Vysoké učení technické v Brně
Fakulta informačních technologií

Sieťové aplikácie a správa sietí

Generovanie NetFlow dát zo zachytenej sieťovej komunikácie

31. októbra 2022 Adam Zvara (xzvara01)

Obsah

1 Úvod

Cieľom tohoto projektu je implementácia NetFlow exportéra, ktorý na vstupe prijíma sieťové dáta uložené vo forme *pcap* súborov a vytvára z nich *NetFlow* záznamy, ktoré sú odoslané kolektoru na prípadné ďalšie spracovanie.

Tento manuál je logický členený na niekoľko častí. Sekcia 2 sa zaoberá samotným protokolom NetFlow ku ktorému patrí definícia internetového toku, popis typickej monitorovacej architektúry a štruktúrou exportovaného záznamu. V sekcii 3 je popísaný design vytvorenej aplikácie, ktorý predstavuje dekompozíciu problematiky exportu dát do logických celkov, ako sú implementované v jednotlivých súboroch odovzdaného riešenia. Sekcia 4 sa venuje implementačným detailom, prezentuje hlavné časti aplikácie a problémy pri jej riešení. Sekcia 5 je zameraná na testovanie aplikácie a v poslednej sekcii 6 sú príklady jej použitia.

2 NetFlow

NetFlow [3] je protokol spoločnosti Cisco[1], ktorý sa používa na získavanie a ukladanie informácií o internetovej komunikácií medzi 2 koncovými stanicami. Základným pojmom spojeným s týmto protokolom je tok. Internetový tok [1] (angl. *internet flow*) je definovaný ako jednosmerná postupnosť paketov prechádzajúca bodom pozorovania počas určitého časového intervalu, kde všetky pakety patriace rovnakému toku majú spoločnú množinu vlastností. Tieto vlastnosti môžu byť položky získané priamo z transportných hlavičiek alebo vlastnosti z nich odvodené. Existujú spoločné vlastnosti pre všetky pakety rovnakého toku (napr. zdrojová/cieľová IP adresa, zdrojový/cieľový port, číslo protokolu) a vlastnosti vypočítané, odvodené alebo prítomné len v určitých paketoch (napr. celkový počet prenesených bytov, doba kedy bol videný prvý a posledný paket).

2.1 Monitorovacia architektúra

Monitorovacia architektúra [4] NetFlow je zložená z nasledujúcich častí.

Exportér slúži na zachytávanie informácií o tokoch v sieti. Najčastejšie sa jedná priamo o smerovač alebo špeciálne sondy napr. nprobe, Flowmon. Práve táto časť (v zjednodušenej podobe) je hlavnou náplňou tohoto projektu.

Komunikačný protokol určuje štruktúru záznamov prenášaných medzi exportérom a kolektorom. Medzi najpopulárnejšie patria NetFlow (v5/v9), IPFIX, sFlow. Vrámci tohoto projektu sa budeme ďalej zaoberať protokolom NetFlow v5.

Kolektor má za úlohu ukladať záznamy, ktoré získava od exportéra. Medzi populárne kolektory radíme Nfdump alebo CESNETom vyvíjaný ipfixcol2.

Analýza zachytenej komunikácie prebieha po uložení dát v kolektore. Týchto aplikácií je celá škála a za spomenutie stoja aplikácie Nfsen alebo Nemea.

2.2 Štruktúra NetFlow záznamov

Exportované NetFlow dáta podliehajú jednému z niekoľkých predpísaných formátov[2]. Ako bolo spomenuté, tento projekt je zameraný na protokol NetFlow v5, ktorého formát je nasledovný:

Bytes	Content	Popis
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

Tabuľka 1: Formát hlavičky NetFlow v5 paketu

[1] https://www.cisco.com/
[2] NetFlow Export Datagram Format

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

Tabuľka 2: Formát záznamu NetFlow v5 paketu

NetFlow v5 je v porovnaní s modernejšími formátmi jednoduchý čo zjednodušuje aj jeho implementáciu. Na druhú stranu kvôli tomu prichádzame o toky obsahujúce IPv6 adriesy alebo o možnosť rozšírenia získavaných informácií o nové/zložitejšie elementy (ako je to možné v protokoloch využívajúcich šablóny, napr. NetFlow v9, IPFIX). Výsledný NetFlow paket pozostáva z hlavičky za ktorou nasleduje niekoľko záznamov so štatistikami o jednotlivých tokoch.

