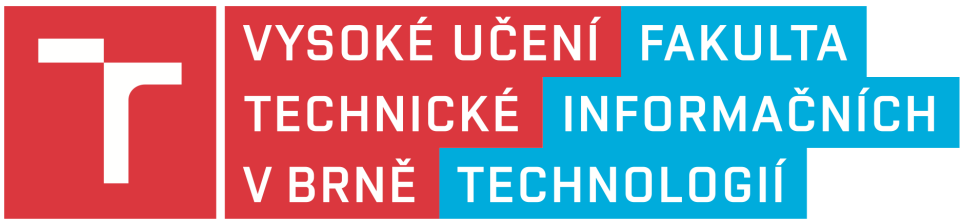


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Projektová dokumentace

Generování NetFlow dat ze zachycené síťové komunikace

ISA - Síťově aplikace a správa sítí

Tadeáš Kachyňa (xkachy00)

14. listopadu 2022

Obsah

1 Úvod do problematiky **2**

2 Návrh aplikace **3**

2.1 Struktury 3

2.2 Funkce 3

3 Program **4**

4 Použití **6**

4.1 Spuštění 6

4.2 Testování 7

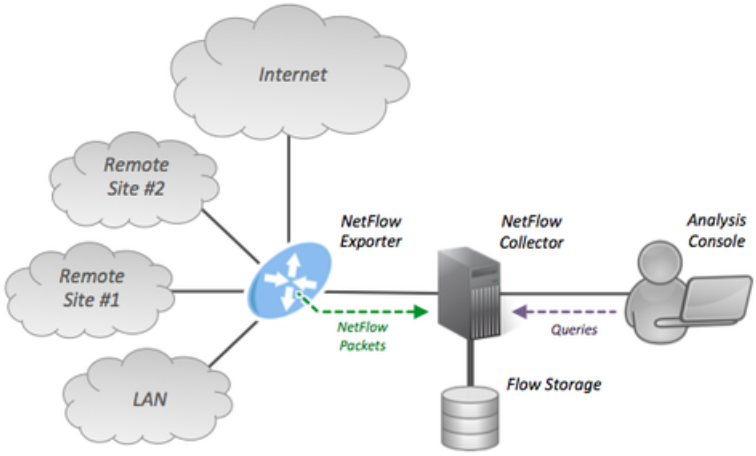
4.3 Příklady použití 7

5 Závěr **8**

6 Zdroje **8**

1 Úvod do problematiky

NetFlow je síťový monitorovací protokol, vyvinutý společností Cisco Systems, určený k zachycování a meření objemu a typu síťového provozu. Poskytuje tak administrátorům podrobný pohled do provozu v reálném čase. Tvoří tak nepostradatelnou část každé sítě. S pomocí tohoto protokolu můžeme odhalovat vnější i vnitřní problémy, úzká místa v sítí.



Obrázek 1: Netflow architektura

NetFlow monitoring se skládá typicky ze tří komponent:

- Exportér - zařízení, které má na starosti shromažďování informací o toku a jejich exprort na kolektor
- Kolektor - program běžící na serveru, který je zodpovědný pro příjem, ukládání a předzpracovávání jednotlivých "flows"
- Analyzátor - aplikace která zpracovává jednotlivé toky získaná kolektorem

Generování NetFlow dat začíná když na síťové zařízení přijde paket. Nejprve zkontroluje podle klíče, zda-li již je ve cache přítomen. Pokud je, agreguje informace s paketem se stejnými klíčovými vlastnostmi, jinak vytvoří nový flow. Samozřejmě, celá pointa NetFlow je data následně posílat na kolektor, takže data jsou periodicky posílána na kolektor v procesu známém jako "flow expiration". Ten zahrnuje několik momentů, kdy k němu dojde:

- aktivní časovač = znamená doba po kterou přichází na exportér pakety, typická doba je na mnoha platformách 30 nebo 60 minut.

