

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Síťové aplikace a správa sítí

Dokumentace k projektu

Obsah

1	Úvodní strana	2
1.1	Zadání	2
2	Uvedení do problematiky	3
2.1	Co je NetFlow	3
2.2	Tento projekt	3
2.3	Návrh aplikace	3
3	Popis implementace a práce aplikace	4
3.1	Struktura	4
3.2	Jádro programu	4
3.3	Analýza a uložení argumentů příkazové řádky	5
3.3.1	Validní hodnoty a vstupy	5
3.4	Otevření UDP socketu a příprava na čtení	5
3.5	Čtení a zpracování vstupu	5
3.6	Export flows	6
3.7	Vytvoření a aktualizace flow	6
3.8	Ukončení	6
4	Implementační detaily a chování	7
4.1	Návratové kódy	7
4.2	Konstanty	8
5	Návod na použití	9

1 Úvodní strana

K vypracování projektu do ISA jsem si zvolil variantu projektu Generování NetFlow dat ze zachycené síťové komunikace. Projekt může být implementován v jazycích C nebo C++, obojí ovšem za pomoci knihovny libpcap.

1.1 Zadání

V rámci projektu implementujte NetFlow exportér, který ze zachycených síťových dat ve formátu pcap vytvoří záznamy NetFlow, které odešle na kolektor.

- Jako export stačí použít NetFlow v5. Pokud byste implementovali v9 se šablonami, bude to bonusově zohledněno v hodnocení projektu.
- Pro vytváření flow stačí podpora protokolů TCP, UDP, ICMP.
- Informace, které neznáte (srcAS, dstAS, next-hop, aj.) nastavte jako nulové.
- Při exportování používejte původní časové značky zachycené komunikace.
- Pro testování můžete využít nástroje ze sady nfdump (nfdump, nfcapd, nfreplay, ...).
- Pro vytvoření vlastního testovacího souboru můžete použít program tcpdump.
- Exportované NetFlow data by měla být čitelná nástrojem nfdump.

2 Uvedení do problematiky

Data se ze zařízení uživatele na router odesílají v podobě paketů. Ty obsahují posílaná data a k nim potřebné informace, jako například zdrojová či cílová IP adresa. Ve spoustě případů však tyto pakety mají mnoho společných dat (v hlavičkách zejména), a proto (i z jiných důvodů) existuje NetFlow.

2.1 Co je NetFlow

Díky NetFlow můžeme monitorovat síťový provoz. K tomu používáme NetFlow exportér a kolektor. Exportér má k dispozici data ve formátu NetFlow, tzv. flows a tato data odesílá na kolektor. Kolektor tato data přijímá a ukládá. [3]

Co jsou ty flows? Jsou to sdružené pakety se stejnou zdrojovou a cílovou adresou, případně porty a stejným protokolem. V tomto projektu pracuji s NetFlow verzí 5, což je podstatná informace, jelikož v jiné verzi se pakety agregují ještě za dalších podmínek.

2.2 Tento projekt

Úkolem tohoto projektu je vytvořit právě NetFlow exportér. Takový exportér, který přijme data ze síťového provozu (ze souboru či standardního vstupu) ve formátu PCAP, pakety zpracuje, vytvoří flows a podle daných kritérií se flows poté odešlou na kolektor.

2.3 Návrh aplikace

Aplikace je navržena tak, aby v maximální možné míře odpovídala zadání a plnila úkony co nejefektivněji. Na první pohled se jedná o prostý úkol - získat pakety, zpracovat je a odeslat. Přesně takto je rozvržena práce této aplikace.

Při návrhu bylo nutné zohlednit veškerá kritéria daná zadáním.

3 Popis implementace a práce aplikace

Samotný počet řádků kódu programu je kolem 1 000 - rozdělení činnosti do funkcí, tříd nebo modulů je naprosto vyžadované, pokud bych tak neudělal, zdrojový kód by byl velmi těžko čitelný.

3.1 Struktura

Zdrojový kód je rozdělen do modulů. Hlavním modulem je `main.cpp`, který tvoří jádro programu. Kromě hlavního modulu jsou všechny ostatní pojmenovány jako `isa_*.cpp` (nebo `isa_*.hpp` v případě hlavičkového souboru), kde `*` prozrazuje význam daného modulu.

Dalšími moduly ve zdrojovém kódu tak jsou:

- `argparse`
 - zpracování argumentů příkazové řádky
- `bitset`
 - práce s bitovým polem a daty
- `classes`
 - definice tříd
- `collector`
 - práce s kolektorem, UDP
- `flow`
 - operace s flows
- `flowexport`
 - příprava a export flows
- `utility`
 - ostatní funkce

3.2 Jádro programu

Nachází se v souboru `main.cpp`, ve kterém se vykonávají veškeré činnosti od zpracování argumentů příkazové řádky po export flowů na kolektor.

