VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Počítačové sítě a komunikace – 2. projekt

Varianta ZETA: Sniffer paketů

24. dubna 2022

Bitomský Marek (xbitom00)

1 Úvod

Zadáním projektu bylo navrhnout a implementovat síťový analyzátor v C/C++/C#. Já jsem si zvolil jazyk C. Tento program je schopný na určitém rozhraní zachytávat a filtrovat pakety podle protokolů `TCP`, `UDP`, `ICMP` nebo `ARP`, dále lze ještě filtrovat podle portu.

2 Spuštění programu a kompilace

Program je funkční na linuxových systémech, kde je k dispozici `GNU Make` a `gcc`. Aplikace byla testována na systému Ubuntu 20.04.

2.1 Kompilace

Kompilování se provádí pomocí příkazu make, ke kterému je potřeba soubor nacházející se v adresářové struktuře Makefile.

Pro sestavení zavoláme na příkazové řádce:

```
$ make
```

2.2 Spuštění

Aplikace se spouští pomocí příkazu:

```
$ sudo ./ipk-sniffer [-i rozhraní | --interface rozhraní] {-p port}
                     {[--tcp|-t] [--udp|-u] [--arp] [--icmp] }
                     {-n num}|
```

- `-h` nebo `--help` vypíše nápovědu, volá se i při chybě
- `-i rozhraní` nebo `--interface rozhraní` vyžadovaný argument, po zadání bude vypisovat pouze pakety z daného rozhraní při chybě se vypíše list rozhraní
- `-p port` nebo `--port port` nastaví port, pro který budou vypisovány pakety, bez udání hodnoty vypisuje všechny porty
- `-t` nebo `--tcp` nastaví filtr jen na tcp pakety, lze kombinovat s ostatními protokoly, bez udání nějakého protokolu se vypisují všechny možnosti
- `-u` nebo `--udp` nastaví filtr jen na udp pakety, lze kombinovat s ostatními protokoly
- `-a` nebo `--arp` nastaví filtr jen na arp pakety, lze kombinovat s ostatními protokoly
- `-m` nebo `--icmp` nastaví filtr jen na icmp pakety, lze kombinovat s ostatními protokoly
- `-n počet` nebo `--num počet` nastaví kolik paketů se vypíše, bez udání hodnoty vypisuje pouze jeden

3 Ukončení

Program se ukončí v případě, že dojde k některé z níže uvedených variant a to s náležitým návratovým kódem.

- Jesliže vše proběhlo bez chyby vrací se hodnota 0
- V případě, že došlo k jakékoliv chybě ať už se jedná o chyby špatně zadaných argumentů nebo nějaké chyby interně způsobené, vrací se hodnota 1

4 Implementace

Program je kompletně implementován v jazyce C s využitím některých síťových knihoven jako je `pcap.h`, `arpa/inet.h` nebo `netinet/ether.h`. V souboru `ipk-sniffer.h` se nacházejí prototypy funkcí a použité struktury, které byli vytvořeny dle standardů, např. pro ARP [2]. V souboru `ipk-sniffer.c` se nachází kód s výpisy dat, pro IPv6 [1] použitý standard.

4.1 Argumenty

Argumenty příkazové řádky řeším ve vlastní funkci `check_args()`, která prochází, co bylo zadáno a jestli všechno sedí, tak jak by mělo. Dle zadaných argumentů plním datovou strukturu `settings`:

```
struct settings
    {
        char *port;
        char *interface;
        bool tcp;
        bool udp;
        bool icmp;
        bool arp;
        int num;
    };
```

4.2 Vytvoření síťového adaptéru

Implementace síťového adaptéru připojující se na existující síť se nachází ve funkci `main()`. Pro správnou funkčnost je potřeba zadat při spuštění `-i rozhraní`, aby bylo možné připojit se k nějakému z nich.

Po nastavení rozhraní se volá funkce `pcap_lookupnet()`, která získá síťovou adresu a masku a také je potřeba pro nadcházející kroky. Při všech voláních funkcí z knihovny `pcap` se používá proměnná `error_buffer`, kam se uloží chyba, která může nastat při volání těchto funkcí a vypíše se.

V tento moment je možné zavolat funkci `pcap_open_live()`, která začne zachytávat pakety z rozhraní.

Když už se zachytávají pakety, tak se vytváří filtr, podle argumentů. Na to slouží funkce `create_filter()`, která vytváří textový filtr typu `tcp or udp` apod. Tento vygenerovaný řetězec dostane funkce `pcap_compile()`, která ho zkompiluje pro to, aby šel použít ve funkci `pcap_setfilter()` a tím se nastaví, co všechno se bude filtrovat.

Jakmile je vše nastaveno, tak se volá funkce `pcap_loop()`, která získá zpracovaný paket a pošlou se data do callback funkce u mě pojmenovanou jako `parse_packet()`.

4.3 Analýza paketu

Ve funkci `parse_packet()` se získá hlavička paketu, ve které se nachází mac adresy, které se ihned vypíší a postupuje se ke zpracování podle protokolu v hlavičce. Pro každý protokol je určená speciální funkce a tato funkce vypisuje na standardní výstup data ve formátu:

```
timestamp: 2022-04-24T22:31:19+02:00
src MAC: ff:ff:ff:ff:ff:ff
dst MAC: 0:15:5d:f:e4:d9
frame length: 305 bytes
src IP: 172.17.128.1
dst IP: 172.17.143.255
src port: 54915
dst port: 54915
0x0000  00 52 59 5a 45 4e 42 45 41 53 54 00 00 00 00 00  .RYZENBEAST.....
0x0010  00 00 00 00 00 00 00 00 90 63 fd 79 c1 02 00 00  .........c.y....
...
```

Reference

[1] Kawamura, S.: *A Recommendation for IPv6 Address Text Representation*. Datatracker, [online], srpen 2010, [cit. 24. dubna 2022].
Dostupné z: `https://datatracker.ietf.org/doc/html/rfc5952`

[2] Plummer, D. C.: *An Ethernet Address Resolution Protocol – or – Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware*. Datatracker, [online], listopad 1982, [cit. 24. dubna 2022].
Dostupné z: `https://datatracker.ietf.org/doc/html/rfc826`