3 Návrh aplikácie

Návrh aplikácie začína dekompozíciou programu na podproblémy. Túto dekompozíciu znázorňuje rozdelenie aplikácie do niekoľkých súborov.

```
src
├── parse.cpp/hpp
├── capture.cpp/hpp
└── store.cpp/hpp
test
└── ...
flow.cpp/hpp
...
```

parse.cpp
Aplikácia na vstupe umožňuje užívateľovi zadávať rôzne parametre jej spustenia (adresa kolektoru, vstupný pcap súbor ...). Úlohou tohoto súboru je spracovať tieto parametre do internej štruktúry, ktorá je ďalej používaná v iných častiach aplikácie. Taktiež tu prebieha rezolúcia mena kolektoru na IP adresu.

capture.cpp
Po spracovaní parametrov nasleduje čítanie paketov zo vstupného pcap súboru. To je realizované pomocou knižnice libpcap. Získané pakety sú uložené v triede `Capture`, ktorá si okrem obsahu aktuálneho paketu uchováva referenciu na otvorený pcap súbor alebo použitý filter.

store.cpp
Obsahuje najdôležitejšiu časť programu, ktorou je vytváranie tokov, ich ukladanie a exportovanie. Okrem toho sa tu nachádza aj odosielanie vytvorených záznamov.

flow.cpp
Súbor `flow.cpp` spája funkcionalitu ostatných častí programu.

4 Implementácia

Táto sekcia obsahuje podrobnejší popis implementácie exportéra.

4.1 Spracovanie vstupu od užívateľa

Vstupné argumenty sú spracované pomocou funkcie `getopt`. Následne sú uložené v štruktúre `arguments`:

```
struct arguments {
    FILE* pcapfile          = stdin;            // pcap file
    std::string collector   = DEF_COLLECTOR;    // collector hostname
    sockaddr_storage address;                   // collector IP address (both IPv4 and IPv6)
    uint16_t port           = DEF_PORT;         // port number
    uint16_t active         = DEF_ACTIVE;       // active timer
    uint16_t inactive       = DEF_INACTIVE;     // inactive timer
    uint32_t cache_size     = DEF_COUNT;        // flow cache size
}
```

Táto štruktúra obsahuje všetky potrebné informácie v ďalších častiach aplikácie, ktoré obsahujú pri inicializácií predvolené hodnoty, ktoré sú po spracovaní argumentov aktualizované.

Prevedenie hostname kolektoru na IP adresu je realizované funkciou `getddrinfo`, ktorá vráti spojový zoznam adries prislúchajúcich k danému menu. Jednoduchým spôsobom by sme mohli vziať prvú adresu a používať ju v programe ďalej, no v prípade, že táto adresa nie je funkčná, nebude možné exportovať dáta. Preto už pri tejto konverzii dochádza ku testovaciemu pripojeniu na adresy v tomto zozname, až kým sa nám nepodarí pripojiť na nejakú z nich.

Ďalším problémom bolo možné **zadanie portu kolektoru** (`-c 127.0.0.1:1234`), pretože nie je jasne definovaný spôsob zápisu portu pre IPv6 adresy. Podľa RFC 5952 [2] je odporúčaný spôsob zápisu portu pri IPv6 adrese pomocou znakov "[]". Príklad zadania IPv6 adresy 2600:: s portom 123 by bol `[2600::]:123`. Tento projekt podporuje tento spôsob zápisu adries, ako aj zápis typu `-c 2600:::1234`, ktorý by predstavoval adresu `2600::` s portom `1234`.