Hlavní činnost programu tvoří funkce `collectAndExport` a je následující:

1. Analýza a uložení argumentů příkazové řádky
2. Otevření UDP socketu, příprava na čtení
3. Čtení a zpracování vstupu
 - Načtení dat z hlaviček
 - Export flows
 - Vytvoření či aktualizace flow
 - Uvolnění flow-cache
4. Ukončení

3.3 Analýza a uložení argumentů příkazové řádky

V této části se neděje nic zajímavého - ke zpracování argumentů používám vlastní třídu `programArguments` a funkci `getopt`. V třídě jsou uloženy veškeré argumenty, jejich stavy, zadané i výchozí hodnoty. Funkce týkající se těchto činností se nachází v modulu `argparse`.

Veškeré potřebné proměnné, pole a informace pro práci programu jsou uloženy ve třídě `data`. Uložení všech důležitých informací do jedné třídy zajišťuje jednodušší práci s daty a zpřehledňuje celý kód.

Kromě výše zmíněných si program vede vlastní statistiky (třída `statistics`) jako je počet zpracovaných paketů nebo počty vytvořených, exportovaných či aktualizovaných flows. Ty jsou užitečné v případě, že se rozhodnete, aby vám program vypisoval svůj postup práce (proměnná `ISA_VERBOSE_PRINT`).

3.3.1 Validní hodnoty a vstupy

V tomto ohledu se program snaží být co nejméně agresivní a řídí se maximálně dle zadání. Zadání však neřeší všechno, a proto jsem rozhodl, že při zadávání číselných argumentů:

- časovače musí být celé nenulové číslo,
- flow-cache hodnota může být jakákoliv.

Všechny číselné parametry jsou totiž vedeny v programu jako `unsigned int`, uživatel tak sice může zadat záporné číslo, program to bude i tak interpretovat jako kladné.

Pro časovače nedává smysl, aby měly nulovou hodnotu, jelikož by se export poté prováděl v každém případě, a pro to je v programu existuje jiná možnost.

Flow-cache může mít jakoukoliv velikost, nejmenší možnou hodnotou je 0. Díky tomu, že je umožněna nulová velikost flow-cache, program nabízí uživateli možnost okamžitého exportu flows, pokud si tak přeje.

V případě hodnot parametru `-c` je možné zadat buď to IPv4 adresu, hostname spolu s portem či bez. Nevalidní vstup v tomto parametru znamená, že flows se neexportují na žádný kolektor, což program v takovém případě patřičně ohlásí.

Podstatným parametrem je `-f`, jelikož je jediným parametrem, který musí být opravdu validní, a pokud zadáte špatně vstupní soubor (či nezadáte, ale standardní vstup nebude v PCAP formátu), nic se neprovede a program pouze vypíše hlášku.

3.4 Otevření UDP socketu a příprava na čtení

Pokud vše proběhlo v pořádku a na příkazové řádce nebyl zadán nějaký nesmysl či jste nechtěli vypsát pouze `help` zprávu, program se přesunuje do této části.

Otevře se zařízení, ze kterého se bude zachytávat síťový provoz, v případě tohoto projektu offline. Aplikuje a nainstaluje se filtr, kterým je zde `icmp or tcp or udp`, jelikož program pracuje pouze s těmito protokoly.

V případě jakéhokoliv neúspěchu v těchto činnostech je program nucen ukončit svoji činnost a dále nepokračovat, jelikož to znamená, že nelze přecházet vstupní soubor. Pro implementaci čtení z pcap souboru jsem se inspiroval z tutoriálu `TcpDump.org` [6].

Následně se vytvoří socket pro UDP spojení, které se poté otevře a zůstane otevřené do ukončení práce. Pokud nastane chyba zde, program dále pokračuje ve své činnosti, avšak výsledné flows se poté nebudou nikam exportovat. Po ukončení práce program skončí s patřičným kódem a chybovou hláškou. Kód pro UDP klienta byl převzat z přednášky ISA, viz. [7].

3.5 Čtení a zpracování vstupu

Nejprve se vynuluje bitové pole pro zápis odesílaných dat, následuje načtení dat z hlaviček a vytvoření pětice dat společné pro pakety v jednom flow.

Při načtení hlaviček může nastat k zjištění, že IP hlavička nemá požadovanou velikost nebo se jedná o paket nepodporovaného protokolu. V těchto případech se ihned začne zpracovávat další paket. Jaké číslo odpovídá kterému protokolu je uvedeno viz [2], podle tohoto seznamu se program řídí při rozlišování, s kterým protokolem pracuje.

3.6 Export flows

V moment, kdy je připravena pětice dat - adresy a porty zdroje a cíle, protokol, se začnou exportovat flows, kterým vypřel nějaký z časovačů, a nebo má nastavené TCP flags v případě TCP paketu.

