Vysoké učení technické v Brně
Fakulta informačních technologií

Síťové aplikace s správa sítí
Generování NetFlow ze zachycené síťové komunikace

2022 Martin Pech (xpechm00)

Obsah

1 Úvod

V rámci projektu do předmětu *Síťové aplikace a správa sítí* byl vytvořen `NetFlow exportér`. Tato aplikace umožnuje zpracovávat zachycená data ve formátu *pcap*. Po zpracování záznamů dojde k exportu analyzovaných dat na kolektor ve formátu NetFlow packetu. Výstupním formátem je tedy NetFlow záznam ve verzi 5[1] (*dále jen* ***flow***). Takto formátovaný záznam je odeslán na kolektor. Program podporuje zpracování protokolů `TCP`, `UDP` a `ICMP`. Aplikace byla vyvíjena v jazyce `C++` a je určena pro `Unixové operační systémy`.

2 Překlad a spouštění

2.1 Překlad

Po stažení zdrojových souborů je aplikaci potřeba přeložit. K tomu je možné využít přiložený soubor `makefile`, nebo překladač `G++` s příkazem:

```
g++ main.cpp -o flow -lpcap
```

2.1.1 Make

Jedná se o utilitu, která umožňuje automatizaci překladu zdrojových kódů. Posloupnost je popsána v souboru `makefile`. `Make`[2] pro tuto aplikaci nabízí následující cíle:

- make (*default*) - dojde k sestavení programu
- make checkFiles - před překladem ověří, zdali jsou v adresáři soubory potřebné pro bezchybný překlad
- clear - smaže přeložené soubory
- clearAll - smaže všechny soubory z daného adresáře

2.2 Spouštění

Aplikace funguje jako terminálová aplikace. Na vstupu očekává vstupní data ve formátu *pcap*. Vstupní data mohou do programu vstupovat jako parametr příkazové řádky (kapitola 2.2.1), nebo pomocí standartního vstupu (*STDIN*). Ke spuštění aplikace dojde příkazem `./flow` (za předpokladu, že je aplikace přeložena). Následovat mohou volitelné parametry.

2.2.1 Parametry příkazové řádky

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

- `-f <file>` - volitelný parametr, který očekává jméno *pcap* souboru. Není-li uveden, program načítá ze standartního vstupu *STDIN*.
- `-c <netflow_collector>:<port>` - volitelný parametr, očekává hostovské jméno/ip adresu místa, na které má odesílat výsledné NetFlow záznamy. Volitelně lze doplnit i port. Jako výchozí hodnoty uvažuje *127.0.0.1:2055*.
- `-a <active_timer>` - volitelný parametr značící maximální dobu sdružování packetů do jedné flow. Příjmá hodnoty v *sekundách*. Výchozí hodnota je *60*. Po vypršení se hodnoty odesílají na kolektor.
- `-i <inactive_timer>` - volitelný parametr značící maximální dobu čekání na následující packet v dané flow. Příjmá hodnoty v *sekundách*. Výchozí hodnota je *10*. Po vypršení se hodnoty odesílají na kolektor.
- `-m <count>` - volitelný parametr označující velikost *flow cache*, tedy maximální počet záznamů udržitelný v jeden moment v paměti. Při dosažení maximální velikosti dojde k odeslání nejstaršího záznamu. Výchozí hodnota je *1024*.

V případě nevyplnění některého z parametrů uvažuje aplikace výchozí hodnotu.

[1] NetFlow v5: `https://www.cisco.com/c/en/us/td/docs/net_mgmt/netflow_collection_engine/3-6/user/guide/format.html`
[2] Make: `https://www.computerhope.com/unix/umake.htm`

2.2.2 Příklady spouštění

Příklady spouštění + ukázkový výstup pomocí nástroje `nfdump` [3].

```
./flow -f tcp-fin.pcap
```

