
# ISA – Síťové aplikace a správa sítí

## Dokumentácia k projektu

Generování NetFlow dat ze zachycené síťové komunikace

**Dátum:** 14.11.2022
**Autor:** František Fúdor (xfudor00)

# Obsah

# Zoznam obrázkov

# Teoretické poznatky

# Úvod

**NetFlow** bol predstavený v roku 1996 pre Cisco smerovače. NetFlow mal za úlohu poskytnúť schopnosť zbierať informácie o IP premávke (traffic), ako vstupuje alebo opúšťa rozhranie. NetFlow môže slúžiť napríklad sieťovému administrátorovi a to práve ako analýza premávky, z ktorej vie administrátor získať informácie ako napríklad:

- Zdrojovú adresu
- Cieľovú adresu
- Typ servisu
- Počet paketov
- Počet paketov konkrétneho protokolu
- Atď..

NetFlow je dobrá pomôcka pre zistenie problému premávky, ktorý môže viesť k zaťaženiu siete.

NetFlow obsahuje 3 hlavné komponenty:

1. **Flow exporter:** agreguje pakety do takzvaných flows a exportuje *flow* záznamy na jeden alebo viac zberačov (collectors).
2. **Flow collector:** je zodpovedný za ukladanie, pred-spracovanie pozbieraných flow dát
3. **Analysis application** (aplikácia analýzy): analyzuje pozbierané *flow* dáta napríklad v kontexte detekcie narušenia, alebo premávkového profilovania.

# Popis protokolu

Sieťové smerovače a prepínače, ktoré podporujú NetFlow, dokážu pozbierať IP premávku na všetkých rozhraniach, kde je NetFlow povolený. Neskôr dokážu exportovať štatistiky ako NetFlow záznamy naprieč (aspoň jedným) zberačom (collector) – typicky sa jedná o server, ktorý túto analýzu vykonáva.

# Network Flows

Cisco štandard NetFlow verzia 5 popisuje *flow* ako jednosmernú sekvenciu paketov, ktoré zdieľajú všetkých sedem hodnôt, ktoré svojou kombináciou definujú unikátny kľúč pre daný *flow*.
Skladá sa z:

1. Vstupné rozhranie (SNMP ifIndex)
2. Zdrojová IP adresa
3. Cieľová IP adresa
4. IP protokol
5. Zdrojový port (pre UDP alebo TCP), inak 0
6. Cieľový port (pre UDP alebo TCP, typ a kód pre ICMP), inak 0
7. IP typ servisu

Táto definícia *flow*-u sa používa taktiež pre IPv6 a podobná definícia je použitá pre MPLS a Ethernet-ové *flows*.

Typický výstup NetFlow nástroja vyzerá ako na obrázku nižšie:

```
Date flow start          Duration Proto   Src IP Addr:Port      Dst IP Addr:Port     Packets    Bytes Flows
2010-09-01 00:00:00.459     0.000 UDP     127.0.0.1:24920   ->  192.168.0.1:22126        1       46     1
2010-09-01 00:00:00.363     0.000 UDP     192.168.0.1:22126 ->  127.0.0.1:24920          1       80     1
```

*Obrázok 1 Výstup NetFlow analýzy*

# Export záznamov

Sieťový smerovač exportuje *flow* záznam, keď je daný *flow* dokončený. Toto dokončenie je spôsobené takzvaným *flow aging*, čo v preklade znamená, starnutie daného *flow*-u. Keď smerovač vidí, že prichádza nová premávka pre daný *flow*, časovač sa vynuluje a začne sa počítanie starnutia na novo. Smerovač je často krát nastavený tak, aby exportoval záznam po uplynutí pevne daného času, aj napriek tomu, že *flow* je stále aktívny (dostatočne mladý).

