VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

ISA

Generování NetFlow dat ze zachycené síťové komunikace

MANUAL

30-10-2022

Václav Korvas (xkorva03)

Obsah

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 <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 <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 cachi 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 stardartního 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 oodevzdaného archivu

- `Makefile`: s cíly `all` (sestaví výsledný program flow.cpp), `clean` (vymaže všechny vzniklé binární soubory a spustitelný program), `archive` (vytvoří archvive 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 přeloži pomocí příkazu `make` nebo pomocí příkazu `make all`. Tímto vznikne v spustitelný soubor v kořenové složce adresáře, ve kterém je spuštěn příkaz `make`. Po přeložení aplikace ji lze jednodušš 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 zadávaní 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

TODO

3 Implemtační detaily

V následující sekci se popisuje strukturu programu a popisuje řešení některých nejduležitějších 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 knihovní funkce byly z této stránky

Využité knihovny

- `pcap/pcap.h` - knihovna na zachytavání, procházení a filtrovaní paketů,
- `getopt.h` - zpracování argumentů příkazové řádky,
- `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 okamžite 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 Zpracovaní 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 funkci něuspěje 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 úspechu tak funkce vrátí hodnotu 1 a běh programu se ukončí s korespondující chybovou hláškou.

3.3 Zpracovaní paketů

Po zpracovaní 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 flows. 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, cilový a zdrojový port a protokolů UDP a TCP. U prvního paketu si uložím jeho čas příchodu a považuji jej za systémový čas, který už nikdy neměním a je uložen v proměnné `firt_packet_time`. A každý následující čas paketu je počítán relativně k tomuto času.

3.4 Vytvaření a zpracování flows

Po extrakci všech důležitých informací z hlaviček se volá funkce `process_flow`. Zde se děje hned několik důležitých věcí a to: kontrola, jestli flow již existuje ve flow-cachi, pokud ne tak se vytvoří nový flow záznam a uloží se do cache. Poté zde probíha kontrola jednotlivých podmínek, při jejichž porušení dochazí k exportování flow záznamů. Mezi tyto podmínky patři 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 linearní seznam, kde každá flow je jednoznačně určena šesticí: `zdrojová a cílová IP adresa, zdrojový a cílový port, ToS a protokolem`. Pokud takovýto záznam v cachi 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 flow se přidá do cache a nastaví se počet paketů této flow na 1. A vypočítá se čas

prvího a posledního paketu této flow vzhledem k prvnímu zpracovávanému paketu podle vzorce: `čas současného - 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 probíha ještě před tím než se provede kontrola záznamu ve flow-cachi. Kontrola probíha tak, že se projde celá cache a nad každým záznamem se provede následujíci porovnání: vezme se aktualní čas zpracovávaného paketu odečte se od nej čas prvního paketu v dáné flow záznamu a porovná se jestli je větší než čas aktivního časovače. Pokud ano tak se dana flow 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 aktualního času odečte čas posledního paketu v dané flow a pokud je větší než neaktivní časovač tak se daná flow opět uloží do pole, které se později exportuje.

Konrola příznaků TCP

Úplně nakonci se provede ješte kontrola, jestli pravě zpracovávaný paket je typu TCP a pokud obsahuje příznak `FIN` nebo `RST` tak se přídá do dané flow ke které patří a pak se celá flow exportuje.

3.5 Odesílaní 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 veliká pamět 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 vratí 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

TODO

Reference

[1] Carstens, T., Guy, H.. *Programming with pcap* [online]. 2022. [accessed 2022-04-17]. Available from: https://www.tcpdump.org/pcap.html

[2] Wikipedia contributors. *OSI model* [online]. Wikipedia, The Free Encyclopedia; 2022-04-14. [accessed 2022-04-17]. Available from: https://en.wikipedia.org/wiki/OSI_model

[3] Wikipedia contributors. *Ethernet frame* [online]. Wikipedia, The Free Encyclopedia; 2022-04-01. [accessed 2022-04-17]. Available from: https://en.wikipedia.org/wiki/Ethernet_frame

[4] Wikipedia contributors. *Address Resolution Protocol* [online]. Wikipedia, The Free Encyclopedia; 2022-04-06. [accessed 2022-04-17]. Available from: https://en.wikipedia.org/wiki/Address_Resolution_Protocol

[5] Veselý, V. *Síťová vrstva – IPv4* [University lecture]. 2021. Available from: https://wis.fit.vutbr.cz/FIT/st/cfs.php.cs?file=%2Fcourse%2FIPK-IT%2Flectures%2FIPK2021-06-IPv6.pdf&cid=14678

[6] Veselý, V.. *IPv6 síťová vrstva* [University lecture]. 2021. Available from: https://wis.fit.vutbr.cz/FIT/st/cfs.php.cs?file=%2Fcourse%2FIPK-IT%2Flectures%2FIPK2021-04-IPv4.pdf&cid=14678

[7] Wikipedia contributors. *Transmission Control Protocol* [online]. Wikipedia, The Free Encyclopedia; 2022-04-09. [accessed 2022-04-17]. Available from: https://en.wikipedia.org/wiki/Transmission_Control_Protocol

[8] Wikipedia contributors. *User Datagram Protocol* [online]. Wikipedia, The Free Encyclopedia; 2022-03-24. [accessed 2022-04-17]. Available from: https://en.wikipedia.org/wiki/User_Datagram_Protocol

[9] Free Software Foundation. *getopt(3) — Linux manual page* [online], [accessed 2022-04-17]. Available from: https://www.man7.org/linux/man-pages/man3/getopt.3.html

[10] Jacobson, V., et al. *Man page of pcap* [online], 2020-09-09 [accessed 2022-04-17]. Available from: https://www.tcpdump.org/manpages/pcap.3pcap.html

[11] Sebastian. *C++ RFC3339 timestamp with milliseconds using std::chrono* [online], [accessed 2022-04-17]. Available from: https://stackoverflow.com/questions/54325137/c-rfc3339-timestamp-with-milliseconds-using-stdchrono

[12] Haskins, K. *I'm trying to build an RFC3339 timestamp in C* [online],[accessed 2022-04-17]. Available from: https://stackoverflow.com/questions/48771851/im-trying-to-build-an-rfc3339-timestamp-in-c-how-do-i-get-the-timezone-offset

[13] Combs, G. *Wireshark* [online] 2022,[accessed 2022-04-17]. Available from: https://www.wireshark.org/