

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ISA
Generování NetFlow dat
ze zachycené síťové komunikace
MANUAL

Obsah

1	Návod na použití	3
1.1	Obsah odevzdaného archivu	3
1.2	Rozbalení a použití	3
2	Teorie	4
2.1	Neflow	4
2.2	Netflow verze 5	4
2.3	Odesílání toků	4
2.4	Analýza paketů	5
2.4.1	Ethernetový rámec	5
2.4.2	Hlavička IPv4 paketu	5
2.4.3	TCP, UDP a ICMP	5
3	Implementační detaily	6
3.1	Parsování argumentů	6
3.2	Zpracování kolektoru	6
3.3	Zpracování paketů	7
3.4	Vytváření a zpracování toků	7
3.5	Odesílání flows	8
4	Testování	8

Zadání

Vytvořte aplikaci, která bude ze zachycených síťových dat ve formátu pcap vytvářet záznamy NetFlow, které následně odešle na kolektor, který je buď zadán příkazovou řádkou nebo se pošle na výchozí kolektor.

Spuštění aplikace:

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

- `-f <file>` jméno analyzovaného souboru nebo STDIN,
- `-c <netflow_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 <inactive_timer>` - 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 cache na kolektor (1024, pokud není specifikováno).

Všechny parametry jsou brány jako volitelné. Pokud některý z parametrů není uveden, použije se místo něj výchozí hodnota.

Upřesnění zadání: Program zpracuje vstupní argumenty, zpracuje vstupní soubor pokud byl zadán jinak čte ze standardního vstupu a poté jednotlivé „flows“ odešle na specifikovaný 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,
- Exportované NetFlow data by měla být čitelná nástrojem nfdump.

1 Návod na použití

1.1 Obsah odevzdaného archivu

- `Makefile`: s cíly `all` (sestaví výsledný program `flow.cpp`), `clean` (vymaže všechny vzniklé binární soubory a spustitelný program), `archive` (vytvoří archive celého projektu),
- `manual.pdf`: soubor s programovou dokumentací,
- `flow.cpp`: zdrojový soubor výsledného programu,
- `flow.1`: manuálová stránka programu.

1.2 Rozbalení a použití

První je nutné si tar-archiv rozbalit pomocí příkazu `tar -xvf xkorva03.tar`. Poté je nutné program přeložit pomocí příkazu `make` nebo pomocí příkazu `make all`. Tímto vznikne spustitelný soubor v kořenové složce adresáře, ve kterém je spuštěn příkaz `make`. Po přeložení aplikace ji lze jednoduše spustit pomocí `./flow <argumenty>`, kde je možné uvést jednotlivé argumenty v libovolném pořadí.

Ovšem pokud není zadán parametr `-f`, který určuje vstupní pcap soubor, ze kterého se bude číst, tak program bude číst ze standartního vstup neboli z `STDIN`. U parametru `-c`, kterým se určuje IPv4 nebo IPv6 adresu kolektoru. Ovšem je nutné při zadávání IPv6 adresy spolu s portem tuto adresu zadat následujícím způsobem: `[IPv6_adresa]:port`. U adres IPv4 nebo doménového jména žádná taková omezení nejsou. Při zadávání IPv6 adresy bez portu žádné omezení není a lze ji zadat normálně a není nutné ji dát do hranatých závorek.

Při nevalidním zadání vstupních argumentů končí program s návratovým chybovým kódem 1 a odpovídající chybovou hláškou.

Příklady spuštění

Zde je uvedeno několik validních spuštění programu.

- `./flow < icmp.pcap`
- `./flow -f icmp.pcap -c [::1]:2055`
- `./flow -f tcp.pcap -c ::1`
- `./flow -c 127.0.0.1:2055 < icmp.pcap`
- `./flow -f icmp.pcap -c localhost -m 5`

2 Teorie

2.1 Neflow

NetFlow [8] je funkce, která umožňuje shromažďovat síťový provoz IP při vstupu nebo výstupu z rozhraní. Analýzou dat poskytovaných NetFlow může správce sítě určit například zdroj a cíl provozu a příčiny přetížení. Typické nastavení monitorování toku se skládá ze tří hlavních součástí: Flow exportér, Flow kolektor a aplikace pro analýzu. Nás dále bude zajímat pouze flow exportér, protože to je hlavním cílem tohoto projektu.