Pořadí exportů je následující:

1. Aktivní časovač
2. Neaktivní časovač
3. TCP Flagy

Vypršením aktivního časovače rozumíme překročení časového intervalu mezi prvním a posledním paketem v jedné flow (nebo také: interval mezi vytvořením flow a jeho poslední aktualizací, [4]). Časový interval, po jehož překročení se flow exportuje na kolektor udává parametr `-a` a jeho výchozí hodnota je nastavena na 60 sekund.

Vypršením neaktivního časovače rozumíme překročení časového intervalu mezi posledním paketem (či: poslední aktualizace flow, [4]) ve flow a současným časem programu (tzv. SysUpTime programu). Výchozí hodnotou je 10 sekund.

Na kolektor se exportuje takový TCP paket, který má nastavené TCP flags `TH_FIN` nebo `TH_RST`.

3.7 Vytvoření a aktualizace flow

Nejprve se aktualizuje čas systému, tzv. SysUpTime. Ten je na počátku nastaven na 0 a jeho hodnotou je počet milisekund, resp. mikrosekund od přijetí prvního paketu. Při přijetí prvního paketu se zaznamená jeho unixový čas, který je považovaný za boot-time našeho zařízení. Podle tohoto času se právě chovají časovače.

Nový flow bude vytvořen právě tehdy, pokud ve flow-cache neexistuje žádný flow, který by měl stejnou pětici dat, v opačném případě se pouze aktualizuje nalezený flow.

Po vytvoření či aktualizaci flows se na kolektor exportuje nejstarší flow - pouze v případě, že současný počet flows (po vložení nového flow) již překračuje limit velikosti flow-cache (standardně 1024 flows).

3.8 Ukončení

Po přijmutí a zpracování posledního paketu se ukončí cyklus čtení ze vstupu a exportují se zbylé neexportované flows z cache do kolektoru. UDP socket se uzavírá a program svoji činnost končí. Pokud v průběhu práce nastala nějaká chyba, program vypíše její popis.

4 Implementační detaily a chování

Implementační detaily či popisy chování:

- Chyby v běhu
 - Vypisují těsně před ukončením programu, vždy jako poslední.
 - Návrátový kód a popis reflektuje pouze poslední chybu.
- Protokol paketu
 - Podporují se pouze protokoly ze zadání, tj. TCP, UDP, ICMP.
 - V případě ICMP paketu jsou hodnoty portů nastaveny na nuly. Agregují se do flows stejně jako ostatní pakety.
 - Flows obsahující TCP pakety se navíc mohou exportovat dle TCP flags.
 - Ostatní protokoly nejsou podporovány a jsou při čtení ze souboru přeskočeny.
 - Pro čtení a práci s pakety jednotlivých protokolů jsou vytvořeny struktury - kód pro strukturu TCP, IP, Ethernet hlaviček jsem převzal z TcmDump.org tutoriálu [6]. Ostatní struktury protokolů byly vytvořeny v souladu s datagramy pro daný protokol, viz [5] a [1].
- Systémový čas, boot-time
 - Počáteční (nulový) čas je shodný s časem prvního přečteného paketu.
- Flow
 - Je struktura obsahující strukturu hlavičky a záznamu, které obsahují požadované položky dle Net-Flow V5. [4]
 - Na kolektor se exportuje v požadovaném formátu tak, že se obsah Flow struktury převede do bitového pole.
- Flow Cache
 - Používá se `std::map`.
 - Maximální velikost z parametru `-f` určuje, kdy se exportuje nejstarší flow záznam (tj. první záznam v cache).
 - Export probíhá až při překročení velikosti.

Zvláštním souborem je hlavičkový soubor `isa.hpp`, který definuje struktury pro ICMP, UDP, TCP, IP a Ethernet hlavičky a obecně definice, patřící do celého programu.

4.1 Návrátové kódy

- 0 - program skončil v pořádku
- 1 - neúspěšné zpracování argumentů
- 2 - otevření pcap zařízení se nezdařilo
- 30 - neúspěch v aplikaci filtru
- 31 - nelze nainstalovat filtr
- 40 - neznámý host kolektoru

- 41 - nepodařilo se vytvořit socket pro UDP spojení
- 42 - nelze se připojit k socketu
- 43 - odeslání dat na kolektor se nezdařilo
- 44 - data na kolektor byla odeslána pouze částečně
- 45 - nespecifikovaná chyba v rámci UDP spojení
- 50 - detekován neznámý protokol v paketu, paketech
- 51 - IP hlavička některého paketu nemá správnou velikost

4.2 Konstanty

Velikost bufferu pro odesílání dat je 1024 bajtů, dané konstantou `UDP_BUFF_SIZE`.