# Hlavička paketu

Všetky NetFlow pakety začínajú s hlavičkou, ktorá je závislá na verzií použitého NetFlow. Hlavička obsahuje aspoň tieto polia:

- Verzia NetFlow (v1, v5, v7, v8, v9..)
- Sekvenčné číslo, ktoré zisťuje straty a duplikáty *flows*
- Čas v momente exportu paketu a to napríklad aktuálny systémový čas alebo absolútny čas
- Počet flow záznamov (v5 alebo v8) alebo list šablón a záznamov (v9)

Hlavička paketu pre (v5):

| Bytes | Contents | Description |
|---|---|---|
| 0-1 | version | NetFlow export format version number |
| 2-3 | count | Number of flows exported in this packet (1-30) |
| 4-7 | SysUptime | Current time in milliseconds since the export device booted |
| 8-11 | unix_secs | Current count of seconds since 0000 UTC 1970 |
| 12-15 | unix_nsecs | Residual nanoseconds since 0000 UTC 1970 |
| 16-19 | flow_sequence | Sequence counter of total flows seen |
| 20 | engine_type | Type of flow-switching engine |
| 21 | engine_id | Slot number of the flow-switching engine |
| 22-23 | sampling_interval | First two bits hold the sampling mode; remaining 14 bits hold value of sampling interval |

*Obrázok 2 Hlavička paketu NetFlow(v5)*

# Záznam (record)

NetFlow záznam môže obsahovať široké spektrum informácií o premávke. Najčastejšie používaná NetFlow verzia 5 (v5) obsahuje nasledovné:

- Vstupné rozhranie
- Výstupné rozhranie
- Čas začiatku a konca *flow*-u v milisekundách
- Počet bajtov a paketov, ktoré boli pozbierané
- Zdrojová a cieľová IP adresa
- IP protokol
- IP typ servisu
- ICMP typ a kód
- Zdrojový a cieľový port
- Pre TCP *flow* aj TCP flags, ktoré boli pozbierané za život *flow*-u
- IP adresa nexthop (nasledujúca)
- Zdrojová a cieľová IP maska siete

| Bytes | Contents | Description |
|---|---|---|
| 0-3 | srcaddr | Source IP address |
| 4-7 | dstaddr | Destination IP address |
| 8-11 | nexthop | IP address of next hop router |
| 12-13 | input | SNMP index of input interface |
| 14-15 | output | SNMP index of output interface |
| 16-19 | dPkts | Packets in the flow |
| 20-23 | dOctets | Total number of Layer 3 bytes in the packets of the flow |
| 24-27 | First | SysUptime at start of flow |
| 28-31 | Last | SysUptime at the time the last packet of the flow was received |
| 32-33 | srcport | TCP/UDP source port number or equivalent |
| 34-35 | dstport | TCP/UDP destination port number or equivalent |
| 36 | pad1 | Unused (zero) bytes |
| 37 | tcp_flags | Cumulative OR of TCP flags |
| 38 | prot | IP protocol type (for example, TCP = 6; UDP = 17) |
| 39 | tos | IP type of service (ToS) |
| 40-41 | src_as | Autonomous system number of the source, either origin or peer |
| 42-43 | dst_as | Autonomous system number of the destination, either origin or peer |
| 44 | src_mask | Source address prefix mask bits |
| 45 | dst_mask | Destination address prefix mask bits |
| 46-47 | pad2 | Unused (zero) bytes |

*Obrázok 3 Záznam (flow) paketu NetFlow(v5)*

# Návrh aplikácie

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

Navrhnutá aplikácia musí vedieť pracovať s týmito vstupmi:

-f <file> jméno analyzovaného souboru nebo STDIN,

-c <neflow_collector:port> IP adresa, nebo hostname NetFlow kolektoru. volitelně i UDP port (127.0.0.1:2055, pokud není specifikováno),

-a <active_timer> - interval v sekundách, po kterém se exportují aktivní záznamy na kolektor (60, pokud není specifikováno),

-i <seconds> - interval v sekundách, po jehož vypršení se exportují neaktivní záznamy na kolektor (10, pokud není specifikováno),

-m <count> - velikost flow-cache. Při dosažení max. velikosti dojde k exportu nejstaršího záznamu v cachi na kolektor (1024, pokud není specifikováno).

Všetky parametre sú brané ako voliteľné a majú predvolenú hodnotu uloženú v kóde aplikácie.