4.2 Zachytávanie paketov

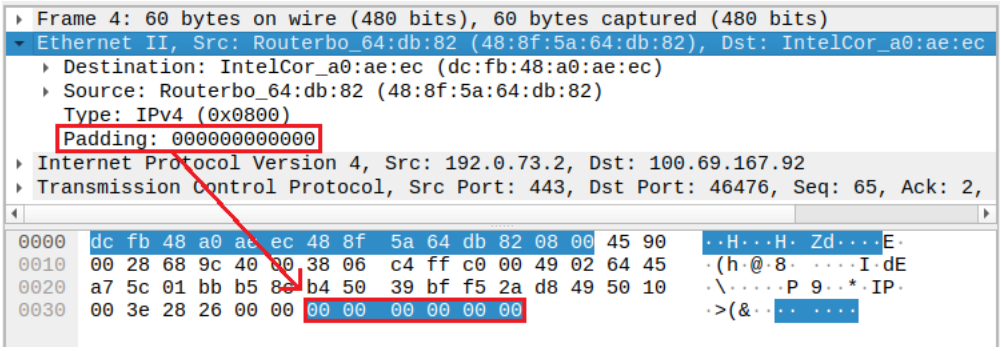
Ako bolo spomenuté v sekcii 3, pakety sú zachytávané pomocou knižnice libpcap a uložené v triede `Capture`.

```
class Capture {
    ...
    public:
        const unsigned char *raw_pkt;       // captured raw packet data
        pcap_pkthdr *header;                // captured packet header
        ether_header *eth_header;           // ethernet header
        iphdr *ip_header;                   // ip header
        void *transport_header;             // trasnport header (later casted to the correct type)
    ...
}
```

Zabezpečenie toho, že pakety pochádzajú z (TCP, UDP, ICMP) komunikácie je spravené pomocou aplikovania filtru na prichádzajúce pakety. Filter je zadaný v BPF syntaxi (Berkeley Packet Filter) a má podobu: `ip proto 1 or ip proto 6 or ip proto 17`.

Práca s transportnou hlavičkou bol jeden z problémov, na ktorý som narazil pri implementácii. Hlavičky rôznych transportných vrstiev majú rôznu štruktúru a veľkosť, napriek tomu bol žiadúci uniformný prístup k tejto hlavičke. Riešením toho bolo vytvorenie ukazateľu `void *transport_header`, ktorý je nutné pretypovať na konkrétny typ hlavičky pred jeho použitím.

Zistenie veľkosti paketu tiež nie je na prvý pohľad úplne jednoznačné, pretože tam existuje viacero možností. 2 možnosti priamo pochádzajú z hlavičky `pcap_pkthdr *header`, ktorá obsahuje položky `caplen` a `len`. Položka `caplen` je dĺžka zachyteného paketu, čo v niektorých prípadoch nemusí byť rovnaká s reálnou dĺžkou paketu. Položka `len` sa zdá byť teda vhodným kandidátom. Narážame tu avšak na problém, kedy ethernetová hlavička môže obsahovať časti, ktoré sa nachádzajú na úplnom konci paketu (trailing, padding) a tým pádom zväčšujú aj veľkosť `len`.



Najvhodnejším spôsobom teda je vyčítať veľkosť paketu priamo z IPv4 hlavičky.

4.3 Ukladanie vytvorených záznamov

Po spracovaní paketu sa zisťuje, či je nutné vytvoriť nový záznam o toku alebo aktualizovať hodnoty existujúceho záznamu. Záznamy sú uložené v asociatívnom poli (std::map), kde kľúčom je šestica identifikujúca tok.

```
/**                              Flow cache key                                    */
//                  SRCIP       DSTIP    SRC_PORT  DST_PORT   PROTO     TOS
typedef std::tuple<uint32_t, uint32_t, uint16_t, uint16_t, uint8_t, uint8_t> netkey_t;}
```

Keďže formát NetFlow v5 obsahuje z historických dôvodov informácie, ktoré dnes nie sme schopní z komunikácie samotnej určiť, niektoré políčka NetFlow v5 záznamu sú nastavené na 0. Konkrétne sa jedná o položky:

NetFlow v5 Header	
engine_type	Type of flow-switching engine
engine_id	Slot number of the flow-switching engine
sampling_interval	First two bits hold the sampling mode; remaining 14 bits sampling interval
NetFlow v5 Record	
nexthop	IP address of next hop router
input	SNMP index of input interface
output	SNMP index of output interface
src_as	Autonomous system number of the source, either origin or peer
dst_as	Autonomous system number of the destination, either origin or peer
src_mask	Source address prefix mask bits
dst_mask	Destination address prefix mask bits

Zisťovanie časových údajov bolo ovplyvnené faktom, že nezachytávame pakety v reálnom čase ale zo súboru. To sa prejavilo na položkách SysUpTime, First a Last. Položka **SysUpTime** predstavuje časový rozdiel exportu toku a času, od ktorého exportér začal pracovať. V našom prípade je to rozdiel od času prvého paketu. Položky **First** a **Last** predstavujú SysUpTime v prvom, resp. poslednom bode aktualizácie toku. Teda **úplne prvý tok, ktorý vytvoríme bude mať stále položku First rovnú 0**.

