VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DOKUMENTACE PROJEKTU DO PŘEDMĚTU SÍŤOVÉ APLIKACE A SPRÁVA SÍTÍ

Matěj Konopík
13. Listopadu 2022

1. Obsah

2. Úvod

Projekt má za cíl implementovat program pro generování NetFlow V5[1] dat ze zachycené síťové komunikace ve formátu *.pcap, jenž je standardem pro záznam packetů průchozích přes síť například z aplikace Wireshark[2]. Cílem je projít data ze souboru se záznamy, vygenerovat NetFlow data na základě argumentů programu a následně tato data odeslat na NetFlow Collector. Program zpracovává packety IPv4 s protokolem UDP, TCP a ICMP.

3. Platforma programu

Program flow je určen k běhu na platformě Linux, specificky na serveru merlin.fit.vutbr.cz[3]. Implementace byla provedena na operačním systém Linux pod distribucí Manjaro, jenž je založený na Arch Linuxu.

4. Překlad a použití

Pro přeložení programu je připraven Makefile s cíly all, clean a run. Použitím příkazu make se zdrojový kód flow.cpp přeloží na binární soubor flow. Zbylé příkazy se chovají v souladu se standardy Makefile.

Program se poté spustí standardně použitím příkazu ./flow. Poskytuje množinu argumentů, které mají svoje výchozí hodnoty a dovolují tak uživateli specifikovat různé parametry výsledných NetFlow dat. Program má následující signaturu:

```
flow       [-f  <filename>] [-c  <collector_ip>:<port>]
           [-a  <active_timer>] [-i  <inactive_timer>]
           [-m  <cache_size>]
```

4.1 Detaily argumentů:

[-f <filename>] - Specifikuje soubor ve formátu .pcap, z nějž bude program načítat síťovou komunikaci. Ve výchozím stavu čte ze STDIN, je ale potřeba mu přesměrovat vstup z nějakého souboru.

[-c <collector_ip>:<port>] - Specifikuje IP adresu ve formátu IPv4 a port v hodnotách od 0 do 65535. Je třeba jej zadat ve formát “adresa_kolektoru:port”. V programu se tento řetězec rozdělí podle oddělovače “:” a obě hodnoty se dále zpracovávají separátně. Výchozí hodnota je 127.0.0.0:2055.

[-a <active_timer>] - Specifikuje maximální životnost flow záznamu v paměti programu, v sekundách. Počítá se od vytvoření flow záznamu a po vypršení se danný flow záznam exportuje na kolektor. Výchozí hodnota je 60 sekund.

[-i <inactive_timer>] - Specifikuje maximální časový interval v sekundách pro životnost flow záznamu od posledního packetu, který jí náleží. Po vypršení tohoto časovače dojde k exportu flow záznamu na kolektor. Výchozí hodnota je 10 sekund.

[-m <cache_size>] - Specifikuje maximální počet flow záznamů v cache paměti programu. Výchozí hodnota je 1024 flow záznamů.

4.2 Chování při vložení špatných argumentů

Pokud-li je zadán jakýkoliv neplatný argument (například -r), program zahlásí, že argument nezná a pokračuje v činnosti. Pokud je při spuštění zadán jakýkoliv jiný znak, program jej ignoruje a spustí se. Pokud-li jsou zadány špatné hodnoty argumentů, program buď použije výchozí hodnotu, nebo skončí chybou.

4.3 Příklad spuštění s nástrojem nfdump

Pokud chceme zkusit, zda program správně funguje, můžeme použít program nfdump a jeho součást nfcapd[7]. Postup je následující:

Spuštění collectoru nfcapd na portu 2055:
```
nfcapd -w ./ -I any -p 2055
```

Spuštění flow:
```
./flow -f <input.pcap>
```

Nyní je třeba ukončit process nfcapd, například takto:
```
pkill nfcapd
```

V aktuálním adresáři se vygenerují nfdump soubory, ty lze přečíst pomocí:
```
nfdump -r <dump_file>
```

5. Strategie řešení projekt

Při řešení bylo třeba vypořádat s několika problematikami, ke kterým je níže vysvětlené řešení.

5.1 Čtení packetů

Pro čtení packetů jsem využil svůj projekt ipk-sniffer[3], který řešil čtení hlaviček a zpracování jednotlivých síťových vrstev a protokolů. Vytvořil jsem funkci, která jednotlivé packety procházela a vracela strukturu s daty packetu, které se využívaly dále.

5.2 Ukládání flow záznamů

Ukládání jsem vyřešil pomocí datové struktury std::map. Klíče se generují z unikátní kombinace položek zdrojová adresa, cílová adresa, zdrojový port, cílový port a IP protokol. V této

struktuře se poté snadno operuje se záznamy. Typ hodnoty ve std::map je speciální struktura, jenž je již připravena pro export na kolektor.

5.3 Řešení časů

Použil jsem přístup, kdy z každého čteného packetu se generuje pseudo aktuální čas zařízení. Z prvního čteného packetu se vypočítává začátek síťového záznamu a tyto dvě hodnoty stačí ke všem operacím s flow záznamy. Jsou ve formátu milisekund, které je třeba převést z příslušných hodnot čteného packetu. Z těchto dvou hodnot poté dopočítávám čas od spuštění zařízení, který je hlavní parametr operací vyžadujících čas a také je použit při exportu. Dále také používám sekundy a nanosekundy od začátku unix epochy[4], jenž jsou též potřeba při exportu.

