VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

PCAP NETFLOW V5 EXPORTÉR
PCAP NETFLOW V5 EXPORTER

ISA - PROJEKT
PROJECT

AUTOR PRÁCE
AUTHOR

PATRIK UHER

BRNO 2024

Obsah

Seznam obrázků

Kapitola 1

Úvedení do problematiky

uvedení do problematiky, návrhu aplikace, popis implementace, základní informace o programu, návod na použití, popis testování aplikace a výsledky testů

Netflow je technologie původně pocházející od cisco routerů[1], která umožňuje sbírat síťový provoz a z něho vytvořit toky. Aby paket byl součástí toku, musí mít stejnou ip adresu zdroje, ip adresu cílu, port zdroje, port cílu a typ protokolu. Pakety u kterých jsou stejné tyto pole, se shromáždí do jednoho toku.

Jeden tok má různé atributy jako celkový počet paketů, celkový počet octetů třetí vrstvy, casovou značku prvního paketu přidaného do toku, časovou značku posledního paketu přidaného do toku, tcp flagy, typ protokolu a další. Samozřejmně je součástí jednoho toku i data podle kterých se do toku přidávají pakety.

Kapitola 2

Návrh aplikace

Aplikace je naprogramovaná v C++, kde na čtení pcap souborů se používají funkce z knihovny pcap. Na posílání paketů se používají API BSD soketů.

2.1 Uspořádání kódu

Aplikace je rozdělená na několik částí, kde každá podstatná část je ve svém vlastním souboru. Soubor "p2nprobe.cpp"je soubor ve kterém je funkce main. Dále existuje třída "client-args"která řeší parsování argumentů od uživatele. Soubor "debug-info"je na vytvoření globální proměné, která řeší zda má program vypisovat ladící informace. Třída "flow"je na vytvoření a ukládání do samotného toku. Třída "flow-manager"řesí do jakého toku se paket přidá. Nakonec třída "packet-composer"samotný paket který se dá poslat přes sockety.

2.2 Popis implementace

Soubor "p2nprobe"vytváří důležité globální promněné, inicializuje ostatní třídy a provádí hlavní smyčku. Na začátku posílá parseru na argumenty, argumenty od uživatele. A začíná číst pakety které jsou v pcap souboru, který program musí dostat jako jeden z argumentů. Pakety hned čte a ukladá je do toků (z testování různých pcap souborů bylo zjistěno že pakety jsou uložené chronologicky a tedy je nemusíme sami třídit). Potom co program prošel celý pcap soubor a vytvořil všechny toky, se všechny toky (aktivní a neaktivní) uloží do datové struktury, kde se protřídí podle identifikačního čísla toku. Výsledkem je datová struktura s toky seřazenými podle toho jak byly vytvořeny (tedy chronologicky).

Každý paket se přidá do toku přes třídu "flow-manager". "flow-manager"vytvoří index paketu podle šablony "[src ip][dst ip] [src port][dst port][packet type]". Podle specifikace tohle je unikátní seskupení a tedy to můžeme použít jako index to hašovací mapy (tahle datová struktura byla vybrána z důvodu rychlého vyhledání příslušného toku). Poté se provede test na aktivní timeout a neaktivní timeout a podle toho se rozhodne jestli se má tok přemístit do vektoru neaktivních toků (vector byl vybrán díky rychlému přidávání prvků a díky tomu že mezi neaktivnímy toky už nemusíme hledat).

Další důležitá část implementace je třída "packet-composer", která vytváří paket jako řetězec do kterého se přidávají čísla jiných velikostech. Tahle část kódu je ukázaná níže:

```
    // 1 flow has 48 bytes and the header has 24 bytes
    pc::allocate_packet(24 + (flow_buffer.size() * 48));

    // netflow v5 header
    pc::packet_add_uint16(5);
    pc::packet_add_uint16(flow_buffer.size());
```

kde metoda packet `packet_add_uint16()` vypadá takhle:

```
/** A method for adding 2 byte numbers into a packet
*/
void pc::packet_add_uint16(uint16_t number)
{
 if(pc::packet == nullptr)
 {
  cerr << "Error: Packet is not initialized!" << endl;
  exit(1);
 }

 if(pc::packet_free_bytes < sizeof(number))
 {
  cerr << "Error: Trying to copy to out of bounds!" << endl;
  exit(1);
 }

 uint16_t h_number = htons(number);

 auto ret = memcpy(reinterpret_cast<void *>(pc::packet_end), reinterpret_cast<void *>(&
    if(ret == nullptr)
    {
        cerr << "Error: memcpy failed to copy to packet" << endl;
  exit(99);
    }

 pc::packet_end += sizeof(number);
 pc::packet_free_bytes -= sizeof(number);
 pc::packet_member_count++;
}
```