Aplikácia je implementovaná v jazyku C++. Zo zadania vyplýva, že aplikácia musí vedieť čítať vstupný *.pcap* súbor, v ktorom je uložená IP premávka. Pokiaľ nebol zadaný vstupný súbor, aplikácia musí vedieť čítať premávku zo štandardného vstupu. Aplikácia sa musí pripájať na zberač (collector) s adresou a portom zadanou užívateľom, alebo predvolene na adresu: 127.0.0.1:2055. Aplikácia by mala vedieť preložiť *hostname* na IP adresu. Aplikácia obsahuje *flow_cache*, ktorá je obmedzená a obmedzuje počet uložených *flows* v rámci aplikácie. Aplikácia kontroluje intervaly, v ktorých sú *flows* stále „živé“ a je možné ich agregovať.

Na čítanie *.pcap* súborov využíva aplikácia hlavičkový súbor *libpcap,* ktorá dokáže čítať jednotlivé pakety v rámci premávky a pomocou *callback* funkcie *paketHandler()* vie aplikácia jednotlivé pakety spracovať. V tejto funkcií pozbiera informácie o pakete a rozhodne, či vytvoriť nový *flow* záznam pre daný paket, alebo ak už *flow* záznam pre tento paket existuje, agreguje ich. Toto rozhodovanie robí na základe porovnávania *flow unique keys*. Ak sa kľúče zhodujú, prebehne agregácia *flow* záznamov. Tieto záznamy aplikácia musí vedieť uložiť do spomínanej *flow_cache*. Na toto som použil štruktúru obojsmerného listu. Do listu ukladám aktuálne záznamy v poradí, v ktorých prišli (zachovávanie časových značiek), takže výstupom je list, ktorý na prvej pozícií obsahuje najstarší záznam, čo sa hodí pre neskorší export najstarších záznamov na zberač.

V rámci aplikácie som definoval niekoľko štruktúr, ktoré mi uľahčili prácu s dátami paketov. Štruktúra *NetFlowHeader*, ktorá obsahuje informácie hlavičky, ktoré budem prikladať ku každému NetFlow paketu, ktorý budem exportovať na zberač. Štruktúra *NetFlowRecord,* ktorá obsahuje informácie o danom *flow* zázname. Tieto informácie budem prikladať do paketu, rovno za hlavičku, ktorý budem exportovať. *Flow_cache,* ktorá je v podobe listu, obsahuje ako hodnoty práve štruktúru *NetFlowRecord*. Štruktúra *NetFlowGlobalVariables*, ktorá obsahuje informácie, ku ktorým treba mať prístup a meniť ich za chodu programu na viacerých miestach. V tejto štruktúre je napríklad uložená hlavička (*NetFlowHeader*), list *NetFlowRecord (flow_cache)*, adresa kolektoru a ďalšie pomocné premenné.

# Implementácia

## Začiatok Main()

Na začiatku programu vo funkcií *Main()* ošetrím vstupné argumenty. Na toto som použil *getOpt()*, ktorý šikovne ošetrí argumenty podľa zadanej predlohy. Vstupné argumenty ukladám do premenných. Vytvorím štruktúru adresy pre zberač. Ďalej vytvorím *buffer*, ktorému alokujem miesto v pamäti. Vytvorím štruktúru *NetFlowGlobalVariables*, do ktorej uložím všetky tieto premenné, ktoré budem ďalej využívať.

