Vysoké učení technické v Brně
Fakulta informačních technologií

Síťové aplikace a správa sítí

PCAP NetFlow v5 exportér

2024 Michal Balogh (xbalog06)

Obsah

1 Uvedenie do problematiky

1.1 Zadanie

Za úlohu som mal vytvoriť program `p2nprobe`, ktorý bude extrahovať informácie o tokoch z PCAP súboru. Tieto toky bude odosielať na kolektor vo formáte NetFlow v5.

Program prijíma rôzne argumenty, pomocou ktorých sa dá nastaviť, ako budú toky agregované a kam ich bude odosielať (viz sekcia 4).

Program bude na vstupe čítať pakety z PCAP súboru zadaného ako argument programu a tie spracuje a agreguje do tokov. Toky potom pomocou protokolu UDP odošle na NetFlow v5 kolektor, kde sú tieto toky ďalej spracované a analyzované. Program `p2nprobe` je len exporter a je zameraný len na záznamy o tokoch TCP.

1.2 Základné informácie

NetFlow v5 protokol vyvinutý spoločnosťou Cisco a slúži na monitorovanie a analýzu sieťového toku dát. Slúži na zbieranie informácií o sieťovej prevádzke a následnú analýzu na účely monitorovania, optimalizácie a bezpečnosti siete.

Toky sú agregované podľa určitých kritérií, ako sú napríklad IP adresa odosielateľa a prijímateľa, porty, protokol a ďalšie. Podľa nich sú prichádzajúce pakety jedinečne identifikovateľné a priradené do tokov.

Tok môže byť ukončený rôznymi spôsobmi. V programe `p2nprobe` je tok ukončený dvoma spôsobmi:

- po uplynutí aktívneho timeoutu,
- po uplynutí neaktívneho timeoutu.

Aktívny timeout je časový interval, po ktorom je tok ukončený, aj keď do neho stále prichádzajú pakety.

Neaktívny timeout je časový interval, po ktorom je tok ukončený, ak do neho neprichádzajú žiadne pakety po nejaký čas.

Exportér expirované toky uchováva v pamäti a po nazbieraní maximálneho počtu tokov v pamäti alebo prečítaní posledného paketu zo súboru ich odošle na kolektor. Odoslaný UDP datagram obsahuje hlavičku a záznamy, ktorých podľa špecifikácie od spoločnosti Cisco je v rozsahu 1–30.

2 Návrh aplikácie

Program je rozdelený do viacerých logických častí, ktoré vykonávajú určité úlohy:

- Načítanie argumentov programu a ich spracovanie – `ArgParser`,
- Načítanie paketov zo súboru a extrahovanie podstatných informácií – `PcapReader`,
- Správa a agregácia tokov – `FlowManager`,
- Odosielanie tokov na kolektor – `Exporter`.