4.4 Exportovanie

Poslednou úlohou exportéra je posielanie vytvorených tokov na kolektor. Ku exportu dochádza v prípade, že je splnená aspoň jedna z nasledujúcich podmienok:

1. došlo ku vypršaniu **aktívneho časovača** (pre príliš dlhé toky, napr. sťahovanie veľkých súborov)
2. došlo ku vypršaniu **neaktívneho časovača** (ak po jeho dobu nebol tok aktualizovaný)
3. pri naplnení `flow_cache` dochadza ku exportu najstaršieho záznamu
4. pri detekcii príznakov *FIN* alebo *RST* pri TCP komunikácií

5 Testovanie

Súčasťou odovzdaných súborov je testovací skript spolu so sadou jednoduchých testov. Existujú 4 sady testov.

```
test
├── out
│   ├── icmp
│   ├── icmp_fcache
│   ├── icmp_inactive
│   ├── tcpdownload_active
│   └── ...
├── src
│   ├── icmp.pcap
│   ├── tcpdownload.pcap
│   └── ...
├── test.sh
└── cases
```

Jednoduché testy na kontrolu ošetrenia **nesprávne zadaných argumentov** programu sa nachádzajú v súbore `cases`. V tomto prípade test prečíta prvý riadok súboru, ktorý obsahuje spustenie programu a druhý riadok, ktorý obsahuje príslušnú chybovú hlášku. Ďalšou skupinou testov sú testy na základnú funkcionalitu, ktoré fungujú zachytávaním komunikácie pomocou nfcapd, následne je spustený exportér s jedným zo zdrojových súborov v priečinku `test/src` a porovnávaný s výstupom z priečinku `test/out`. Výstupy v priečinku `out` boli generované ručne, keďže sa jednalo o pomerne malé vstupné testovacie súbory. Podobne sú vytvorené testy aj pre **obmedzenie veľkosti** *flow_cache*, nastavenie **aktívneho** a **neaktívneho** časovača alebo **exportovanie TCP tokov na základe príznakov FIN a RST**. Testy je možné jednoducho spustiť pomocou príkazu `make test`. Príklad testovania:

```
student@student-vm:~/ISA/netflow_generator$ make test
g++ -g -Wall -Wextra -MMD -MP -c flow.cpp -o flow.o
g++ -g -Wall -Wextra -MMD -MP -c src/capture.cpp -o src/capture.o
g++ -g -Wall -Wextra -MMD -MP -c src/store.cpp -o src/store.o
g++ -g -Wall -Wextra -MMD -MP -c src/parse.cpp -o src/parse.o
g++ -g -Wall -Wextra flow.o src/capture.o src/store.o src/parse.o -o flow -lpcap
cd test; ./test.sh
Running options tests
./../flow -f abc Passed
./../flow -f test.sh Passed
./../flow -a -20 Passed
./../flow -i -20 Passed
./../flow -m -20 Passed
./../flow -a abc Passed
./../flow -i abc Passed
./../flow -m abc Passed
./../flow -m abc -i -20 -a omg Passed
./../flow -c abcdefgh Passed

Running basic tests
icmp.pcap: Passed
mixed.pcap: Passed
tcpdownload.pcap: Passed
tcpfin.pcap: Passed
tcp.pcap: Passed
udp.pcap: Passed
wifi.pcap: Passed

Running flow cache capacity tests
icmp.pcap: Passed
udp.pcap: Passed

Running inactive timer tests
icmp.pcap: Passed
udp.pcap: Passed

Running active timer tests
tcpdownload.pcap: Passed

Passed 22 out of 22 tests
student@student-vm:~/ISA/netflow_generator$ make clean
rm -f flow  flow.d  src/capture.d  src/store.d  src/parse.d  flow.o  src/capture.o  src/store.o  src/parse.o
rm -f test/out/*.pcap_out test/out/*.pcap_fcache test/out/*.pcap_inactive test/out/*.pcap_active
```