### PCAP konfigurácia

Vytvorím *pcap descriptor* pre prácu s hlavičkovým súborom *libpcap*. Skúsim otvoriť vstupný *.pcap* súbor. Ak niečo v tomto procese zlyhá, vypíšem chybu na štandardný chybový výstup. Ak vstupný súbor nebol užívateľsky zadaný, začnem čítať zo štandardného vstupu. V tomto momente začnem prijímať pakety a čítať z nich informácie pomocou funkcie *pcap_loop()*, do ktorej posielam aj *callback* funkciu *packetHandler()*.

```
///LIBPCAP LOGIC
pcap_t *descr;
char errorBuffer[PCAP_ERRBUF_SIZE];

///IF FILE FLAG IS TRUE OPEN FILE WITH GIVEN NAME
if(file_flag){
    descr = pcap_open_offline(file_name.c_str(), errorBuffer);

    if(descr == NULL){
    fprintf(stderr, "Something went wrong with the file \"%s\".\nError: %s\nExiting...\n", file_name.c_str(), errorBuffer);
    exit(EXIT_FAILURE);
    }

}else{
    ///ELSE OPEN STDIN AND READ PACKETS FROM THERE
    descr = pcap_fopen_offline(stdin, errorBuffer);

    if(descr == NULL){
    fprintf(stderr,"Something went wrong with reading standard input\nError: %s\nExiting...\n", errorBuffer);
    exit(EXIT_FAILURE);
    }
}

///START READING PACKETS FROM FILE/STDIN
if(pcap_loop(descr, 0, packetHandler, (u_char*)global) < 0 ){
    fprintf(stderr, "Live capture failed!\nError: %s\nExiting...\n", pcap_geterr(descr));
    exit(EXIT_FAILURE);
}
///READ ALL PACKETS FROM INPUT///
```

*Obrázok 4 Pcap konfigurácia*

Vo funkcií *packetHandler()* si deklarujem premenné, do ktorých ukladám informácie o pakete, potrebné pre NetFlow. Vypočítam čas pre NetFlowHeader. A zavolám funkciu *createOrAggregate()*.

# Záznamy (flows)

## Agregácia záznamov

Vo funkcií *createOrAggregate()* prejdem každý *flow* vo *flow_cache* a porovnám kľúče. Ak som našiel už existujúci *flow*, agregujem aktuálne čítaný vstupný paket do tohto flowu a skontrolujem, či sa nejedná o TCP paket, ktorý obsahuje TCP príznak FIN alebo RST. Ak obsahuje, daný *flow* exportujem. Zároveň v tomto prechode všetkými *flows* kontrolujem, či daný *flow* ešte stále „žije“. Ak nie, exportujem ho. Ak sa kľúč nezhoduje so žiadnym *flow*-om, vytvorím nový zavolaním funkcie *createNewFlow()*.

## Vytvorenie záznamu

Vo Funkcií *createNewFlow()* vytvorím objekt nového *flow*, do ktorého uložím informácie z aktuálne čítaného paketu. Pokiaľ je *flow_cache* plná, uvoľním miesto exportovaním najstaršieho *flow*-u. Opäť skontrolujem, či sa nejedná o TCP paket s príznakom FIN alebo RST. Ak nie pridávam tento *flow* do *flow_cache* na koniec (najnovší).

# Buffer (odosielaný paket)

Buffer (paket), ktorý sa bude odosielať na zberač je pole znakov o veľkosti maximálnej dĺžky NetFlow paketu. NetFlow paket verzie 5 obsahuje hlavičku paketu o veľkosti 24 bajtov a za ňu sa priliepajú flow záznamy o veľkosti 48 bajtov. Týchto záznamov môže byť v jednom pakete maximálne 30. Preto som si definoval konštanty *HEADER_LENGTH = 24*, *FLOW_LENGTH = 48* a *MAX_FLOWS_IN_PACKET = 30*. Preto aj veľkosť buffer-u je rovnica: Veľkosť = *HEADER_LENGHT + (FLOW_LENGTH * MAX FLOWS_IN_PACKET) + 1*. Plus jedna na koniec, pretože sa jedná o pole znakov, ktoré je korektne ukončené znakom '\0'.

