VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta informačních technologií

Síťové aplikace

2016/2017

Projekt – Analyzátor síťového provozu

Roman Nahálka, xnahal01

V Brně dne 16.11.2016

Obsah

1 Úvod

Tato dokumentace vznikla k projetu do předmětu Síťové aplikace a správa sítí (ISA). Cílem projektu je analyzovat zachycenou a uloženou internetovou komunikace v souboru pcap.

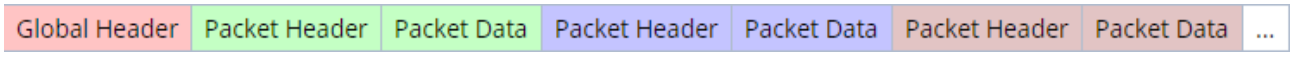
2 Analýza problému

2.1 Zadání problému

Program má analyzovat síťoví provoz uložený ve formátu libpcap. Program má za úkol počítat počet přenesených bajtů na základě filtru, zadaného uživatelem jako argument programu. Možné filtry jsou MAC, IPv4. IPv6, TCP a UDP. K filtru se také nastavuje hodnota. Zadaný filtr se poté aplikuje buď aplikuje na zdroj nebo na cíl, popřípadě na obě dvě možnosti. Nakonec lze filtru přiřadit speciální hodnotu, top10, která vypíše 10 položek s největším počtem bajtů, seřazených od největšího po nejmenší. Výstupem programu budou proměnné *hodnota1* a *hodnota2*. *Hodnota1* je součet všech bajtů od druhé vrstvy. *Hodnota2* je jiná pro každý filtr. Pro filtr MAC se počítají data od konce L2 hlavičky, pro IPv4 a IPv6 od konce L3 hlavičky a pro TCP a UDP data od konce L4 hlavičky. Hodnoty se poté vytisknou na standartní výstup v podobě *hodnota1_hodnota2*. V případě filtru top10 bude výstup maximálně 10 řádků vypadat *adresa_hodnota1_hodnota2*.

2.2 Formát Libpcap

Libpcap formát [1] ukládá zachycené pakety např. pomocí programu Wireshark. Struktura toho souboru je velice jednoduchá. Na začátku souboru se nachází Globální Hlavička, následována žádnými nebo více zachycenými pakety. Paket je složen z Hlavičky Paketu a s Daty Paketu. Pakety se v souboru nacházejí neodděleně za sebou, jak je ukázáno na následujícím obrázku.



Obr.1: Formát souboru Libpcap

2.3 Globální Hlavička

Jak bylo uvedeno výše, každý soubor formátu libpcap začíná Globální Hlavičkou, která má tuto podobu:

```
typedef struct pcap_hdr_s {
        guint32 magic_number;
        guint16 version_major;
        guint16 version_minor;
        gint32  thiszone;
        guint32 sigfigs;
        guint32 snaplen;
        guint32 network;
} pcap_hdr_t;
```

Obr.2: Globální Hlavička

Každá Globální Hlavička tedy obsahuje magické číslo, verzi souboru, subverzi souboru, časovou zónu, přesnost časového razítka, velikost snímků a délku L2 hlavičky. Pro naši aplikace není tato hlavička důležitá.

2.4 Hlavička Paketu

Každý zachycený paket začíná touto hlavičkou, která má tuto podobu:

```
typedef struct pcaprec_hdr_s {
        guint32 ts_sec;
        guint32 ts_usec;
        guint32 incl_len;
        guint32 orig_len;
} pcaprec_hdr_t;
```

Obr.3: Hlavička Packetu

Každá Hlavička Paketu tedy obsahuje časové razítko zachycení, délku zachycených dat a původní velikost dat. Tato hlavička nás na rozdíl od té předchozí již bude zajímat. Konkrétně nás bude zajímat hodnota *orig_len*, kterou využijeme při čtení souboru a při počítání druhé výstupní hodnoty.