- neaktivní časovač = nám říká, po jakou dobu nepřišel žádný paket (od posledního), po té době tok exportujeme, běžná doba je nastavena na 15 sekund.
- zaplnění cache flow
- detekce konce toku = např. u TCP příznak RST nebo FIN

Existujě několik verzí protokolu. Mezi ty nejznámější patří verze 5, kterou implemenetuji, a 9. Verze 9 se líší v rozšíření o šablony a o informace v L2 vrstvě.

Jak exportér, tak i kolektor můžeme ještě rozšířit o vzorkování, což je technika, která vede ke snížení nároku na hardware. Vzorkování může být náhodné, tak i deterministické.

2 Návrh aplikace

Mým úkolem bylo podle zadání implementovat exportér. Rozhodl jsem se implementovat verzi 5 v jazyce C++ a to z důvodu jednodušší práce se stringy.
Program je z důvodu přehlednosti strukturován do několika zdrojových souborů. Konkrétněji do `flow.cpp, client.cpp , arguments.cpp` a jejich hlavičkových souborů.

2.1 Struktury

`NetFlowV5Packet` - def. v `flow.hpp`
Struktura NetFlow paketu verze 5 obsahující tělo i hlavičku.
`flowInfo` - def. v `flow.hpp`
Struktura která agreguje a zároveň uchovává hodnoty pro pakety se stejnou šestící klíčů.
`Arguments` - def. v `arguments.hpp`
Struktura držící hodnoty programových argumentů.

2.2 Funkce

soubor `flow.hpp`
`packetParser` = hlavní funkce, zpracovává, analyzuje pakety
`storePacket` = funkce k uložení paketu do nově vytvořeného nebo již existujícího flow
`activeTimer` = kontrola aktivního časovače

`inActiveTimer` = kontrola neaktivního časovače

`countMiliseconds` = přepočet času na milisekundy

`exportToCollector` = funkce zajišťující exportování na vzdálený netflow kolektor.

soubor `arguments.h`

`argumentsParsing` = parsuje programované argumenty a ukládá je do struktury

`resolveIPAddrFromName` = získává IP adresu z jména

`resolveHostPort` = z parametru -c rozděluje string na ip adresu a port

`printHelp`

soubor `client.hpp`

`setUDPClient` = nastavuje prvotní spuštění UDP klienta

3 Program

Po spuštení programu se nejprve načtou programové argumenty do struktury `args`. Pro jejich zpracování jsem používal knihovnu `<getopt.h>`. Program počítá s tím, že uživatel zadává validní hodnoty. Jinak je v některých případech použita defaultní hodnota. Protože pracuji v programu primárně s milisekundami, násobím časovače konstantou 1000.

Vytvořím a nastavím soket pro UDP spojení přes které se budou následně zasílát NetFlow pakety na vzdálený kolektor. Adresa i port jsou můžou být speficikovány uživatelem. Kód klienta jsem převzal z přednášek. Citace a odkaz jsou uvedeny ve zdrojovém kódu `client.cpp`.

Nyní je již vše připravené ke zpracování paketů. Funkcí `pcap_open_offline()` načtu soubor (pokud je v prvním parametru funkce uveden znak "-", tak ze standardního vstupu) a `pcap_loop()` zpracovávám pomocí jednotlivé pakety voláním tzv. call-back funkce `packetParser`.

Nadefinuji si potřebné hlavičky protokolů a inicializuji bootovací čas, což v mém případě reprezentuje příchod prvního paketu z `.pcap` souboru. (Pokud by byla prováděna online analýza, tento čas by reprezentoval čas zapnutí síťového zařízení).

Zpracuji ethernet a ip hlavičku a analyzuji jaký protokol se nachází ve vyšší vrstvě. Exportér zpracovává pouze protokoly TCP, UDP a ICMP, jiné ignoruje.