## Hlavička paketu

Funkcia *addHeaderToPacket()* rieši zápis informácií z NetFlow hlavičky do buffer-u. Posúvam ukazovateľ buffer-u o X bajtov, podľa tabuľky obrázka č.2.

```
void addHeaderToPacket(NetFlowGlobalVariables &global){

    *(uint16_t*)global.buffer = htons(5);
    *(uint16_t*)(global.buffer + 2) = htons(global.header.count);
    *(uint32_t*)(global.buffer + 4) = htonl(global.header.SysUptime);
    *(uint32_t*)(global.buffer + 8) = htonl(global.header.unix_secs);
    *(uint32_t*)(global.buffer + 12) = htonl(global.header.unix_nsecs);
    *(uint32_t*)(global.buffer + 16) = htonl(global.header.flow_sequence);
    *(uint8_t*)(global.buffer + 20) = global.header.engine_type;
    *(uint8_t*)(global.buffer + 21) = global.header.engine_id;
    *(uint16_t*)(global.buffer + 22) = htons(global.header.sampling_interval);
}
```

*Obrázok 5 Funkcia addHeaderToPacket()*

## Telo paketu (flows)

Funkcia *addFlowToPacket()* rieši zápis informácií z odosielaného / odosielaných *flows* do buffer-u. Princíp je rovnaký ako pri hlavičke paketu, s tým rozdielom, že nad touto funkciou je možná iterácia. Iterácia je podporovaná vďaka premennej *offset*, ktorá rieši posuv o násobky *flows*. Ak posielam v jednom pakete len jeden *flow*, *offset* bude rovný 0, čo značí, že zapisujem do buffer-u hneď za hlavičku. Ak by som posielal 30 *flows* v jednom pakete, pre každý paket *offset* určí, koľko bajtov sa má preskočiť, aby sa zapisovalo hneď *flow* za *flow*-om.

```
void addFlowToPacket(NetFlowGlobalVariables &global, NetFlowRecord &r, int offset){
    *(uint32_t*)(global.buffer + HEADER_LENGHT + offset*FLOW_LENGHT)= htonl(r.srcaddr);
    *(uint32_t*)(global.buffer + 4 + HEADER_LENGHT + offset*FLOW_LENGHT) = htonl(r.dstaddr);
    *(uint32_t*)(global.buffer + 8 + HEADER_LENGHT + offset*FLOW_LENGHT) = htonl(r.nexthop);
    *(uint16_t*)(global.buffer + 12 + HEADER_LENGHT + offset*FLOW_LENGHT) = htons(r.input);
    *(uint16_t*)(global.buffer + 14 + HEADER_LENGHT + offset*FLOW_LENGHT) = htons(r.output);
    *(uint32_t*)(global.buffer + 16 + HEADER_LENGHT + offset*FLOW_LENGHT) = htonl(r.dPkts);
    *(uint32_t*)(global.buffer + 20 + HEADER_LENGHT + offset*FLOW_LENGHT) = htonl(r.dOctets);
    *(uint32_t*)(global.buffer + 24 + HEADER_LENGHT + offset*FLOW_LENGHT) = htonl(r.First);
    *(uint32_t*)(global.buffer + 28 + HEADER_LENGHT + offset*FLOW_LENGHT) = htonl(r.Last);
    *(uint16_t*)(global.buffer + 32 + HEADER_LENGHT + offset*FLOW_LENGHT) = htons(r.srcport);
    *(uint16_t*)(global.buffer + 34 + HEADER_LENGHT + offset*FLOW_LENGHT) = htons(r.dstport);
    *(uint8_t*)(global.buffer + 36 + HEADER_LENGHT + offset*FLOW_LENGHT) = r.pad1;
    *(uint8_t*)(global.buffer + 37 + HEADER_LENGHT + offset*FLOW_LENGHT) = r.tcp_flags;
    *(uint8_t*)(global.buffer + 38 + HEADER_LENGHT + offset*FLOW_LENGHT) = r.prot;
    *(uint8_t*)(global.buffer + 39 + HEADER_LENGHT + offset*FLOW_LENGHT) = r.tos;
    *(uint16_t*)(global.buffer + 40 + HEADER_LENGHT + offset*FLOW_LENGHT) = htons(r.src_as);
    *(uint16_t*)(global.buffer + 42 + HEADER_LENGHT + offset*FLOW_LENGHT) = htons(r.dest_as);
    *(uint8_t*)(global.buffer + 44 + HEADER_LENGHT + offset*FLOW_LENGHT) = r.src_mask;
    *(uint8_t*)(global.buffer + 45 + HEADER_LENGHT + offset*FLOW_LENGHT) = r.dst_mask;
    *(uint16_t*)(global.buffer + 46 + HEADER_LENGHT + offset*FLOW_LENGHT) = htons(r.pad2);
}
```