2.2 Netflow verze 5

Cisco standart NetFlow verze 5 definuje tok (neboli flow) jako jednosměrnou sekvenci paketů. A tyto pakety mají sedm společných hodnot a tyto hodnoty definují jednoznačný klíč pro tento tok. Tok je tedy identifikován následující sedmicí: vstupní rozhraní, zdrojová IP adresa, cílová IP adresa, protokol IP, zdrojový port¹, cílový port², typ služby IP. V našem případě vstupní rozhraní neznáme, jelikož provádíme pouze analýzu z pcap souboru, tedy tuto informaci vynecháváme a tok je definován zbylou šesticí. Na obrázku 1 lze vidět co vše obsahuje hlavička jednotlivých toků a co se nachází v těle lze najít zde [2].

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

Obrázek 1: Hlavička NetFlow v5

2.3 Odesílání toků

V našem případě dochází k exportu jednotlivých toků v následujících čtyřech případech. Zprvu dochází k periodickému exportování toků vždy, když vyprší aktivní časovač. Poté pokud je některý z toků neaktivní delší dobu než je specifikováno neaktivním časovačem. Zatřetí dojde k exportu nejstaršího záznamu v paměti pokud je paměť plná a posledním případem je ukončení relace TCP toku.

¹Zdrojový port je pouze pro protokoly TCP a UDP, u ostatních protokolů je to 0.

²Port se uvádí pro protokoly UDP nebo TCP, pro protokol ICMP je to typ a kód, pro ostatní protokoly je to 0.

2.4 Analýza paketů

2.4.1 Ethernetový rámec

V ethernetovém rámci má hlavička stále stejnou fixní velikost a to 14 bytů. Z těchto bytů je pro nás důležitá informace, která se nachází na posledních dvou bytech a to je takzvaný Ethertype, neboli určení protokolu vyšší vrstvy. Tady nás bude zajímat pouze, zda-li se na vyšší vrstvě nachází IPv4 hlavička.

2.4.2 Hlavička IPv4 paketu

Hlavička IPv4 datagramu [3] má variabilní délku, proto je nutné její délku zjistit z pole `header length`, jenže toto číslo je nutné ještě vynásobit číslem 4, abychom dostali velikost hlavičky. Další informace, které nás zajímají je délka datagramu, typ protokolu, který se nachází na vyšší vrstvě, zdrojová a cílová IP adresa a nakonec ještě ToS. Délka datagramu nás zajímá, protože je potřeba vědět velikost paketu bez ethernetového rámce a kontrolního součtu, který se někdy přikládá nakonec.

2.4.3 TCP, UDP a ICMP

TCP [4], UDP [5] jsou protokoly operující na transportní vrstvě a nachazejí se nad IP datagramem. Z TCP a UDP datagramů budou důležité informace zdrojový a cílový port. Budou totiž sloužit k jednoznačné identifikaci jednotlivých záznamů toku.

Z ICMP [3] hlavičky, která se nachází za hlavičkou IP, budeme potřebovat informace takzvaný ICMP kód a ICMP typ. Kde ICMP kód určuje o jaký typ se jedná (například echo nebo request) a kód nám, ještě přesněji specifikuje typ ICMP zprávy. Z těchto informací se poté vypočítá číslo, které se uloží jako cílový port a bude tak identifikovat tento tok. Vzorec pro výpočet je následující: `icmp_type * 256 + icmp_code`.

3 Implementační detaily

V následující sekci se popisuje strukturu programu a popisuje řešení některých nejdůležitějších částí tohoto projektu. V programu je hojně využita knihovna `pcap` a několik funkcí z této knihovny, jako třeba funkce na čtení `pcap` souboru nebo funkce na vytvoření filtru, aby byly čteny pouze `ICMP`, `UDP` nebo `TCP` pakety. Všechny potřebné informace, jak použít tyto knihovnní funkce byly z této stránky [1].