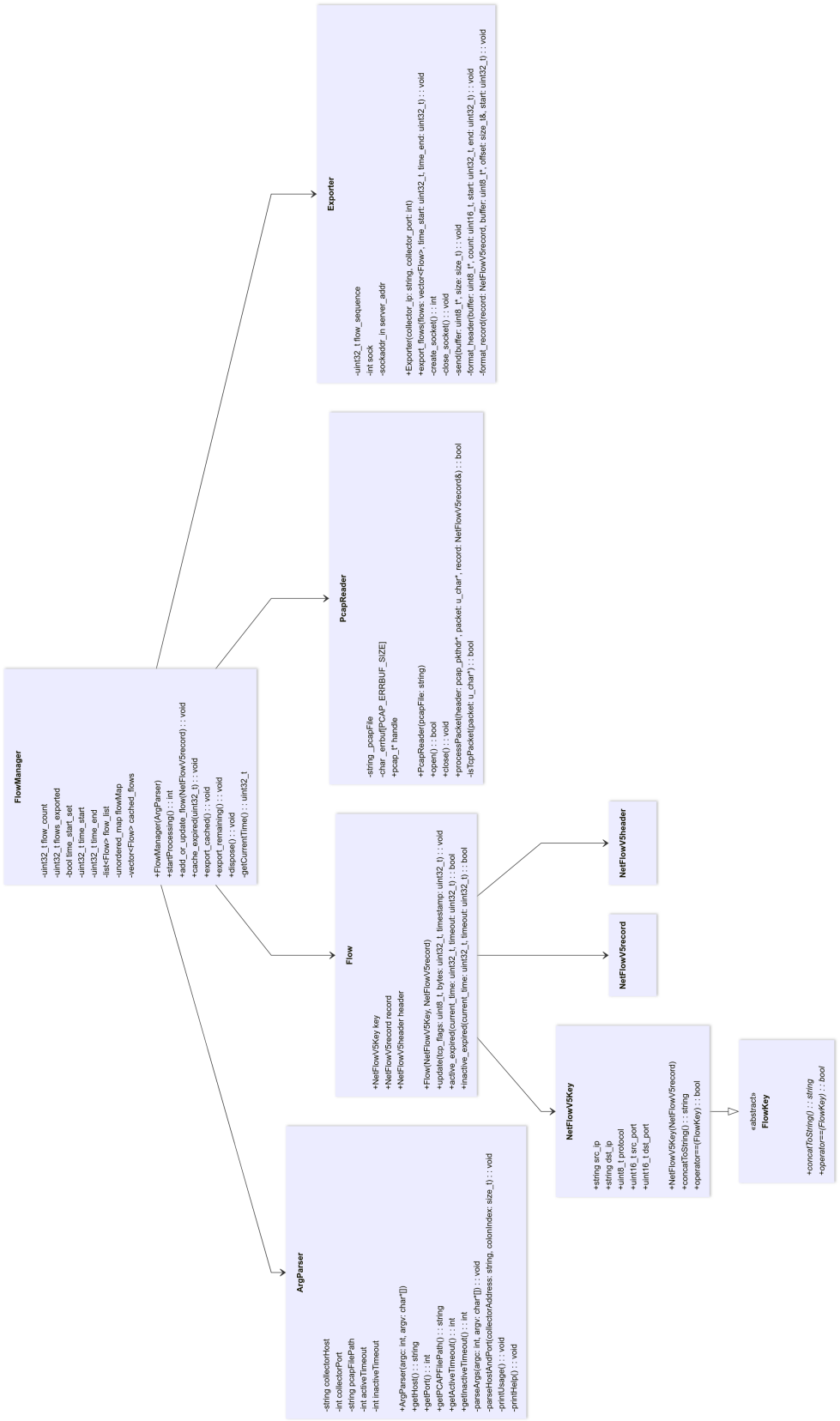
Program najprv spracuje vstupné argumenty a skontroluje ich validitu pomocou modulu `ArgParser`. Následne modul `PcapReader` načíta pakety zo súboru zadaného pomocou argumentu. Každý paket je spracovaný a sú z neho extrahované informácie, ktoré sú potrebné na identifikáciu toku a informácie potrebné pre štatistiky. Tieto informácie sú poslané do modulu `FlowManager`, ktorý z nich vytvorí jedinečný kľúč, pomocou ktorého identifikuje tok. Ak nejaký tok s týmto kľúčom existuje, aktualizuje

ho, inak vytvorí nový. Ak `FlowManager` vyhodnotí tok ako expirovaný pomocou informácií z paketu alebo informácií od modulu `PcapReader`, že prečítal posledný paket z PCAP súboru, je odoslaný na kolektor pomocou modulu `Exporter`.

3 Popis implementácie

Nižšie je zobrazený class diagram, ktorý zobrazuje, ako sú jednotlivé moduly programu `p2nprobe` implementované a ako spolu komunikujú.

3.1 Diagram



Obr. 1: Class diagram

3.2 Implementácia

Program `p2nprobe` je implementovaný v jazyku C++ verzia C++17 a využíva knižnicu `libpcap` na čítanie a spracovanie paketov.

3.2.1 Použité knižnice

- **libpcap** – čítanie PCAP súborov a extrahovanie informácií z paketov,
- **arpa/inet.h** – konverzia IP adries a portov medzi sieťovým a hostiteľským formátom,
- **netinet/ip.h**, **netinet/tcp.h** – parsovanie sieťových hlavičiek,
- **sys/socket.h** – odosielanie NetFlow záznamov na kolektor,
- **unordered_map**, **vector**, **string** – ukladanie a správa tokov a záznamov,
- **ctime**, **chrono** – práca s časovými značkami a meranie času,
- **iostream**, **sstream** – formátovanie výstupu (správa help, rôzne chybové hlášky),
- **memory** – smart pointre a správa pamäte.

