VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta informačních technológií

Počítačové komunikace a sítě 2020

Manuál k programu

Sniffer paketů - Varianta ZETA

Damián Sova(xsovad06)

Senica, 1.5.2020

OBSAH

ÚVOD

Znenie zadania programu je nasledovné: „Navrhněte a implementujte síťový analyzátor v C/C++/C#, který bude schopný na určitém síťovém rozhraním zachytávat a filtrovat pakety.“

ODOVZDANÉ SÚBORY

Odovzdaný je jeden skomprimovaný súbor xsovad06.tar, ktorý obsahuje:

Ipk-sniffer.cpp

Súbor s funkčným zdrojovým kódom, implementáciou jednotlivých pomocných funkcií a funkcie main.

Header.hpp

Hlavičkový súbor obsahujúci importované knižnice, globálne premenné, deklaráciu pomocných štruktúr, deklaráciu triedy *Packet_time* a prototypy pomocných funkcií.

Makefile

Súbor obsahujúci príkaz na kompiláciu a príkaz na vymazanie súboru s príponou .o

README.md

Súbor obsahuje základné informácie o programe, jeho funkčosti, používaní a zoznam odovzdaných súborov.

NÁVRH RIEŠENIA

1. Začneme určovaním rozhrania, na ktorom chceme odchytávať balíčky. V Linuxe to môže byť niečo ako eth0, v BSD to môže byť xl1, atď. Toto zariadenie definujeme buď v reťazci cez argumenty alebo môžeme požiadať pcap, aby nám poskytol názov rozhrania, ktoré bude odchytávať balíčky.
2. Inicializujeme pcap. To je miesto, kde skutočne povieme pcap, na ktorom zariadení odchytávame. Ak chceme, môžeme odchytávať na viacerých zariadeniach. Používanie popisovačov súborov. Rovnako ako pri otváraní súboru na čítanie alebo zápis, musíme pomenovať našu „odchytávaciu“ reláciu, aby sme ju mohli odlíšiť od iných takýchto relácií.
3. Ak chceme odchytávať len určité balíčky(len balíčky TCP/UDP, iba balíčky smerujúce na port 23, atď.) musíme vytvoriť množinu pravidiel, „kompilovať“ ju a použiť. Je to trojfázový proces, ktorý úzko súvisí. Sada pravidiel je udržiavaná v reťazci a je konvertovaná do formátu, ktorý dokáže pcap prečítať (teda ju skompilovať). Kompilácia sa v skutočnosti robí iba vyvolaním funkcie v našom programe. Nezahŕňa použitie externej aplikácie. Potom povieme pcap, aby ho použil na ktorúkoľvek reláciu, ktorú si želáme, aby sa filtroval.
4. Nakoniec povieme pcap, aby zadal svoju primárnu spúšťaciu slučku. V tomto stave čaká pcap, kým nedostane toľko paketov, koľko chceme. Zakaždým, keď dostane nový paket, volá ďalšiu funkciu, ktorú sme už definovali. Funkcia, ktorú volá, môže robiť všetko, čo chceme, môže vypísať obsah, môže ho uložiť do súboru alebo nemusí urobiť vôbec nič.
5. Po splnení našich potrieb „sniffingu“ ukončíme reláciu a sme hotoví.

IMPLEMENTÁCIA

Pre účel implementácie bol z možných programovacích jazykov vybraný C++. Program nie je objektovo-orientovaný, i keď je použitá jedna trieda, ktorá reprezentuje prácu s časom balíčku. Srdcom celého „snifferu“ je práca s knižnicou *<pcap.h>* . Ako bolo spomínané v návrhu riešenia, tak sa postupovalo aj pri implementácii.

Úplne prvá vec programu, ktorá sa vykoná je spracovanie argumentov. To sa deje prostredníctvom funkcie *int arg_processor(int argc, char** argv).* Tá preberá počet argumentov a pole týchto argumentov. V tele prostredníctvom funkcie *getopt_long* spracuje argumenty, tie sa prostredníctvom „switchu“ spracujú.

Ďalej prichádza na definovanie zariadenia na odchytávanie balíčkou. Ak nebolo zadané argumentom programu, pomocou funkcie *pcap_findalldevs* zo spomínanej knižnice zistí všetky dostupné zariadenia. Tie sa vypíšu na štandardný výstup a program sa úspešne ukončí. Ak bol zadaný, pokračuje sa funkciou *pcap_lookupnet* , tá vracia jedno zo svojich sieťových čísel IPv4 a zodpovedajúcu sieťovú masku. Pokračujeme otvorením daného zariadenia funkciou *pcap_open_live.* Skontrolujeme, či otvorené zariadenie je ethernet, ak nie, ukončujeme program chybou.

