FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

IPK – 2. projekt
Packet sniffer

28. 4. 2020 Mirka Kolaříková(xkolar76)

Contents

1 Zadání

Cílem druhé úlohy bylo vytvořit program v jazyku C/C++/C#, který na určitém síťovém rozhraní zachytává a filtruje pakety.

2 Implementace programu

Program je implementovaný v jazyce C.

2.1 Vstupní argumenty

Nejdříve proběhne kontrola vstupních parametrů, a to s využitím funkce na parsování krátkých i dlouhých argumentů `getopt_long()`. Podporované argumenty programu jsou: -i INTERFACE, -p PORT, -t/--tcp, -u/--udp, -n NUM. Všechny parametry je možné zapsat za přepínač i bez mezery (např. -n 12/-n12). V případě neznámého vstupního argumentu program vypíše odpovídají chybovou hlášku a ukončí se s chybovým kódem `10`.

2.1.1 -i INTERFACE

Očekává se uvedení rozhraní, na kterém se bude poslouchat. V případě chybějícího parametru INTERFACE či úplně chybějícího přepínače -i, je s využitím funkce `pcap_findalldevs()` vypsán seznam dostupných rozhraní na systému. Jinak je rozhraní otevřeno pomocí funkce `pcap_open_live()` a začíná sniffování paketů ve smyčce pomocí funkce `pcap_loop()`.

2.1.2 filtry -p PORT, -t/--tcp, -u/--udp

Kombinací těchto tří přepínačů získáme filtr, který poté využijeme při sniffování paketů. Filtr vznikne konkatenací stringů. Například při zadaní přepínačů –tcp -u -p 13 vzniká filtr `(tcp or udp) and port 13`. Takto vzniklý filtr se poté aplikuje pomocí funkcí `pcap_lookupnet()`, `pcap_compile()` a `pcap_setfilter()`.

2.1.3 -n NUM

V případě chybného formátu či negativního čísla NUM je vypsána chybová hláška a program je ukončen s chybovým kódem `11`. Číslo NUM (defaultně 1) je poté použito jako druhý argument funkce `pcap_loop()` pro určení počtu sniffovaných paketů. Pokud NUM je 0, sniffování packetů jede do nekonečna a program musí zastavit uživatel.

2.2 Zpracování packetu

Zpracování jednotlivých packetů probíhá ve funkci `process_packet()`. Na základě protokolu packetu jsou volaný funkce `print_tcp_packet()` a `print_udp_packet()`.

2.2.1 Hlavička

Každá z těchto funkcí nejříve vypíše hlavičku v podobě `čas zdrojová IP : port > cílová IP : port`. K získání času jsou využity funkce `time()`, `localtime()` a formátování pomocí funke `strftime()`. IP adresa se překládá pomocí funkce `getnameinfo()`. Port je získán funkcí `ntohs()`.

2.2.2 Data

Po hlavičce si vypisují data ve funkci `PrintData()`. Pro počítaní vypsaných bajtů je implementován interní čítač, který se po každém vypsaném bajtu inkrementuje. Nejdříve je vypsána IP hlavička, poté TCP/UDP hlavička nakonec samotná data packetu. Hlavička a samotná data jsou oddělena prázdným řádkem. Jednotlivé pakety jsou taktéž odděleny prázným řádkem.

3 Testování

Na testování byl použit referenční stroj k předmětu IPK. Jako testovací rozhraní bylo převážně využiváno enp0s3. Na přenos packetů jsem využiívala používání internetového prohlížeče. Na porovnání výstupu jsem využila nástroj WireShark.

4 Použité zdroje

Při tvorbě projektu byly využity a dle potřeb upraveny kódy z těchto zdrojů:

`https://www.binarytides.com/packet-sniffer-code-c-libpcap-linux-sockets/`
`https://www.tcpdump.org/pcap.html`
`http://embeddedguruji.blogspot.com/2014/01/pcapfindalldevs-example.html`
`https://stackoverflow.com/questions/28566424/linux-networking-gethostbyaddr`