Jak jde z kódu vydět tak přes atributy jako je `packet_free_bytes` se můžeme ujistit že nezapisujem za alokované místo. Jiné atributy jako `packet_member_count` jsou jen jako statistika a využívají se při ladících výpisků.

2.3 časové značky

Časové značky fungují trochu jinak než normální netflow exportér a to protože načítají pakety z pcap souboru a né z interface. Tím pádem je skoro jasné že pakety přišly a byli

zapsány do pcap souboru před tím než se aplikace na export spustila. Tohle znamená že že políčka v tocích jako je `first` nebo `last`, budou v minulosti. Program to řeší, jak bylo napsané na fóru, tak že vytvoří relativní časové značky od té doby když začal program. V průběhu programu jsou tyhle značky v microsekundách (vzhledem k tomu že pakety můžou dojít o 1 microsekundu od sebe a program je musí rozlišit), ale při exportování toku jsou časové značky přetypovány na milisekundy (tohle je dáno specifikací netflow v5[2]).

Kapitola 3

Návod na použití aplikace

Vysvětlení všech informací, které jsou potřebné k používání programu, vysvětlí help tabulka:

Usage

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

Arguments

- `<host>` → IP address or the domain name of a NetFlow V5 collector
- `<port>` → port of the NetFlow V5 collector

Flags

- `[-h | -help]` → Displays this help message and exits
- `[-d | -debug]` → Enables debugging (warning: will print out a lot of text to standard output)
- `[-v | -version]` → Displays the version of this program and exits
- `[-a | -active <active timeout>]` → number of seconds for the active timeout of the NetFlow V5 exporter (default 60 seconds)
- `[-i | -inactive <inactive timeout>]` → number of seconds for the inactive timeout of the NetFlow V5 exporter (default 60 seconds)

Způsob použití může být následný:

`‘./p2nprobe 127.0.0.1:2552 ./pcaps/10packets.pcap -a 15 -i 21‘`

spustí program aby na adresu 127.0.0.1, na port 2552 poslal packety NetFlow v5, kde packety jsou ze souboru `./pcaps/10packets.pcap`. Aktivní timeout je nastaven na 15 sekund a neaktivní timeout na 21 sekund.

`‘./p2nprobe localhost:1234 ./pcaps/249packets.pcap‘`

spustí program aby na adresu 127.0.0.1, na port 1234 poslal packety NetFlow v5, kde packety jsou ze souboru `./pcaps/249packets.pcap`. Aktivní timeout je nastaven na 60 sekund a neaktivní timeout na 60 sekund.

`‘./p2nprobe 192.168.10.5:1456 ./pcaps/1packet.pcap --debug‘`

spustí program aby na adresu 192.168.10.5, na port 1456 poslal packety NetFlow v5, kde packety jsou ze souboru `./pcaps/1packet.pcap`. Aktivní timeout je nastaven na 60 sekund a neaktivní timeout na 60 sekund. Dále jsou nastaveny ladící výpisky, které vypíšou veškeré důležité informace o programu (varování: program vypíše hodně textu na standardní výstup).

`‘./p2nprobe --help‘`

vypíše pomocní text na to jak program používat.

Literatura

[1] Cisco. *NetFlow Export Datagram Format* online. Dostupné z: https://www.cisco.com/c/en/us/td/docs/net_mgmt/netflow_collection_engine/3-6/user/guide/format.html#wp1006108. [cit. 2024-11-18].

[2] IBM. *NetFlow V5 formats* online. Dostupné z: https://www.ibm.com/docs/en/npi/1.3.0?topic=versions-netflow-v5-formats. [cit. 2024-11-18].