Z IP vrsvy mě zajímá zdrojová a cílová IP adresa, Type of Service (ToS) a délka. Délku, která v NetFlow stuktuře vyznačuje jako celkové číslo všech bytů v L3 struktuře, získávám z `->ip_len`.

Z TCP protokolu zpracovávám zdrojový a cílový port a kontroluji přítomost příznaků FIN a RST, které budou použity jako indikátor exportu toku. U UDP je to téměř podobné. U ICMP protokolu, protože se nenachází na transportní vrstvě, tak u něj nenalezneme žadné porty. Proto jsem se rozhodl je nahradit nulami. Ačkoliv je trochu sporné zda-li bychom měli pakety u tohoto protokolu agregovat, já jsem se rozhodl pro.

Pro uchovávání toků(flows) jsem zvolil vestavěnou stukturu jazyka `std::map`, kde si jako klíč držím šestici `std::tuple` obsahující zdrojovou a cílovou IP adresu, porty, typ protokolu a ToS (type of service). Dle manuálu je standardem si uchovávat sedmici, ale protože položku interface nevyužiji, tak jsem rozhodl takhle. Další důležité informace pro každou šestici klíčů jako čas prvního a poslední paketu, jejich počet, TCP flagy a další si ukládám do nadefinované struktury `flowInfo`.

Nejprve než začnu paket ukládat, tak zkotroluji aktivní a ineaktivní časovače. Ty mohou být ovlivěny uživatelem, v opačném případě se použijí defaultní hodnoty 60 a 10 sekund. U obou časovačů si sestrojím cyklus, kde procházím každý uchovaný tok. U každédo z nich, na základě který časovač kontroluji, se dívám na čas prvního nebo posledního odeslaného paketu. Poté podmínkou zkontrolují, zda byla splněna. Pokud ano, tak tok exportuji na kolektor a vymažu z cache.

Následně probíhá kontrola flow cache a její zaplněnosti. Pokud je zaplněná, tak vyexportuji tok s nejstarším časem prvně zaznamenaného paketu daného toku.

Poté si paket ukládám. Zkontroluji zda-li tok s danou šesticí kličů již existuji. Pokud ano, tak jej agreguji. Jinak vytvořím nový tok.

V poslední řadě, a to jen u TCP protokolu, kontroluji výskyt FIN a RST příznaků, které znamenají (ne)úspěšné ukončení spojení. Takový paket exportuji.

Zkontroluji zaplněnost flow cache (toky exportuji pokud se aktuální zaplněnost rovná maximální velikosti), a až následně paket uložím. Dle jeho klíče ho zahrnu do již existující flow nebo vytvořím novou. Navíc u TCP kontroluji ještě přítomnost flagů `FIN` a `RST`. Protože ICMP pakety neobsahují zdrojový a cílový port, rozhodl jsem se je v mé implementaci nahradit nulami.

Pokud je třeba tok exportovat, zavolá se funkce `exportToCollector()`. Zvolil jsem variantu jednodušší implementace, takže mám nadefinovanou jednu stukturu pro hlavičku i tělo NetFlow protokolu a posílám stylem jeden flow jeden export. Potřebné informace do protokolu si vyextrahuji ze struktury pro uchovávání flows, převedu pomocí funkcí do tzv. "network byte

order"a následně pomocí UDP klienta odešlu. Původně jsem měl nadefinované dvě struktury, bohužel se mi nepodařilo v nějakém rozumném čase správně zprovoznit buffer, do kterého bych vkládal reference na jednotlivé struktury a ten odesílal. Většinou kolektor již zahlásil špatnou verzi NetFlow protokolu, ačkoliv bylo vše správně vyplněno. Proto jsem rozhodl pro jinší implementaci.