3.3 Konkrétne moduly

3.3.1 ArgParser

Modul na načítanie vstupných argumentov programu, spracovanie a overenie, či sú argumenty validné. V prípade chyby alebo chýbajúcich povinných argumentov vypíše chybovú hlášku a ukážku, ako spustiť program. Argument -h vypíše "help" správu, ktorá informuje užívateľa o použití programu a jednotlivých argumentoch.

3.3.2 ErrorCodes

Definície rôznych chybových hlášok pre lepšie rozpoznávanie, aká chyba nastala. Možné chybové kódy:

- 0 - Bez chyby
- 1 - Vnútorná chyba, napríklad chyba alokácie pamäti
- 2 - Nesprávne argumenty
- 3 - Chyba pri otváraní súboru
- 4 - Chyba pri čítaní paketu
- 5 - Paket obsahuje nevalidné informácie

Implementuje aj funkciu ExitWith pre konzistentné ukončenie programu, na zjednodušenie správy chybových stavov.

3.3.3 PcapReader

Trieda na čítanie paketov zo súboru a ich spracovanie. Poskytuje rozhranie pre extrakciu TCP paketov a základných informácií z paketov a tie uloží do štruktúry NetFlowV5record. Konštruktor triedy PcapReader má jeden parameter - cestu k PCAP súboru, ktorý sa má čítať. Metóda `processPacket` načíta jeden paket a spracuje ho. Metóda `isTcpPacket` kontroluje, či je daný paket TCP paket - ostatné pakety sú ignorované.

3.3.4 FlowManager

Hlavná trieda zodpovedná za správu a agregáciu tokov. Rieši komunikáciu medzi jednotlivými triedami. Vytvára toky a kľúče pre ne podľa informácií z paketu. Pomocou týchto kľúčov potom vie identifikovať, či tok už existuje alebo nie. Ak tok existuje, pridá paket do toku pomocou metódy `add_or_update_flow`. Ak tok neexistuje, vytvorí nový tok a pridá paket do neho. Na efektívne vyhľadávanie využíva kombinovanú dátovú štruktúru (hash mapa + linked list) pre efektívne vyhľadávanie a správu tokov. Hash mapa slúži na rýchle vyhľadanie tokov pomocou kľúča a linked list obsahuje odkazy do hashmapy, ale zároveň udržuje poradie, v akom sa toky vytvorili. Toky, ktoré expirovali neexportuje hneď, ale "cachuje" pomocou metódy `cache_expired` a exportuje ich až keď je naplnený maximálny počet tokov v pamäti (30) alebo je prečítaný posledný paket zo súboru. Má dve metódy na exportovanie tokov na kolektor - `export_cached` a `export_remaining`. Prvá metóda exportuje všetky toky, ktoré sú uložené v cache keď sa naplní kapacita, druhá metóda exportuje všetky toky, keď sa načíta posledný paket ale zároveň cache ešte nie je plná.

3.3.5 NetFlowV5Key

Implementácia unikátneho kľúča pre identifikáciu tokov. Kombinuje 5 kľúčových atribútov: zdrojová/cieľová IP, porty a protokol pre jednoznačnú identifikáciu toku. Využíva preťaženie operátora `==` pre porovnávanie kľúčov a implementuje vytvorenie kľúča (jednoduchá konkatenácia všetkých atribútov kľúča) z údajov, ktoré sú mu dané pre použitie v hash mape.

3.3.6 Exporter

Modul pre formátovanie a export NetFlow záznamov na kolektor. Modul formátuje NetFlow záznamy (štruktúra NetFlowV5record) a NetFlow hlavičky (štruktúra NetFlowV5header) podľa špecifikácie NetFlow v5 a odosiela ich na kolektor špecifikovaný v programových argumentoch pomocou protokolu UDP. Okrem toho aj počíta počet odoslaných paketov, keďže tento údaj je potrebný v hlavičke NetFlow záznamu. Podporuje resolvovanie hostname na IP adresu pomocou funkcie `getaddrinfo` (knižnica `arpa/inet.h`).

3.3.7 Flow

Reprezentácia jednotlivého sieťového toku, ktorá zapuzdruje všetky potrebné informácie o toku a jeho štatistiky. Taktiež obsahuje metódy na pridanie paketu do toku a kontrolu, či je tok expirovaný buď pomocou aktívneho alebo neaktívneho timeoutu. Jeho konštruktor má 2 parametre - kľúč, podľa ktorého je daný flow identifikovateľný a prvý záznam z paketu, ktorý je pridaný do toku.

