMP     192.168.0.106:0     ->          8.8.8.8:0.0           0.0.0.0:0      ->          0.0.0.0:0          420        0
2022-10-23 00:43:42.945 INVALID  Ignore ICMP          8.8.8.8:0     ->    192.168.0.106:0.0           0.0.0.0:0      ->          0.0.0.0:0          420        0
2022-10-23 00:43:45.172 INVALID  Ignore UDP        192.168.0.1:50944 ->  239.255.255.250:1900        0.0.0.0:0      ->          0.0.0.0:0         4377        0
2022-10-23 00:43:47.604 INVALID  Ignore TCP      192.168.0.106:41322 ->    51.178.65.231:443           0.0.0.0:0      ->          0.0.0.0:0          104        0
2022-10-23 00:43:47.639 INVALID  Ignore TCP      51.178.65.231:443   ->    192.168.0.106:41322         0.0.0.0:0      ->          0.0.0.0:0          104        0
2022-10-23 00:43:48.697 INVALID  Ignore TCP      192.168.0.106:44850 ->    35.186.224.41:443           0.0.0.0:0      ->          0.0.0.0:0          147        0
2022-10-23 00:43:48.725 INVALID  Ignore TCP      35.186.224.41:443   ->    192.168.0.106:44850         0.0.0.0:0      ->          0.0.0.0:0          144        0
Summary: total flows: 243, total bytes: 3.2 M, total packets: 3019, avg bps: 957986, avg pps: 114, avg bpp: 1046
Time window: 2022-10-23 00:43:23 - 2022-10-23 00:43:49
Total flows processed: 243, Blocks skipped: 0, Bytes read: 19568
Sys: 0.005s flows/second: 43579.6    Wall: 0.010s flows/second: 24224.9
```

Literatúra

[1] Wikipedia contributors: NetFlow — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=NetFlow&oldid=1115766584, 2022, [Online; accessed 14-November-2022].

[2] Jacobson, V.; Leres, C.; McCanne, S.: *libpcap*. 2021, [Online; accessed 14-November-2022].
URL https://www.tcpdump.org

[3] Koenig, T.; Kerrisk, M.: *getopt(3) — Linux manual page*. [Online; accessed 14-November-2022].
URL https://www.man7.org/linux/man-pages/man3/getopt.3.html

[4] *Man page of pcap-filter*. 2022, [Online; accessed 14-November-2022].
URL https://www.tcpdump.org/manpages/pcap-filter.7.html