Prichádza na rad filtrovanie a proces jeho realizácie. Táto časť procesu je závislá na argumentoch programu, ak neboli zadané, filtrovanie sa preskakuje a pokračuje sa ďalej. Ak bol zadaný niektorý, prípadne kombinácia týchto argumentov, filtrovanie prebehne:

- -p <číslo_portu> - filtrovanie zadaného čísla portu
- --tcp / -t - filtrovanie packetov(balíčkou) s TCP protokolom
- --udp/-u - filtrovanie packetov(balíčkou) s UDP protokolom

Posledný krok je pustiť cyklus na odchytenie požadovaného počtu balíčkov a jeho nasledovné spracovanie našou funkciou *packet_processor*, ktorú rozoberieme neskôr. Na ustanovenie tejto slučky/cyklu slúži funkcia *pcap_loop*.

Po vykonaní celej slučky sa zavolá funkcia *pcap_close*, ktorá uzavrie spojenie. Týmto je proces odchytávania na konci a program sa úspešne ukončí.

IMPLEMENTÁCIA POMOCNÝCH FUNKCIÍ

void print_help() Vypíše pomocnú správu, ak je zadaný samostatný argument programu --help/-h.

int arg_processor(int argc, char** argv) Funkcia na spracovanie argumentov, ktorú už sme popísali.

void packet_processor(unsigned char *args, const struct pcap_pkthdr *header, const unsigned char *packet) Najdôležitejšia funkcia, ktorá je srdcom celého programu. Funkcia pracuje so samotným packetom. Zadefinuje si premenné, prvé vytvorí inštanciu triedy *Packet_time*, tú si popíšeme neskôr. Naplní ju časovými údajmi: hodiny, minúty, sekúnd a mikrosekundy, vypočítanými z časového údaju uloženého v štruktúre hlavičky packetu. Vytvorí si pointer na IP hlavičku pomocou pointerovej aritmetiky, posunie pointer začiatku packetu o stálu šírku ethernetovej hlavičky. Ak je veľkosť IP nedostatočná ukončujeme spracovávanie tohto packetu. Z IP hlavičky si vytiahneme všetko, čo potrebujeme. Napríklad IP adresu zdroju packetu, ako aj jeho cieľ. Tieto adresy si preložíme na doménové meno funkciou *getnameinfo*. Ak sa nepodarí preložiť, použijeme IP adresu pre neskorší výpis. Ďalej podľa zadaného argumentu na protokolu packetu vyberieme vetvu pokračovania. Rozdiel je, že ak neboli zadané parametre na protokol, tak sa explicitne kontroluje z IP hlavičky. Ak bol zadaný práve jeden, protokol je už vyfiltrovaný a nie je treba ho kontrolovať. Po tomto kroku sú vetvy rovnaké. Pomocou objektovej metódy *void Packet_time:: print_time()* sa vypíše čas packetu. Ďalej vypíše IP adresu/doménové meno zdrojového zariadenia, zdrojový port, IP adresu/doménové meno cieľového zariadenia a cieľový port. Následne sa volá funkcia *print_packet*, ktorá vypíše obsah packetu v požadovanom formáte. Týmto je ukončené obstarávanie jedného balíčku.

void print_packet(const unsigned char *packet, int len) Funkcia vypisuje kompletný obsah packetu. Preberá pointer na packet a jeho dĺžku. V cykle počíta aktuálnu dĺžku riadku a volá funkciu *print_hex_ascii* na výpis jednotlivých riadkov packetu.

int print_hex_ascii(const unsigned char *packet, int len, int printed_bytes) Funkcia vypisuje obsah packetu, najprv hexadecimálne a potom pomocou ASCII znakov. Vypíše počet vypísaných bajtov

hexadecimálne, potom obsah packetu hexadecimálne a nakoniec obsah packetu pomocou ASCII znakou.

Pomocná trieda na sprehľadnenie práce s časom a vyskúšanie objektovo-orientovaného programovania:

```
class Packet_time {
    private:
        int hours;
        int minutes;
        int seconds;
        suseconds_t u_seconds;
    public:
        void set_time(int hh, int mm, int ss, suseconds_t us);
        void print_time();
}
```

Kde v private sú deklarované premenné, ku ktorým je možné pristupovať jedine cez metódy *set_time* a *print_time*. Prvá uloží hodnoty z parametrov funkcie do objektových premenných a druhá vypíše tieto hodnoty v požadovanom časovom formáte.

POUŽITÉ MATERIÁLY

https://www.tcpdump.org/pcap.html

https://stackoverflow.com/questions/22206791/packet-sniffer-under-linux-in-c-c

https://stackoverflow.com/questions/29822809/error-in-use-of-the-pcap-findalldevs-ex-function-in-c

https://stackoverflow.com/questions/530614/print-leading-zeros-with-c-output-operator

https://stackoverflow.com/questions/34362095/getnameinfo-can-it-be-used-to-return-multiple-hostnames-for-a-single-ip-addres

https://stackoverflow.com/

https://www.google.com/