*Obrázok 6 Funkcia addFlowToPacket()*

# Export

Funkcia *exportFlow()* slúži na poslanie jedného *flow*-u. Pridá sa hlavička do buffer-u, pridá sa jeden *flow* do buffer-u. UDP paket sa pošle na zberač pomocou funkcie *sendUDPPacket()*.

## Posielanie paketu

Funkcia *sendUDPPacket()* rieši vytvorenie *socket*-u, pre ktorý sa priradí adresa zberača. Na *socket* posielam obsah buffer-u pomocou funkcie *sendto()*. Po úspešnom poslaní sa *socket* zavrie.

```
void sendUDPPacket(sockaddr_in address, char *buffer, unsigned short bytes)
{
    int sock;

    if ((sock = socket(AF_INET, SOCK_DGRAM, 0)) == -1)
    {
        fprintf(stderr, "Socket failed!\n");
        exit(EXIT_FAILURE);
    }
    if (sendto(sock, buffer, bytes, 0, (const struct sockaddr*)&address, sizeof(ip)) == -1)
    {
        fprintf(stderr, "Send to failed!\n");
        exit(EXIT_FAILURE);
    }
    close(sock);
}
```

*Obrázok 7 Funkcia sendUDPPacket()*

# Koniec Main()

Pokiaľ vo *flow_cache* zostanú uložené *flows*, ktoré sa do teraz neexportovali pomocou starnutia, alebo uvoľnenia *flow_cache*, poprípade TCP príznakmi FIN alebo RST a *pcap_loop()* ukončil čítanie prichádzajúcich paketov, treba všetky exportovať na zberač. Tu začnem prechádzať všetky uložené *flows* z *flow_cache* pomocou *for()* cyklu. Každý prečítaný *flow* (začínam od najstarších) uložím do buffer-u, do kým nenatrafím na koniec *flow_cache* alebo som nenarátal 30 *flows* na buffer-i. Ak som narátal 30 *flows*, pošlem ich spolu v jednom pakete a pokračujem ďalej v iterácií. Výsledkom je posielanie paketov s čo najviac *flows* v rámci jedného paketu.

```
///FLOWS THAT ARE STILL ACTIVE WILL BE SEND TO THE COLLECTOR BECAUSE THERE IS NO PACKET LEFT FOR US TO READ FROM INPUT
///ITERATION THROUGH STILL ACTIVE FLOWS
int i = 0;
std::list<NetFlowRecord>::iterator it;
for(it = global->listOfFlows.begin(); it != global->listOfFlows.end(); ++it){

    ///GET FLOW AND ADD IT TO THE BUFFER
    addFlowToPacket(*global, *it, i);
    uint32_t addOne = 1;
    global->header.flow_sequence = global->header.flow_sequence + addOne;
    ///REMOVE FLOW FROM LIST OF FLOWS
    it = global->listOfFlows.erase(it);
    --it;

    ///IF THERE IS 30 FLOWS IN BUFFER ALREADY
    if(i == 29){
        ///MAKE HEADER FOR 30 FLOWS
        uint16_t maxFlows = 30;
        global->header.count = maxFlows;

        ///SEND PACKET WITH 30 FLOW RECORDS TO THE COLLECTOR
        addHeaderToPacket(*global);
        sendUDPPacket(global->servaddr, global->buffer, HEADER_LENGHT + (FLOW_LENGHT * MAX_FLOWS_IN_PACKET));
        ///CLEAR BUFFER
        memset(global->buffer, '\0', HEADER_LENGHT + (FLOW_LENGHT * MAX_FLOWS_IN_PACKET) + 1);

        ///RESET COUNTER (AFTER THIS SCOPE i WILL BY INCREASED BY 1 TO START FROM 0 AGAIN)
        i = -1;
    }
    i++;
}

///i REPRESENTS FLOWS THAT LEFT IN FLOW CACHE (LIST OF FLOWS)
if(i > 0){
    ///MAKE HEADER FOR i FLOWS
    uint16_t leftovers = i;
    global->header.count = leftovers;
    addHeaderToPacket(*global);

    ///SEND PACKET WITH i FLOW RECORDS TO THE COLLECTOR
    sendUDPPacket(global->servaddr, global->buffer, HEADER_LENGHT + (FLOW_LENGHT * leftovers));
    ///CLEAR BUFFER
    memset(global->buffer, '\0', HEADER_LENGHT + (FLOW_LENGHT * MAX_FLOWS_IN_PACKET) + 1);
}
```