5.4 Export flow záznamů

Export samotný využívá protokolu UDP a socket serveru. Procedura připojení a odesílání packetů byla převzata od doktora Petra Matouška[5], konkrétně z jeho programu udp-echo-client2[6]. Exportuje se pokaždé pouze jeden flow záznam, jemuž je vždy vytvořena hlavička a tyto dva prvky se dále složí do výstupního packetu a odešlou se na NetFlow collector. Export je volán v několika bodech: Pokud vyprší časovače, pokud je plná cache (exportuje se nejstarší záznam) nebo pokud jsou přečteny všechny packety a flow záznam doposud nebyl exportován.

6. Implementační detaily

6.1 Zvolené technologie

Jako jazyk jsem zvolil C++, hlavně kvůli integrovaným datovým strukturám pro ukládání flow záznamů a snazší prací s řetězci

6.2 Popis implementace

Program ve funkci main funguje v následující posloupnosti:

1. Zpracování argumentů
2. Otevření .pcap souboru
3. Smyčka přes všechny packety vstupního souboru
    a. parsing packetu a aktualizace časových proměnných
    b. kontrola vypršení flow záznamů a případný export
    c. tvorba flow záznamu nebo aktualizace existující
4. Export všech zbývajících flow záznamů.

Potřebné detailnější operace řeší funkce. Hlavní jsou parse_packet() pro zpracování dat z packetu, get_flow_key() pro vytvoření unikátního klíče do std::map, init_flow() pro vytvoření flow záznamu, jenž je typem pro hodnotu ve std::map, set_netflow_header() pro nastavení položek hlavičky exportovaného NetFlow packetu, export_flow() pro kompletní řešení exportu packetu a

get_oldest_flow(), jenž vyhledá klíč nejstarší flow v mapě, pokud je třeba uvolnit cache a exportovat nejstarší flow záznam.

6.3 Datové struktury

Jak již bylo zmíněno, pro ukládání flow záznamů slouží struktura std::map<map_key_t , flow_data>. Typ map_key_t je zde klíčem, jenž je definován jako tuple<string, string, uint16_t, uint16_t, uint8_t>.
Flow_data je struktura, jelikož se v ní položky skládají v paměti za sebe a hodí se proto pro export. Pomocné struktury jsou packet_data, jenž uchovává data ze zpracovaného packetu a netflow_header, která podobně jako flow_data slouží k uchovávání dat pro hlavičku NetFlow packetu a přímý export.

7. Testování

Pro testování jsou používal nástroj nfdump[7], jako je popsáno v paragrafu 4.3. Testoval jsem UDP, TCP i ICMP. Pro referenci jsem využil program softflow, který vykonává stejnou činnost.
Příklad výstupu z testovacího souboru:

```
Date first seen             Duration     Proto      Src IP Addr:Port          Dst IP Addr:Port   Packets    Bytes Flows
2022-09-28 00:34:00.211     00:00:00.017 UDP     10.190.100.195:17130 ->   10.190.103.255:59668        2     1606     1
2022-09-28 00:33:59.441     00:00:00.012 UDP     10.190.100.195:21382 ->   10.190.103.255:59668        2     1504     1
2022-09-28 00:33:58.858     00:00:01.000 UDP     100.64.192.180:33750 ->   100.64.223.255:33750        2      582     1
2022-09-28 00:34:00.265     00:00:00.000 UDP     100.64.192.223:5601  ->   100.64.223.255:5601         1       72     1
2022-09-28 00:33:58.777     00:00:00.998 UDP      100.64.195.73:33750 ->   100.64.223.255:33750        2      582     1
2022-09-28 00:33:59.609     00:00:00.000 UDP     100.64.204.255:60872 ->   100.64.223.255:57065        1      244     1
2022-09-28 00:33:59.615     00:00:00.000 UDP     100.64.204.255:61384 ->   100.64.223.255:57065        1      158     1
2022-09-28 00:33:58.588     00:00:01.009 UDP     100.64.216.215:33750 ->   100.64.223.255:33750        2      582     1
Summary: total flows: 8, total bytes: 5330, total packets: 13, avg bps: 25426, avg pps: 7, avg bpp: 410
Time window: 2022-09-28 00:33:58 - 2022-09-28 00:34:00
Total flows processed: 8, passed: 8, Blocks skipped: 0, Bytes read: 752
Sys: 0.0023s User: 0.0046s Wall: 0.0016s flows/second: 5044.3 Runtime: 0.0017s
```

8. Závěr

Projekt se mi podařilo včasně zpracovat a osvojil jsem si mnohé nové zkušenosti. Výsledný program by mohl sloužit komukoliv, kdo potřebuje snadno a rychle vytvořit NetFlow záznamy pro analýzu dat na síti.

9. Reference

1:
https://www.cisco.com/c/en/us/td/docs/net_mgmt/netflow_collection_engine/3-6/user/guide/format.html#wp1006108
2:
https://www.wireshark.org/
3:
https://github.com/terrorgarten/IPK_P2
4:
https://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap04.html#tag_04_16
5:
https://www.vut.cz/lide/petr-matousek-2520
6:
https://moodle.vut.cz/pluginfile.php/502893/mod_folder/content/0/udp/echo-udp-client.c
7:
https://nfdump.sourceforge.net/