Date first seen	Duration	Proto	Src IP Addr:Port		Dst IP Addr:Port	Packets	Bytes	Flows
2022-10-07 18:59:30.402	00:00:00.000	TCP	192.0.73.2:443	->	100.69.167.92:46476	3	183	1
2022-10-07 18:59:30.402	00:00:00.000	TCP	100.69.167.92:46476	->	192.0.73.2:443	3	120	1
2022-10-07 18:59:34.722	00:00:00.048	TCP	100.69.167.92:41064	->	104.70.109.120:443	11	4796	1
2022-10-07 18:59:34.735	00:00:00.050	TCP	104.70.109.120:443	->	100.69.167.92:41064	11	2566	1
2022-10-07 18:59:41.285	00:00:00.000	TCP	192.0.73.2:443	->	100.69.167.92:46476	1	40	1
2022-10-07 18:59:41.054	00:00:00.000	TCP	107.23.110.60:443	->	100.69.167.92:32992	1	52	1
2022-10-07 18:59:40.926	00:00:00.000	TCP	100.69.167.92:32992	->	107.23.110.60:443	1	52	1
2022-10-07 18:59:44.733	00:00:00.000	TCP	142.251.36.74:443	->	100.69.167.92:48652	1	52	1
2022-10-07 18:59:44.714	00:00:00.000	TCP	100.69.167.92:48652	->	142.251.36.74:443	1	52	1
2022-10-07 18:59:56.414	00:00:00.000	TCP	107.23.110.60:443	->	100.69.167.92:32992	1	83	1
2022-10-07 18:59:56.415	00:00:00.001	TCP	100.69.167.92:32992	->	107.23.110.60:443	3	168	1
2022-10-07 18:59:48.353	00:00:00.140	TCP	142.251.36.147:443	->	100.69.167.92:38530	4	329	1
2022-10-07 18:59:48.334	00:00:00.141	TCP	100.69.167.92:38530	->	142.251.36.147:443	6	812	1
2022-10-07 18:59:40.052	00:00:17.437	TCP	162.159.135.234:443	->	100.69.167.92:59936	10	2107	1
2022-10-07 18:59:52.753	00:00:00.000	TCP	100.69.167.92:48654	->	142.251.36.74:443	1	52	1
2022-10-07 18:59:59.246	00:00:00.138	TCP	142.251.36.147:443	->	100.69.167.92:38530	5	381	1
2022-10-07 18:59:50.705	00:00:00.000	TCP	100.69.167.92:46116	->	142.250.102.188:5228	1	52	1
2022-10-07 18:59:47.341	00:00:10.989	TCP	100.69.167.92:53686	->	3.65.102.105:443	5	424	1
2022-10-07 18:59:52.778	00:00:00.000	TCP	142.251.36.74:443	->	100.69.167.92:48654	1	52	1
2022-10-07 18:59:50.705	00:00:00.000	TCP	100.69.167.92:45596	->	162.159.129.232:443	1	40	1
2022-10-07 18:59:47.420	00:00:10.910	TCP	3.65.102.105:443	->	100.69.167.92:53686	5	372	1
2022-10-07 18:59:57.899	00:00:00.205	TCP	107.23.110.60:443	->	100.69.167.92:32992	2	150	1
2022-10-07 18:59:59.233	00:00:00.131	TCP	100.69.167.92:38530	->	142.251.36.147:443	6	3699	1
2022-10-07 18:59:47.344	00:00:10.987	TCP	100.69.167.92:53694	->	3.65.102.105:443	5	424	1
2022-10-07 18:59:40.052	00:00:17.437	TCP	100.69.167.92:59936	->	162.159.135.234:443	10	400	1
2022-10-07 18:59:47.380	00:00:10.951	TCP	3.65.102.105:443	->	100.69.167.92:53694	4	320	1
2022-10-07 18:59:50.740	00:00:00.000	TCP	142.250.102.188:5228	->	100.69.167.92:46116	1	52	1
2022-10-07 18:59:50.724	00:00:00.000	TCP	162.159.129.232:443	->	100.69.167.92:45596	1	40	1

```
Summary: total flows: 28, total bytes: 17870, total packets: 105, avg bps: 4932, avg pps: 3, avg bpp: 170
Time window: 2022-10-07 18:59:30 - 2022-10-07 18:59:59
Total flows processed: 28, passed: 28, Blocks skipped: 0, Bytes read: 2432
Sys: 0.0000s User: 0.0030s Wall: 0.0007s flows/second: 39608.9 Runtime: 0.0008s
```

```
./flow -c mpech.net:2056 -a 45 < tcp-fin.pcap
./flow -f tcp-fin.pcap -c mpech.net -i 11 -m 512
```

3 Implementace

3.1 Přiblížení chování aplikace

3.1.1 Čísla portů

Za validní čísla portů jsou považovány pouze porty v rozsahu 0 až 65535 (tedy rozsah 16 bitového bezznaménkového integeru). Dojde-li k zadání nesprávného čísla portu, aplikace sama přepočítá zadanou hodnotu na validní port přetečením rozsahu. Například číslo -1 je považováno za port číslo 65535.

3.1.2 Parametr count

Jelikož ke kontrolování parametru count dochází jen v případě, že nalezený packet nenáleží do žádné flow a je tedy potřeba vytvořit novou, je hodnota `-m 0` považována za nevalidní a její chování odpovídá hodnotě `-m 1`.

3.2 Chybové kódy a oznámení

Dojde-li v programu k nějaké chybě, která je zapříčiněna uživatelským vstupem, nebo nesprávným fungováním spojení s kolektorem (např. chybně vložená adresa), je o tom uživatel spraven pomocí chybových kódů. Dále má možnost povolit chybové hlášky a oznámení změnou parametru v kódu.

[3] NFDUMP: `https://nfdump.sourceforge.net/`

3.2.1 Seznam chybových kódů

- 0 - standartní ukončení programu
- 10 - neexistující argument příkazové řádky
- 11 - nevalidní hodnota parametru argumentu
- 12 - jiný problém při čtení argumentu/parametru
- 20 - nevalidní vstupní data (chybný formát dat)
- 30 - neočekávaný typ packetu
- 41 - nepodařilo se přeložit doménové jméno
- 42 - nepodařilo se vytvořit socket
- 43 - selhalo připojení na server
- 44 - nesprávný formát portu
- 50 - selhání funkce send() - packet nebyl odeslán
- 51 - selhání funkce send() - packet byl odeslán částečně

3.2.2 Povolení oznámení a chybových výpisů

Uživatel může drobnou změnou v kódu povolit výpis informativních a chybových hlášek. Pro povolení chybových výpisů stačí změnit hodnotu proměnné `debugToggler` na hodnotu *true*. Pro povolení informativních výpisů je potřeba upravit proměnnou `msgToggler` taktéž na hodnotu *true*. Zobrazované zprávy se budou vypisovat na standartní chybový výstup (*STDERR*).