Využité knihovny

- `pcap/pcap.h` - knihovna na zachytávání, procházení a filtrování paketů [11],
- `getopt.h` - zpracování argumentů příkazové řádky [6],
- `netinet/ether.h` - zpracování ethernetové hlavičky,
- `sys/socket.h` - knihovna na otevírání síťových schránek,
- `netinet/ip_icmp.h` - zpracování `ICMP` hlaviček paketů,
- `netinet/tcp.h` - zpracování `TCP` hlaviček paketů,
- `netinet/udp.h` - zpracování `UDP` hlaviček,
- `netdb.h` - překlad doménového jména na `IP` adresu,
- `netinet/ip.h` - zpracování `IPv4` hlavičky.

3.1 Parsování argumentů

První věc, která se po spuštění programu provede, je zpracování argumentů z příkazové řádky. Toto se děje ve funkci `parse_arguments`. V této funkci se k tomuto účelu používá funkce `getopt` z knihovny `getopt.h`. Pokud je použit neplatný argument program okamžitě končí s návratovým kódem 1.

Postupně se pak kontrolují všechny argumenty a jejich parametry, například jestli soubor existuje nebo jestli bylo zadáno validní nezáporné číslo pro časovače. Během průchodu všemi argumenty se postupně plní speciální struktura `arguments_t`, ve které jsou uloženy všechny argumenty zadány uživatelem.

3.2 Zpracování kolektoru

Poté co jsou zpracovány všechny programové argumenty tak se volá funkce `parse_collector`. V této funkci probíhá zpracování zadaného kolektoru uživatelem. K tomuto účelu se používá funkce `inet_pton` z knihovny `arpa/inet.h`. Nejprve se otestuje jestli je to validní `IPv4` nebo `IPv6` adresa bez portu, pokud ano tak funkce končí úspěchem a do struktury `arguments_t` si ukládá informaci, jestli se jednalo o `IPv4` nebo `IPv6` adresu.

Pokud ovšem ani jedna z těchto funkcí neúspěje tak se najde poslední : a cokoliv za ní se odstraní a je bráno jako číslo portu. A opět se otestuje, jestli je tato `IP` adresa validní. Pokud ani toto neprojde

tak zbývá poslední možnost a tou je, že uživatel zadal adresu kolektoru pomocí doménového jména. Proto se doménové jméno zkusí přeložit pomocí funkce `getaddrinfo`. Poté se zjistí, jakou adresu nám tato funkce vrátila, tedy jestli se jedná o IPv4 nebo IPv6 adresu. Pokud ani zde nedojde k úspěchu tak funkce vrátí hodnotu 1 a běh programu se ukončí s korespondující chybovou hláškou.

3.3 Zpracování paketů

Po zpracování kolektoru, přichází na řadu čtení paketů z pcap souboru. K tomuto účelu slouží funkce `pcap_open_offline`, která buď čte ze specifikovaného souboru, nebo pokud místo souboru zadáme – tak čte ze standartního vstupu. Dále následuje kontrola, že čteme ze souboru, který podporuje ethernetové hlavičky, pokud ne tak je program ukončen s chybovým kódem 1.

Poté se vytvoří filtr, aby se vyfiltrovali nežádoucí protokoli, a my dostávali pouze pakety s protokoly UDP, TCP a ICMP a pouze IPv4. Poté tento filtr přeložíme pomocí funkce `pcap_compile` a aplikujeme na vstupní soubor. A nakonec se pak prochází přes všechny pakety funkcí `pcap_loop` a pakety se čtou až dokud žádný není. A na každý paket se aplikuje funkce `process_packet`.

V této funkci se nejprve inicializuje struktura `nf_v5_body_t`, která slouží k uchování informací ohledně jednotlivých toků. Poté dochází k extrakci důležitých informací z jednotlivých vrstev a jejich ukládání do této struktury. Je to například zdrojová a cílová adresa, poté počet bytů, cílový a zdrojový port a protokolů UDP a TCP. U prvního paketu si uložíme jeho čas příchodu a považujeme jej za systémový čas, který už nikdy neměníme a je uložen v proměnné `first_packet_time`. A každý následující čas paketu je počítán relativně k tomuto času.