3.3.8 Štruktúra NetFlowV5header

Štruktúra, ktorá reprezentuje hlavičku NetFlow záznamu. Obsahuje informácie uvedené v tabuľke nižšie alebo v dokumentácii od spoločnosti Cisco[4][3].

Offset	Field	Popis
0-1	version	Číslo verzie formátu exportu NetFlow
2-3	count	Počet tokov exportovaných v tomto pakete (1-30)
4-7	SysUptime	Aktuálny čas v milisekundách od spustenia exporteru
8-11	unix_secs	Aktuálny počet sekúnd od 0000 UTC 1970
12-15	unix_nsecs	Zvyškové nanosekundy od 0000 UTC 1970
16-19	flow_sequence	Počítadlo sekvencie celkového počtu tokov
20	engine_type	Typ engine-u na prepínanie tokov
21	engine_id	Číslo slotu engine-u na prepínanie tokov
22-23	sampling_interval	Prvé dva bity obsahujú režim vzorkovania; zvyšných 14 bitov obsahuje hodnot

3.3.9 Štruktúra NetFlowV5record

Štruktúra, ktorá reprezentuje záznam NetFlow v5. Obsahuje informácie uvedené v tabuľke nižšie alebo v dokumentácii od spoločnosti Cisco[4][3].

Bajty	Obsah	Popis
0-3	srcaddr	Zdrojová IP adresa
4-7	dstaddr	Cieľová IP adresa
8-11	nexthop	IP adresa ďalšieho smerovača
12-13	input	SNMP index vstupného rozhrania
14-15	output	SNMP index výstupného rozhrania
16-19	dPkts	Počet paketov v toku
20-23	dOctets	Celkový počet bajtov vrstvy 3
24-27	First	SysUptime na začiatku toku
28-31	Last	SysUptime v čase prijatia posledného paketu toku
32-33	srcport	Zdrojový port TCP/UDP alebo ekvivalent
34-35	dstport	Cieľový port TCP/UDP alebo ekvivalent
36	pad1	Nepoužité (nulové) bajty
37	tcp_flags	Kumulatívny OR TCP príznakov
38	prot	Typ IP protokolu (napríklad TCP = 6; UDP = 17)
39	tos	IP typ služby (ToS)
40-41	src_as	Číslo autonómneho systému zdroja
42-43	dst_as	Číslo autonómneho systému cieľa
44	src_mask	Bitová maska zdrojovej adresy
45	dst_mask	Bitová maska cieľovej adresy
46-47	pad2	Nepoužité (nulové) bajty

4 Návod na použitie

4.1 Kompilácia programu

Program je možné skompilovať pomocou `Makefile`. Pre kompiláciu programu je potrebné mať nainštalované knižnice `libpcap` a `g++` (verzia C++17). Preloženie programu je možné pomocou príkazu:

```
make
```

4.2 Spustenie programu

Program je možné spustiť takto:

```
./p2nprobe <host>:<port> <pcap_file_path> [-a <active_timeout> -i <inactive_time
```

Kde:

- `<pcap_file_path>` – cesta k PCAP súboru, ktorý sa má čítať,
- `<host>` – IP adresa kolektora, kam sa majú odosielať toky,
- `<port>` – port kolektora, kam sa majú odosielať toky,
- `-a <active_timeout>` – aktívny timeout, po ktorom je tok ukončený (defaultná hodnota 60 sekúnd),
- `-i <inactive_timeout>` – neaktívny timeout, po ktorom je tok ukončený (defaultná hodnota 60 sekúnd).

Poradie parametrov je ľubovoľné. Ukážka spustenia programu:

```
./p2nprobe localhost:9995 pcap_file.pcap -a 5 -i 30

./p2nprobe 127.0.0.1:9995 pcap_file.pcap
```

Keďže `p2nprobe` je len exporter, je potrebné mať spustený NetFlow kolektor, ktorý bude prijímať toky. Napríklad pomocou programu `nfcapd`:

```
nfcapd -l . -p 9995
```

Následne je možné sledovať toky pomocou programu `nfdump`:

```
nfdump -r nfcapd.202411052020
```

5 Popis testovania aplikácie

Všetky skripty použité na testovanie su v priečinku `tests`.

5.1 Testovanie CLI argumentov

Na testovanie command line argumentov som použil vlastný skript napisany v pythone: `test_args.py`. Skript testuje rozne argumenty a vždy testuje všetky možne permutácie týchto argumentov, aby sa zaistilo, že argumenty môžu byť v ľubovoľnom poradí.

