

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

IPK – 2. projekt
Packet sniffer

Obsah

1	Postup riešenia	2
1.1	Spracovanie argumentov programu	2
1.2	Vypis všetkých dostupných rozhraní	2
1.3	Ustanovenie spojenia s rozhraním	2
1.4	Filtrovanie	2
1.5	Zachytávanie packetov	2
1.6	IPv4 packet	2
1.7	ARP packet	2
1.8	IPv6 packet	2
1.9	Výpis dat v ASCII a hexadecimale	3
2	Použité knižnice	3
3	Testovanie	3
3.1	Použité ďalšie programy	3
4	Literatura	4

1 Postup riešenia

1.1 Spracovanie argumentov programu

Programové argumenty boli spracované pomocou if-else podmienok v jazyku C. Postupne sa prechádzali programové argumenty a získané data sa zapisovali do datovej štruktúry `argFields`, ktorá obsahovala atribúty `char *interface`, `int port`, `int n`, `bool tcp`, `bool udp`, `bool arp` a `bool icmp`.

1.2 Vypis všetkých dostupných rozhraní

Pri výpise všetkých dostupných rozhraní sme použili knižnicu `pcap.h` a jej vstavanú funkciu `pcap_findalldevs`, ktorá nám poskytla list všetkých dostupných rozhraní.

1.3 Ustanovenie spojenia s rozhraním

Pri ustanovaní a otvorení spojenia na danom rozhraní sme použili funkcie `pcap_open_live`, `pcap_loopnet` a `pcap_datalink` viz [1].

1.4 Filtrovanie

Pri filtrovaní sme si vo funkcií `FilterStringCreating` vytvorili `filterString`, ktorý sme vyskladali na základe programových argumentov, ktoré sa rozparsované nachádzali v štruktúre `argFields`. Pri filtrovaní sme použili funkcie `pcap_compile` a `pcap_setfilter` viz [2].

1.5 Zachytávanie packetov

Vo for-cykle sme začali zachytávať požadovaný počet packetov a zachytávali sme ich pomocou funkcie `pcap_next`. Zo získaného packetu sme zistili a vypísali timestamp podľa požadovaného formátu. Tento packet sme pretypovali na ethernetovú hlavičku do datovej štruktúry `ether_header` a následne si z tejto ethernetovej hlavičky zistili source a destination MAC adresy, ktoré sme následne vypísali v požadovanom formáte. Na základe `ether_type` v ethernetovej hlavičke sme zistili, či sa jedná o IPv4, IPv6 alebo ARP packet viz [4].

1.6 IPv4 packet

Pretypovanie na IPv4 packet sme robili z packetu získaného z funkcie `pcap_next` posunutého o 14 bytov, ktoré zaberala ethernetová hlavička. Pretypovali sme packet do štruktúry `ip`, ktorá reprezentovala IPv4 hlavičku. Z tejto datovej štruktúry sme následne mohli zistiť source, destination IP adresu a taktiež protokol (`icmp`, `tcp` a `udp`). Na základe získaného protokolu sme si pretypovali IPv4 packet na packet UDP [8], TCP [7] alebo ICMP [5] a zistili z nich ak to bolo možné source a destination port.

1.7 ARP packet

Packet ktorý sme získali z funkcie `pcap_next` sme pretypovali do štruktúry `ether_arp`, ktorá reprezentovala ARP hlavičku. Keďže ARP hlavička [3] neobsahuje TCP, UDP ani ICMP protokol a teda nemá source a destination porty, vypísali sme z nej len source a destination IP adresy.

1.8 IPv6 packet

Pretypovanie IPv6 [6] hlavičky a získavanie dat z nej bolo veľmi podobné získavaniu dát v IPv4 hlavičky aj keď tieto hlavičky sú rozdielne. IPv6 hlavička má pevnú veľkosť 40 bytov na rozdiel od IPv4 hlavičky, ktorej veľkosť nie je fixná a teda pretypovanie na UDP, TCP alebo ICMP prebeha s iným (dynamickým) offsetom.

1.9 Výpis dat v ASCII a hexadecimale

Data sme vypisovali vo for-cykle podľa formátu, ktorý bol uvedený v zadaní a vo Wiresharku.

2 Použité knižnice

- <stdio.h>
- <pcap.h>
- <arpa/inet.h>
- <netinet/if_ether.h>
- <netinet/ether.h>
- <netinet/ip6.h>
- <netinet/tcp.h>
- <netinet/ip_icmp.h>
- <netinet/udp.h>
- <stdlib.h>
- <stdbool.h>
- <string.h>
- <time.h>

3 Testovanie

Testovanie prebiehalo pomocou softwaru Wireshark, v ktorom sme sledovali aktuálny packetový traffic, filtrovali si packety podľa požadovaných protokolov a zisťovali či nám sedia všetky požadované dáta vrátane IP a MAC adres.

3.1 Použité ďalšie programy

Pre vytvorenie konkrétneho ICMP packetu sme použili nástroj `ping`, ktorý na zvolenú IP adresu / doménu pošlal ICMP dotaz a čaká na odpoveď.

Pre vytvorenie APR packetu sme použili nástroj `arping`, ktorý na zvolenú IP adresu / doménu pošlal ARP dotaz.

Pre vytvorenie UDP a TCP packetu sme otvorili prehliadač, ktorý automaticky začal komunikovať so sieťou a posielal UDP a TCP packety.

4 Literatura

Literatúra

- [1] ARORA, H.: How to Perform Packet Sniffing Using Libpcap with C Example Code. [online], [vid. 2022-04-16]. Dostupné z: <https://www.thegeekstuff.com/2012/10/packet-sniffing-using-libpcap/>
- [2] Group, T. T.: Programming with PCAP. [online], [vid. 2022-04-16]. Dostupné z: <https://www.tcpdump.org/pcap.html>
- [3] Wikipedia: Address Resolution Protocol. [online], [vid. 2022-04-16]. Dostupné z: https://en.wikipedia.org/wiki/Address_Resolution_Protocol
- [4] Wikipedia: Ethernet frame. [online], [vid. 2022-04-16]. Dostupné z: https://en.wikipedia.org/wiki/Ethernet_frame
- [5] Wikipedia: Internet Control Message Protocol. [online], [vid. 2022-04-16]. Dostupné z: https://en.wikipedia.org/wiki/Internet_Control_Message_Protocol
- [6] Wikipedia: IPv6 packet. [online], [vid. 2022-04-16]. Dostupné z: https://en.wikipedia.org/wiki/IPv6_packet
- [7] Wikipedia: Transmission Control Protocol. [online], [vid. 2022-04-16]. Dostupné z: https://en.wikipedia.org/wiki/Transmission_Control_Protocol
- [8] Wikipedia: User Datagram Protocol. [online], [vid. 2022-04-16]. Dostupné z: https://en.wikipedia.org/wiki/User_Datagram_Protocol