3.4 Vytváření a zpracování toků

Po extrakci všech důležitých informací z hlaviček se volá funkce `process_flow`. Zde se děje hned několik důležitých věcí a to: kontrola, jestli tok již existuje ve flow-cache, pokud ne tak se vytvoří nový záznam toku a uloží se do cache. Poté zde probíhá kontrola jednotlivých podmínek, při jejichž porušení dochází k exportování jednotlivých toků. Mezi tyto podmínky patří kontrola aktivního a neaktivního časovače, kontrola, jestli je plná cache nebo kontrola, zda-li jsme neobdrželi FIN nebo RST příznak v TCP hlavičce.

Kontrola existence flow záznamu

Kontrola se provádí tak, že se iteruje skrz flow-cache. Flow-cache je implementována jako jednosměrně vázaný lineární seznam, kde každý tok je jednoznačně určen šestíci: zdrojová a cílová IP adresa, zdrojový a cílový port, ToS a protokolem. Pokud takovýto záznam v cache už existuje tak se pouze aktualizují položky jako počet paketů, počet bytů ve flow, provede se kumulativní OR příznaků TCP, a aktualizuje se čas příchodu posledního paketu flow. Pokud neexistuje tak se vytvoří nový záznam.

Přidání flow záznamu

U přidávání nového záznamu se prvně provede kontrola, jestli náhodou není plná flow-cache, pokud ano tak se exportuje nejstarší záznam, tedy první prvek jednosměrně vázaného seznamu. Pokud cache plná není tok se přidá do cache a nastaví se počet paketů této flow na 1. A vypočítá se čas prvního

a posledního paketu tohoto toku vzhledem k prvnímu zpracovávanému paketu podle vzorce: $\text{čas_současného} - \text{čas_prvního}$ a tento čas je v milisekundách. Tyto položky se uloží do políček `First` a `Last`, jedná se tedy o relativní čas vzhledem k prvnímu paketu.

Kontrola časovačů

Kontrola časovačů probíhá ještě před tím než se provede kontrola záznamu ve flow-cache. Kontrola probíhá tak, že se projde celá cache a nad každým záznamem se provede následující porovnání: vezme se aktuální čas zpracovávaného paketu odečte se od něj čas prvního paketu v dané flow záznamu a porovná se jestli je větší než čas aktivního časovače. Pokud ano tak se daný tok přidá do pole 30 záznamů, které se pak na konci procházení celé exportuje, protože lze najednou posílat až 30 záznamů.

Pro neaktivní časovač se od aktuálního času odečte čas posledního paketu v daném toku a pokud je větší než neaktivní časovač tak se daný tok opět uloží do pole, které se později exportuje.

Kontrola příznaků TCP

Úplně nakonci se provede ještě kontrola, jestli právě zpracovávaný paket je typu TCP a pokud obsahuje příznak `FIN` nebo `RST` tak se přidá do daného toku, ke kterému patří a pak se celý tok exportuje.

3.5 Odesílání flows

Odesílání jednotlivých flow záznamů probíhá ve funkci `send_flow`, kde se prvně připraví flow k odeslání ve funkci `prepare_flow`. V této funkci se alokuje dostatečně velká paměť pro všechny flow i s hlavičkou, která je definována ve struktuře `nf_v5_header_t`. Tato hlavička se naplní informacemi, které se převedou pomocí `htons` a `htonl` do síťového formátu. A to stejné se provede s informacemi v jednotlivých flow záznamech.

Poté funkce vrátí alokovaný a naplněný buffer. Dále se otevře buď IPv4 nebo IPv6 schránka. A poté se na danou IP adresu a port pošle tento buffer pomocí funkce `sendto()`.