5.2 Aplikačné testy

Aplikáciu som testoval pomocou porovnávania s odporúčaným programom ako referenciu – `softflowd`[12]. NetFlow toky som zachytával pomocou programu `nfcapd`.

```
nfcapd -l . -p 9995
```

V druhom termináli som potom spustil:

```
sudo softflowd -r netflow_capture1.pcap -n localhost:9995
```

Bohužiaľ, aktívny a neaktívny timeout v `softflowd` nefungoval, takže som mohol testovať len predvolené hodnoty.

Následne som toky analyzoval a porovnával pomocou programu `nfdump`.

```
nfdump -r nfcapd.202411052100
```

Takto som získal referenčný výstup. Tento postup som opakoval pre môj program `p2nprobe`. Na porovnanie som si vytvoril Python skript, ktorý porovnáva výstupy od `nfdump` pre oba programy.

Keďže nebolo možné takto testovať aktívny a neaktívny timeout, tak som potom zachytával pomocou vlastného Python skriptu `test_probe.py`, ktorý vie vypísať všetky údaje z NetFlow záznamov na porovnanie alebo ladenie, ale aj porovnať s iným kolektorom, napríklad `softflowd`. Skript `test_probe.py` sa dá spustiť napríklad takto:

```
python3 tests/test_probe.py tcp.pcap 9995 60 60
```

kde:

- `tcp.pcap` – je pcap súbor na testovanie,
- `9995` – je port kolektora,
- `60` – aktívny timeout (nepovinný argument),
- `60` – neaktívny timeout (nepovinný argument).

6 Výsledky testov

6.1 Výsledky testovania CLI argumentov

Testy na arugmenty programu možno spustiť z koreňového adresára projektu takto:

```
python3 tests/test_args.py
```

Výsledky testov pre argumenty programu `test_args.py`:

```
Running p2nprobe argument tests with permutations...

[PASS] Valid arguments
  Total permutations tested: 2
[PASS] Help message
  Total permutations tested: 1
[PASS] Help message 2
  Total permutations tested: 2
[PASS] Help message 3
  Total permutations tested: 6
[PASS] Help message 3
  Total permutations tested: 2
[PASS] Missing arguments
  Total permutations tested: 1
[PASS] Invalid port
  Total permutations tested: 2
[PASS] Invalid active timeout
  Total permutations tested: 24
[PASS] Invalid inactive timeout
  Total permutations tested: 24
[PASS] Missing port
  Total permutations tested: 2
[PASS] Invalid port format
  Total permutations tested: 2
[PASS] Valid timeouts
  Total permutations tested: 24
[PASS] Extra arguments
  Total permutations tested: 6

Summary: 13/13 tests passed
Total permutations tested: 98
```

Vo výsledkoch je vždy či test prešiel alebo nie, názov toho, čo sa testovalo, a počet permutáci argumentov prítomných v danom teste. Toto robustné testovanie by malo zabezpečiť funkčnosť programu s ľubovoľným poradím argumentov.

6.2 Výsledky aplikačných testov

Pri porovnávaní výstupu mojho programu a programu softlow mohli výsledky z nfacpd vyzerať napríklad takto:

```
Ident: 'none' Flows: 60, Packets: 1752, Bytes: 1063737, Sequence Errors: 0,
Bad Packets: 0
Total ignored packets: 0
Terminating nfcapd.

Ident: 'none' Flows: 60, Packets: 1752, Bytes: 1063737, Sequence Errors: 0,
Bad Packets: 0
Total ignored packets: 0
Terminating nfcapd.
```

Po zobrazeni s programom `nfdumb` to vyzera takto (len poslednych par riadkov vypisu):

```
2024-10-09 20:27:46.445 INVALID  Ignore TCP      192.168.50.54:46998 ->     77.75.76.209:443           0.0.0.0:0     ->
0.0.0.0:0          3048        0
2024-10-09 20:27:46.500 INVALID  Ignore TCP       77.75.78.104:443   ->    192.168.50.54:47944         0.0.0.0:0     ->
0.0.0.0:0         22482        0
2024-10-09 20:27:46.500 INVALID  Ignore TCP      192.168.50.54:47944 ->     77.75.78.104:443           0.0.0.0:0     ->
0.0.0.0:0          1932        0
Summary: total flows: 60, total bytes: 1.1 M, total packets: 1752, avg bps: 2.4 M, avg pps: 503, avg bpp: 607
Time window: 2024-10-09 20:27:43 - 2024-10-09 20:27:46
Total flows processed: 60, Blocks skipped: 0, Bytes read: 4928
Sys: 0.002s flows/second: 25895.6     Wall: 0.000s flows/second: 66006.6
```

INVALID je tam kvôli tomu, že načítavame súbor, ktorý je "offline". Keďže boli niektoré toky v inom poradí, vytvoril som si už vyššie spomínaný python skript, ktorý mi výstupy porovnal a ignoroval pri tom poradie tokov.