3.3 Implementační detaily

3.3.1 Obecný popis implementace

Po spuštění aplikace dojde k volání hlavní funkce `main`. V této funkci dochází ke čtení volitelných parametrů příkazové řádky a jejich zpracování. Následně dochází k ověřování vstupních souborů a volání funkce `pcap_loop`, která je zodpovědná za postupné načítání packetů. Tato funkce dále volá funkci `process_packet`, která zpracovává jednotlivé packety.

Funkce `process_packet` nejprve na základě protokolu určí, o jaký typ packetu se jedná. Na základě toho pak z příchozího packetu sbírá požadovaná data. Pro ukládání jednotlivých flows do paměti byla zvolena struktura `std::map` (kapitola 3.3.2). Po přijetí packetu a rozhodnutí o jeho typu dojde k porovnání časovačů (active a inactive timer). Vypršel-li čas některé flow, dojde k exportu na kolektor. Následně dochází k prověřování packetu. Na základě toho jej aplikace přiřadí do extistující flow, nebo vytvoří novou. V takovém případě dále aplikace ověřuje velikost flow cache a v případě naplnění taktéž exportuje na kolektor.

Export na kolektor je realizován pomocí funkce `collectorExport`. Tato funkce přijímá jako argument konkrétní flow. Informace o ní jsou zpracovány a vloženy do struktury, která kopíruje prvky NetFlow packetu. Packet je následně poskládán. Pak dochází k ověření hostovského jména a portu a navázání spojení. To je realizováno pomocí funkce `connect`. Dochází k posílání dat a uzavírání spojení. Po zpracování všech vstupních dat exportuje aplikace všechny zbývající flows na kolektor a to v pořadí, v jakém jsou momentálně uloženy ve struktuře `std::map`.

V programu se dále vyskytuje řada obslužných funkcí. Mezi ně patří funkce `debug` zodpovídající za chybové výpisy na *STDERR*, funkce `msg`, která obdobně jako funkce předešlá zodpovídá za výpis informačních sdělení z programu, funkce `checkFormat` ověřující validitu zadaného čísla a funkce `checkHostname` určující složení hostovského jména a portu a jejich korektnost.

3.3.2 std::map

Jedná se o seřazenou asociativní strukturu obsahující dvojice `klíč - hodnota`. Tato struktura byla zvolena pro svou jednoduchost vyhledávání a je základním stavebním prvkem této aplikace. Vyskytuje se hned na dvou místech. Tím prvním je struktura *packetu*. Toto použití je výhodné, protože lze ukládat například dvojice *{"srcIP", 192.168.0.5}*. Dále je tato struktura použita i pro účely uchovávání informací o existujících flows. Samotné ukládání je obdobné jako u packetu, ovšem klíč pro jednotlivé flows je tvořen unikátní hodnotou.

Unikátní hodnoty je dosaženo tak, že se z každého packetu posbírají informace o zdrojových a cílových IP adresách a portech, informace o typu služby (`ToS`) a o protokolu. Z nich je následně vytvořen `hash`, který je přidružen ke konkrétní flow a v daný okamžik je unikátní.

3.3.3 Exportovaný NetFlow packet

Pro tvorbu NetFlow v5 packetu je využito dvou struktur, které prvky a jejich velikostmi přesně odpovídají normám pro tuto verzi NetFlow záznamů. Hodnoty, které program dokázal ze získaných dat vyčíst jsou do struktur doplněny. Ostatní hodnoty jsou implicitně nastaveny na hodnotu 0.

4 Reference

[1] TCPDUMP & LIBPCAP: *Man page of PCAP* [online]. (9. 9. 2020). Dostupné z: `https://www.tcpdump.org/manpages/pcap.3pcap.html`

[2] SourceForge: *NFDUMP*. (1. 12. 2014). Dostupné z: `https://nfdump.sourceforge.net/`

[3] Wikipedia: *NetFlow*. (13. 10. 2022). Dostupné z: `https://nfdump.sourceforge.net/`

[4] Cisco: *NetFlow Export Datagram Format*. (14. 9. 2007). Dostupné z: `https://www.cisco.com/c/en/us/td/docs/net_mgmt/netflow_collection_engine/3-6/user/guide/format.html`

[5] Lars Wirzenius: *Writing manual pages*. (1. 6. 2019). Dostupné z: `https://liw.fi/manpages/`

[6] doc. Ing. Petr Matoušek Ph.D., M.A.: *echo-udp-client2.c*. (2019). Dostupné z: `https://moodle.vut.cz/course/view.php?id=231021`