6 Príklady použitia aplikácie

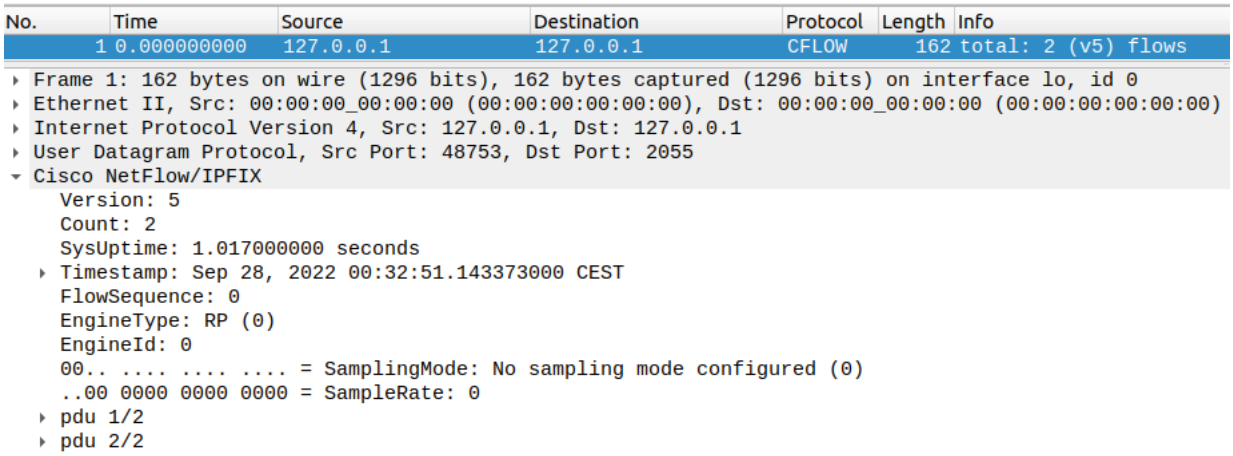
V tejto sekcii je demonštrácia výsledného programu na jednoduchom príklade. Budeme vychádzať zo vstupného súboru `test/src/icmp.pcap`, ktorý obsahuje ICMP správy medzi 2 zariadeniami.

```
student@student-vm:~/ISA/netflow_generator$ tcpdump -n -r test/src/icmp.pcap | awk '{print $1,$3,"->",$5,$6,$8,$13,$14}
reading from file test/src/icmp.pcap, link-type EN10MB (Ethernet)
23:32:50.126919 100.64.208.103 -> 66.254.114.41: ICMP request, length 64
23:32:50.137922 66.254.114.41 -> 100.64.208.103: ICMP reply, length 64
23:32:51.127619 100.64.208.103 -> 66.254.114.41: ICMP request, length 64
23:32:51.143373 66.254.114.41 -> 100.64.208.103: ICMP reply, length 64
```

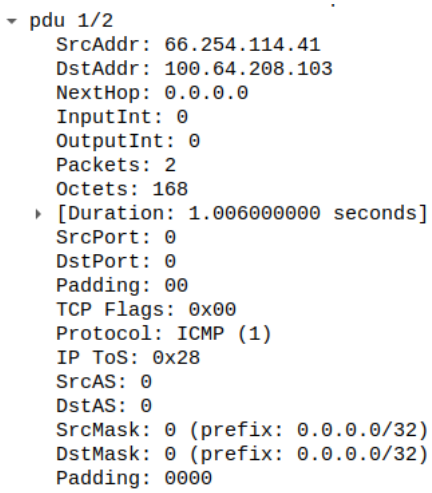
Z výpisu vidíme, že tento súbor sa skladá zo 4 správ typu ICMP. V stĺpcoch vidíme v poradí:

1. čas, kedy bol paket odoslaný
2. zdrojovú IP adresu
3. cieľovú IP adresu
4. typ ICMP správy
5. dĺžku paketu (bez IP hlavičky)