2.5 Data Paketu

Po hlavičce paketu následují data paketu. Ty obsahují hlavičky L2, L3, L4 a

samotná data. Ne všechny rámce ale obsahují všechny hlavičky a je potřeba s tím v aplikaci počítat.

3. Popis řešení

Aplikace musí číst soubor paket po paketu. V případě, že byl aplikace zadán filtr *MAC*, zkontroluje se hodnota filtru s hodnotou v L2 hlavičce. Jestliže se hodnoty rovnají, aplikace přičte k hodnotám *hodnota1* a *hodnota2* příslušný počet paketů a pokračuje ve četní dalšího souboru až do konce souboru. Obdobně bude aplikace pracovat i v případě ostatních filtrů. Pouze se v případě filtrů *IPv4* a *IPv6* budou hodnoty porovnávat v L3 hlavičce a v případě filtrů *TCP* a *UDP* v L4 hlavičce.

Trošku odlišně se bude aplikace chovat v případě, že uživatel zadá hodnotu filtru *top10*. V takovém případě nejdříve aplikace projede celý soubor za účelem naleznutí všech možných hodnot zadaného filtru. Aplikace všechny tyto hodnoty uloží a poté s nimi bude postupně pracovat jako v normálním případě. Po provedení výpočtů všech hodnot filtrů aplikace seřadí všechny hodnoty podle velikosti a největších 10 hodnot vytiskne na standartní výstup.

4 Implementace

Program je implementován v jazyce C/C++. Program není navržen objektově, ale používá některé objekty ze standartních knihoven jazyka C++. Aplikace je navržena pro OS Linux, přenositelnost na OS Windows není testována. Aplikace byla vyvíjena a testována pouze na linuxových distribucích Ubuntu, konkrétně na verzích 16.04 (64-bit) a 14.04 (32-bit). V obou případech se jednalo o virtuální počítač, běžící pod programem VritualBox ve Windows 10.

5 Použití aplikace

Program se spouští následujícím způsobem:
./analyzer [-i soubor] [-f typ] [-v hodnota] [-s] [-d]
Jednotlivé argumenty jsou následující:
Argument **-i** je povinný parametr a jedná se o vstupní soubor ve formátu libpcap.
Argument **-f** je povinný parametr a určuje podle které položky se bude počítat objem dat. Možné hodnoty argumentu jsou: MAC, IPv4, IPv6, TCP a UDP.
Argument -v je povinný parametr a určuje hodnotu zadaného filtru. Možné hodnoty jsou např. Pro MAC 00:00:00:00:00:01 (mac adresa), pro IPv4 192.168.0.1 (ipv4 adresa), pro IPv6 2001::1 (ipv6 adresa) a pro TCP a UDP např. 80 (číslo portu.)
Argument **-s** zajistí, že se filtr aplikuje na zdrojové adresy.
Argument **-d** zajistí, že se filtr aplikuje na cílové adresy.
Alespoň jeden z argumentů **-s** a **-d** musí být zadán.

5.1 Příklad spuštění

./analyzer -i isa.pcap -f tcp -v 101 -s -d
Výstup: 6162_5484

5.2 Chybové kódy

Pokud při běhu programu dojde k chybě, program vypíše na standartní chybový výstup patřičné hlášení a ukončí program s návratovým kódem odpovídající chybě.

0 – program skončil bez chyby
1 – Chyba při kontrole parametrů
2 – Zadaný soubor nelze otevřít

6 Závěr

Program analyzuje zadaný pcap soubor podle zadaného filtru. Funkčnost aplikace byla ověřena na virtuálních strojích Ubuntu 14.04 a 16.04. Program je překládán překladačem *g++*. Pro překlad aplikace je přiložen soubor *Makefile*.

A Metriky kódu

Počet souborů: 2

Celkový počet řádku zdrojového textu: 1079

Literatura

[1] https://wiki.wireshark.org/Development/LibpcapFileFormat