Pre detailnejšie testovanie som potom tento skript prerobil na `test_probe.py`, ktorý porovnával výstup s referenčným a kde som vedel preskúmať napríklad aj dĺžku toku. Výstup tohto testovacieho skriptu vyzeral napríklad takto (zase len posledných pár riadkov výstupu)

```
Flow ('192.168.50.54', 46998, '77.75.76.209', 443) matched!
Time difference: 0ms

Discrepancy in flow ('77.75.79.195', 443, '192.168.50.54', 33630):
Collected: 12 packets | Reference: 12 packets
Collected: 6385 octets | Reference: 6385 octets
Time difference: 12ms

Flow ('192.168.50.54', 33630, '77.75.79.195', 443) matched!
Time difference: 0ms

Discrepancy in flow ('77.75.78.104', 443, '192.168.50.54', 47944):
Collected: 24 packets | Reference: 24 packets
Collected: 22482 octets | Reference: 22482 octets
Time difference: 49ms

Flow ('192.168.50.54', 47944, '77.75.78.104', 443) matched!
Time difference: 0ms

Matched flows: 60/60
Total packets: 1752 (Reference: 1752)
Total octets: 1063737 (Reference: 1063737)
```

7 Bibliografia

Literatúra

[1] CPlusPlus.com, *C++ Reference*, [Online]. Dostupné na: https://cplusplus.com/reference/ [Citované: 17. novembra 2024]

[2] W3Schools, *C++ OOP*, [Online]. Dostupné na: https://www.w3schools.com/cpp/cpp_oop.asp [Citované: 17. novembra 2024]

[3] IBM Documentation, *NetFlow V5 Formats*, [Online]. Dostupné na: https://www.ibm.com/docs/en/npi/1.3.0?topic=versions-netflow-v5-formats [Citované: 17. novembra 2024]

[4] Cisco Systems, *NetFlow Collection Engine User Guide*, [Online]. Dostupné na: https://www.cisco.com/c/en/us/td/docs/net_mgmt/netflow_collection_engine/3-6/user/guide/format.html [Citované: 17. novembra 2024]

[5] Tcpdump/Libpcap, *PCAP Documentation*, [Online]. Dostupné na: https://www.tcpdump.org/pcap.html [Citované: 17. novembra 2024]

[6] Voldemur, *PCAP Reader Implementation*, GitHub Gist, [Online]. Dostupné na: https://gist.github.com/voldemur/261b5bc42688b9cf425fbaedc2d5fcbe#file-pcap_reader-cpp [Citované: 17. novembra 2024]

[7] Linux manual pages, *inet_ntop(3) - Linux manual page*, [Online]. Dostupné na: https://man7.org/linux/man-pages/man3/inet_ntop.3.html [Citované: 17. novembra 2024]

[8] Linux manual pages, *getaddrinfo(3) - Linux manual page*, [Online]. Dostupné na: https://man7.org/linux/man-pages/man3/getaddrinfo.3.html [Citované: 17. novembra 2024]

[9] Scaler Topics, *Virtual Base Class in C++*, [Online]. Dostupné na: https://www.scaler.com/topics/virtual-base-class-in-cpp/ [Citované: 17. novembra 2024]

[10] GeeksforGeeks, *Copy Constructor in C++*, [Online]. Dostupné na: https://www.geeksforgeeks.org/copy-constructor-in-cpp/ [Citované: 17. novembra 2024]

[11] Educative.io, *How to implement UDP sockets in C*, [Online]. Dostupné na: https://www.educative.io/answers/how-to-implement-udp-sockets-in-c [Citované: 17. novembra 2024]

[12] Irino, *softflowd*, GitHub repository, [Online]. Dostupné na: https://github.com/irino/softflowd/ [Citované: 17. novembra 2024]