4 Testování

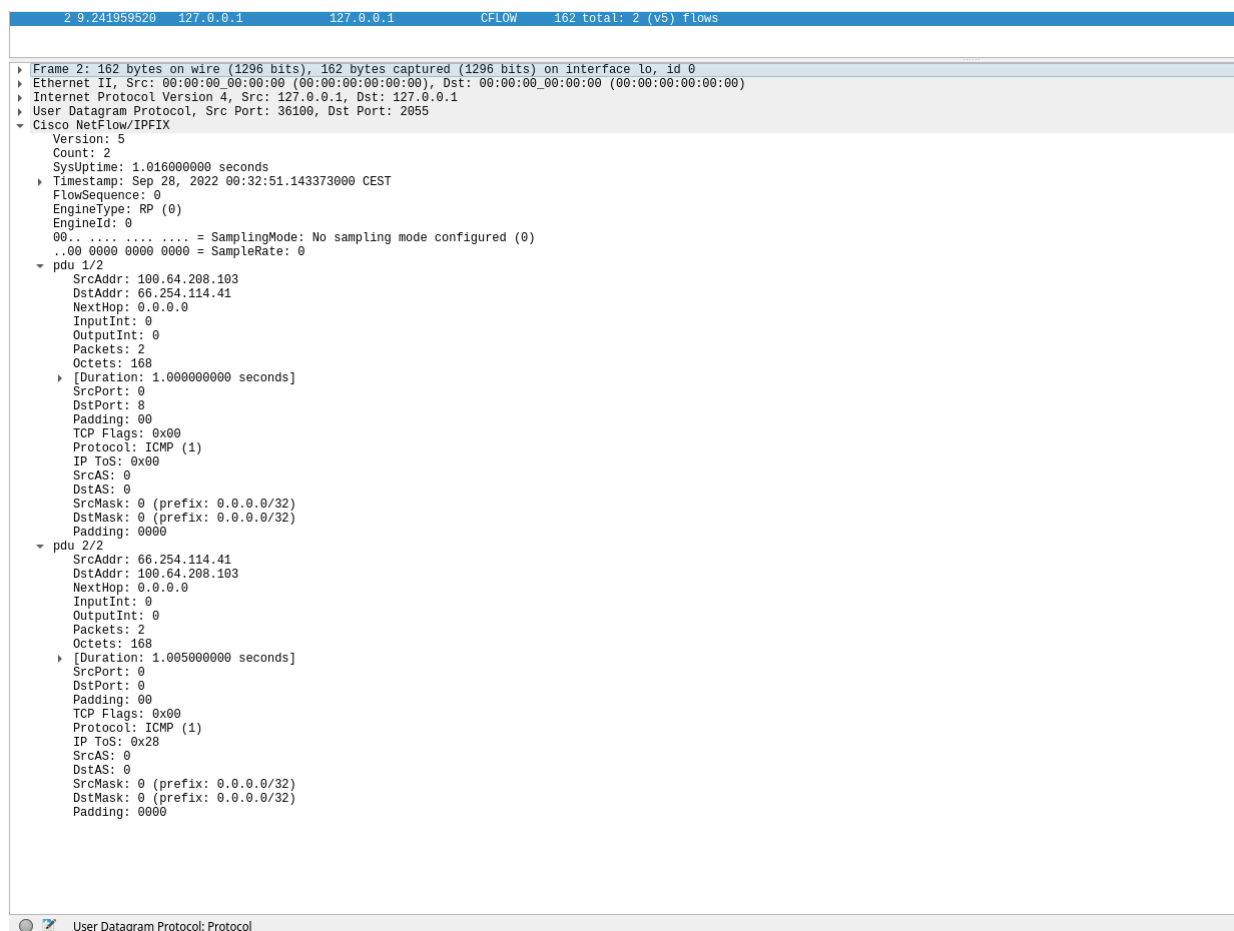
Testování bylo prováděno manuálně za pomoci nástrojů `Wireshark` [7], `nfdump` [9] a `nfcapd` [10]. Kdy jsme si první v nástroji `Wireshark` odchytil komunikaci a tu vyexportoval do souboru `pcap`. Následně byl spuštěn nástroj `nfcapd` následujícím příkazem: `nfcapd -D -T all -l /logs -I any -S 2 -p 2055 -t 60`, který zachytává netflow komunikaci na portu 2055 a ukládá ji do souboru. A poté byl spuštěn program, kde na vstupu byl vygenerovaný soubor `pcap`. A nakonec byla komunikace zobrazena pomocí nástroje `nfdump`: `nfdump -r logs/2022/*`. Dodatečně bylo ověřeno, že nástroj `Wireshark` zachytil odeslaný paket bez chyby.

```

~/skool/5.sem/isa/ISA_project main !2
> nfcapd -D -T all -l ~/skool/5.sem/isa/ISA_project/logs -I any -S 2 -p 2055 -t 60
~/skool/5.sem/isa/ISA_project main !2
> ./flow -f pcaps/icmp.pcap
~/skool/5.sem/isa/ISA_project main !2
> nfdump -r logs/2022/10/24/14/nfcapd.202210241415
Date first seen      Duration Proto      Src IP Addr:Port      Dst IP Addr:Port      Packets      Bytes Flows
2022-09-28 00:32:50.127 1.000 ICMP      100.64.208.103:0      -> 66.254.114.41:0.8      2            168      1
2022-09-28 00:32:50.138 1.005 ICMP      66.254.114.41:0      -> 100.64.208.103:0.0      2            168      1
Summary: total flows: 2, total bytes: 336, total packets: 4, avg bps: 2645, 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: 288
Sys: 0.001s flows/second: 1174.4      Wall: 0.000s flows/second: 16129.0