Počet bajtů dat, která se odešlou na kolektor je 576 bajtů, dané konstantou `NETFLOW_V5_DATASIZE`.

Pokud je `ISA_VERBOSE_PRINT` nastaveno na `true`, bude se provádět průběžný výpis vytváření, aktualizace a exportu flows. Ve výchozím stavu je tento výpis vypnut.

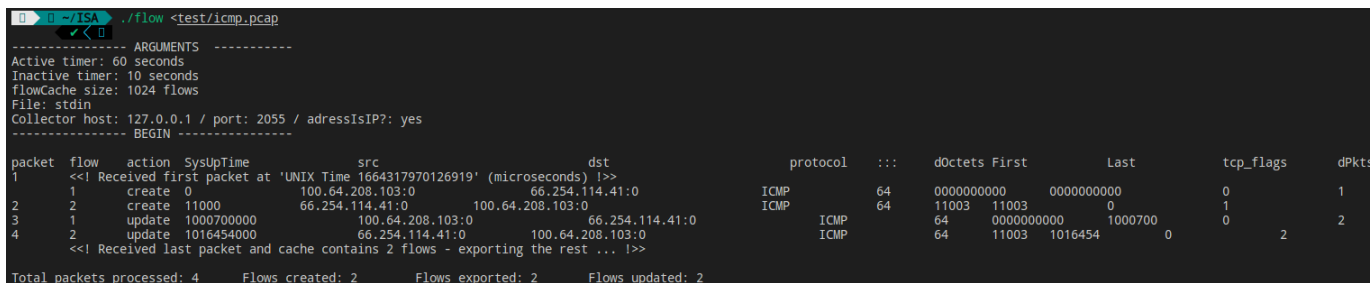
5 Návod na použití

Po rozbalení archivu program zkompilujete příkazem `make` a poté spustíte takto:

```
./flow [-f F] [-c C] [-a A] [-i I] [-m M]
```

V případě, že si nejste jisti syntaxem, spusíte program s parametrem `-h`, který vám podrobně popíše syntaxi spuštění. Pro úspěšný překlad je nutné mít na stroji překladač `g++` podporující verzi C++17. Makefile mimo jiné podporuje další příkazy (spouštíte `make <příkaz>`):

- `clean` - smaže pracovní soubory
- `remake` - zkratka pro `make clean` a `make`
- `pack` - zabalí obsah adresáře (pro odevzdání do IS)
- `manpage` - zobrazí manuálovou stránku `flow`



```
~/.ISA ./flow -t test/icmp.pcap
----- ARGUMENTS -----
Active timer: 60 seconds
Inactive timer: 10 seconds
flowCache size: 1024 flows
File: stdin
Collector host: 127.0.0.1 / port: 2055 / addressIsIP?: yes
----- BEGIN -----
packet  flow  action  SysUpTime      src      dst      protocol  :::  dOctets First      Last      tcp_flags  dPkts
1      1      <<I Received first packet at 'UNIX Time 1664317970126919' (microseconds) I>>
2      2      create  0            100.64.208.103:0      66.254.114.41:0      ICMP      64      00000000000 0000000000 0          1
3      1      create  11000        66.254.114.41:0      100.64.208.103:0      ICMP      64      11003 11003 0          1
4      2      update  10007000000  100.64.208.103:0      66.254.114.41:0      ICMP      64      00000000000 1000700 0          2
4      2      update  1016454000   66.254.114.41:0      100.64.208.103:0      ICMP      64      11003 1016454 0          2
<<I Received last packet and cache contains 2 flows - exporting the rest ... I>>
Total packets processed: 4      Flows created: 2      Flows exported: 2      Flows updated: 2
```

Obrázek 1: Příklad spuštění programu se zapnutým výpisem postupu (vstup obsahuje 4 ICMP pakety)

Použitá literatura a zdroje

- [1] Internet Control Message Protocol.
URL https://en.wikipedia.org/wiki/Internet_Control_Message_Protocol
- [2] List of assigned Internet Protocol Numbers.
URL <https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml>
- [3] NetFlow - Wikipedia.
URL <https://en.wikipedia.org/wiki/NetFlow>
- [4] NetFlow Export Datagram format.
URL https://www.cisco.com/c/en/us/td/docs/net_mgmt/netflow_collection_engine/3-6/user/guide/format.html
- [5] User Datagram Protocol.
URL https://en.wikipedia.org/wiki/User_Datagram_Protocol
- [6] Carthens, T.: PCAP file reader tutorial. 2002.
URL <https://www.tcpdump.org/pcap.html>
- [7] Matoušek, P.: UDP client with two parameters. 2016.
URL https://moodle.vut.cz/pluginfile.php/502893/mod_folder/content/0/udp/echo-udp-client2.c?forcedownload=1