Nejzajímavější částí mé implementace , kterou bych rád zmínil, jsou časové značky. Strávil jsem nad tím spoustu času, nespočetkrát jsem tuto část reimplementoval, protože jsem si až ke konci uvědomil v jakém formátu je po mně NetFlow protokol požaduje. Struktura `timeval` obsahuje časovou značku v sekundách a mikrosekundách, ale ve většině případů je požadovaný formát v milisekundách. Proto jsem vytvořil funkci, která mi tento převod zajišťuje - `ts.tv_sec * 1000 + (ts.tv_usec + 500)/1000` - konstanta 500 je ve výrazu přidána pro lepší zaokrouhlování.

4 Použití

4.1 Spuštění

Ke spuštění použijeme následující příkaz:

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

---

kde:

- -f je jméno analyzovaného souboru nebo STDIN
- -c je IP adresa, nebo hostname NetFlow kolektoru. volitelně i UDP port (defaultně 127.0.0.1:2055)
- -a je interval v sekundách, po kterém se exportují aktivní záznamy na kolektor (defaultně 60)
- -i je interval v sekundách, po jehož vypršení se exportují neaktivní záznamy na kolektor (defaultně 10) specifikováno)

- -m je velikost flow-cache. Při dosažení max. velikosti dojde k exportu nejstaršího záznamu v cachi na kolektor (defaultně 1024).

Všechny parametry jsou volitelné. Pro zobrazení nápovědy zadejte parametr `-h`. Ostatní parametry budou ignorovány.

4.2 Testování

Testování v mém případě probíhalo za pomocí kolektoru `nfcapd`, který jsem nechal běžet v druhém okně. Mezitím jsem spustil můj exportér. Po té jsem si nástrojem `nfdump` zobrazil výsledky. Při tom jsem měl ještě spuštěný Wireshark, kde jsem tuto komunikaci zachytával a kontroloval. Projekt byl vypracováván na operačním systému Linux, distr. Linux Mint a jeho přeložení i spuštění otestováno na školním serveru Merlin. Pro správný překlad na Merlinovi jsem musel přidat do zdrojového souboru `define __FAVOR_BSD`. Bez něj nastávaly chyby u struktur `tcp.h` a `udp.h`.

Zapnetě si v okně v druhém okně terminálu kolektor nfcapd, například za pomocí příkazu:

```
nfcapd -T all -l . -I any -p 2055
```

Poté spusťte exportér, např. uvedenými příkazy uvedenými této sekce.
Kolektor nfcapd exportuje nashromážděné informace pouze jednou za 5 minut.
Po té můžete kolektor vypnout stisknutím CTRL+C.
Výsledné informace si můžete zobrazit za pomocí nástroje nfdump.

```
nfdump -r nfcapd.x
```

kde x je datum ve formátu yymmddhhmm, neboli kdy kolektor zpracoval vaše datové toky.

4.3 Příklady použití

```
./flow -f input.pcap -c 192.168.0.1:2055
./flow -f udp.pcap -c 192.168.0.1:2055 -i 5
```

```
./flow -m 10 <tcp.pcap
```

5 Závěr

Myslím si, že se mi podařilo naplnit celé zadání projektu. Určitě je tu spousta věcí, která by šla ještě vylepšit. Například podpora NetFlow protokolu verze 9. Nejnáročnější fází na celém projektu bylo určite testování a validace. V rámci projektu jsem se zlepšil v práci s knihovnou libpcap a zpracováním paketů, porozumnění v jakých strukturách se informace o paketech uchovávají a prohloubil si znalosti o jazyce C++, především strukturách jako jsou mapa nebo tuple a o protokolu NetFlow.

6 Zdroje

- Obrázek 1: https://en.wikipedia.org/wiki/File:NetFlow_Architecture_2012.png
- https://en.wikipedia.org/wiki/NetFlow
- zadání projektu v IS
- Formát NetFlow protokolu - http://www.cisco.com/c/en/us/td/docs/net_mgmt /netflow_collection _engine/3-6/user/guide/format.htmlwp1003394
- přednáškové slajdy o NetFlow