```

Obrázek 2: Ukázka spuštění jednotlivých programů a výstupu nástroje nfdump.



Obrázek 3: Zachycený paket nástrojem wireshark.

Reference

- [1] Carstens, T., Guy, H.. *Programming with pcap* [online]. 2022.
Dostupné z: <https://www.tcpdump.org/pcap.html>
- [2] Cisco Systems. *NetFlow Datagram Format* [online]. Dokumentace Cisco; 2014-09-17.
Dostupné z: www.cisco.com/c/en/us/td/docs/net_mgmt/netflow_collection_engine
- [3] Veselý, V. *Síťová vrstva – ICMP* [Univerzitní přednáška]. 2021.
Dostupné z: <https://wis.fit.vutbr.cz/FIT/st/cfs.php.cs?file=%2Fcourse%2FIPK-IT%2Flectures%2FIPK2021-06-IPv6.pdf&cid=14678>
- [4] Wikipedia contributors. *Transmission Control Protocol* [online]. Wikipedia, The Free Encyclopedia; 2022-04-09.
Dostupné z: https://en.wikipedia.org/wiki/Transmission_Control_Protocol
- [5] Wikipedia contributors. *User Datagram Protocol* [online]. Wikipedia, The Free Encyclopedia; 2022-03-24.
Dostupné z: https://en.wikipedia.org/wiki/User_Datagram_Protocol
- [6] Free Software Foundation. *getopt(3) — Linux manual page* [online].
Dostupné z: <https://www.man7.org/linux/man-pages/man3/getopt.3.html>
- [7] Combs, G. *Wireshark* [online] 2022. Dostupné z: <https://www.wireshark.org/>
- [8] Wikipedia contributors. *NetFlow* [online]. Wikipedia, The Free Encyclopedia; 2022-04-09.
Dostupné z: <https://en.wikipedia.org/wiki/NetFlow>
- [9] Free Software Foundation. *nfdump(1) — FreeBSD manual page* [online].
Dostupné z: <https://www.freebsd.org/cgi/man.cgi?query=nfdump>
- [10] Free Software Foundation. *nfcapd(1) — FreeBSD manual page* [online].
Dostupné z: <https://www.freebsd.org/cgi/man.cgi?query=nfcapd>
- [11] Jacobson, V., et al. *Man page of pcap* [online], 2020-09-09.
Dostupné z: <https://www.tcpdump.org/manpages/pcap.3pcap.html>