*Obrázok 8 Export zvyšných (flow) záznamov*

# Návod na spustenie

```
FLOW(1)                    General Commands Manual                   FLOW(1)

NAME
       flow  -  Netflow  exportér  verzia 5, ktorý posiela na zberač zachytené
       záznamy z premávky.

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

DESCRIPTION
       flow  -  NetFlow  exportér,  ktorý číta vstupný .pcap súbor alebo štan-
       dardný vstup. Úlohou  je  analyzovať  a  exportovať  NetFlow  flows  na
       zberač.

OPTIONS
       -f     <file> meno analyzovaného .pcap súboru alebo štandarný vstup,

       -c     <neflow_collector:port>  IP  adresa  a  port zberača (predvolene
              127.0.0.1:2055),

       -a     <active_timer> Interval v  sekundách,  po  ktorom  sa  exportujú
              aktívne záznamy na zberač (predvolene 60),

       -i     <inactive_timer>  Interval  v sekundách, po jeho vypršaní sa ex-
              portujú neaktívne záznamy na zberač (predvolene 10),

       -m     <count> Veľkosť flow_cache. Pri dosiahnutí  maximálnej  veľkosti
              dôjde  k  exportu najstaršieho záznamu v cachi na zberač(predvo-
              lene 1024)

AUTHOR
       František Fúdor (xfudor00)

                                                                      FLOW(1)
```

*Obrázok 9 Návod na spustenie*

# Záver

Oboznámil som sa s analýzou a štatistikami IP paketov za pomoci NetFlow(v5). Projekt bol z môjho hľadiska stavaný viacej na štúdium problematiky ako programovania. Ako pomôcku som využil aj kreslenie štruktúry paketov pre ľahšie pochopenie.

# Použitá literatúra


NetFlow - Wikipedia. [online]. Dostupné z: https://en.wikipedia.org/wiki/NetFlow

NetFlow Export Datagram Format - Cisco. *Cisco - Networking, Cloud, and Cybersecurity Solutions* [online]. Dostupné z: https://www.cisco.com/c/en/us/td/docs/net_mgmt/netflow_collection_engine/3-6/user/guide/format.html#wp1003394

std::list - cppreference.com. *301 Moved Permanently* [online]. Dostupné z: https://en.cppreference.com/w/cpp/container/list

UDP Klient – VUT FIT Doc. Ing. Petr Matouˇsek, Ph.D., M.A [online]. Dostupné z: https://moodle.vut.cz/pluginfile.php/502879/mod_resource/content/2/isa-sockets.pdf

Offline packet capture analysis with C/C++ & libpcap - Tony Lukasavage. *Tony Lukasavage* [online]. Copyright © 2015 [cit. 14.11.2022]. Dostupné z: http://tonylukasavage.com/blog/2010/12/19/offline-packet-capture-analysis-with-c-c----amp--libpcap/?fbclid=IwAR2aO4LUfUetD-6O4F58l71fogAqvqWmftwNh2sIKwgRpnMH02J-5R3RyaY

ntohs(3) - Linux man page. *Linux Documentation* [online]. Dostupné z: https://linux.die.net/man/3/ntohs

Home | TCPDUMP & LIBPCAP. *Home | TCPDUMP & LIBPCAP* [online]. Copyright © 2010 [cit. 14.11.2022]. Dostupné z: https://www.tcpdump.org/