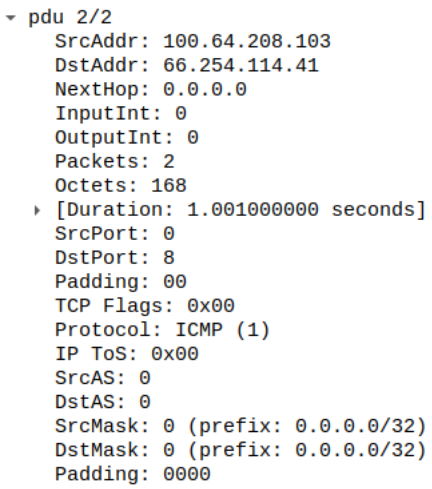
Po spustení príkazu `./flow -c localhost:2055 -f test/src/icmp.pcap` by sme v aplikácií Wireshark mali vidieť odoslanie jedného CFLOW (CiscoFLOW) paketu obsahujúci hlavičku a 2 vygenerované toky.



Obr. 1: Zachytený CFLOW v aplikácií wireshark



(a) Prvý vygenerovaný tok



(b) Druhý vygenerovaný tok

Vygenerované NetFlow pakety je možné zobraziť aj priamo pomocou aplikácie nfdump.

```
student@student-vm:~/ISA/netflow_generator$ nfcapd -n "ISA-NetFlow,127.0.0.1,." -p 1234
Add extension: 2 byte input/output interface index
Add extension: 4 byte input/output interface index
Add extension: 2 byte src/dst AS number
Add extension: 4 byte src/dst AS number
Add extension: 4 byte output bytes
Add extension: 8 byte output bytes
Add extension: NSEL Common block
Add extension: NSEL xlate ports
Add extension: NSEL xlate IPv4 addr
Add extension: NSEL xlate IPv6 addr
Add extension: NSEL ACL ingress/egress acl ID
Add extension: NSEL username
Add extension: NSEL max username
Add extension: NEL Common block
Bound to IPv4 host/IP: any, Port: 1234
Startup.
Init IPFIX: Max number of IPFIX tags: 68
Process_v5: New exporter: SysID: 1, engine id 0, type 0, IP: 127.0.0.1, Sampling Mode: 0, Sampling Interval: 1

^CIdent: 'ISA-NetFlow' Flows: 2, Packets: 4, Bytes: 336, Sequence Errors: 0, Bad Packets: 0
Total ignored packets: 0
Terminating nfcapd.
student@student-vm:~/ISA/netflow_generator$ nfdump -r nfcapd.202210301246 -o 'fmt:%sa:%sp %da:%dp %pr %pkt %byt'
     Src IP Addr Src Pt      Dst IP Addr Dst Pt Proto  Packets    Bytes
   66.254.114.41:     0   100.64.208.103:   0.0 ICMP         2      168
  100.64.208.103:     0    66.254.114.41:   0.8 ICMP         2      168
Summary: total flows: 2, total bytes: 336, total packets: 4, avg bps: 2643, avg pps: 3, avg bpp: 84
Time window: 2022-09-28 00:32:50 - 2022-09-28 00:32:51
Total flows processed: 2, Blocks skipped: 0, Bytes read: 232
Sys: 0.001s flows/second: 1197.6     Wall: 0.000s flows/second: 22988.5
```

Obr. 2: Zachytený NetFlow v aplikácií nfdump

Literatúra

[1] Claise, B.; aj.: Cisco Systems NetFlow Services Export Version 9. Október 2004, doi:10.17487/RFC3954.
URL https://www.rfc-editor.org/rfc/rfc3954

[2] Kawamura, S.; Kawashima, M.: A Recommendation for IPv6 Address Text Representation. August 2010, doi: 10.17487/RFC5952.
URL https://www.rfc-editor.org/rfc/rfc5952#page-11

[3] Parker, J.: What is Netflow? 2021, [Online; navštívené 23-Október-2022].
URL https://www.pcwdld.com/what-is-netflow

[4] Wikipedia contributors: NetFlow – Wikipedia, The Free Encyclopedia. 2022, [Online; navštívené 29-Október-2022].
URL https://en.wikipedia.org/w/index.php?title=NetFlow&oldid=1115766584