Počítačové komunikace a sítě

2021/2022

Obsah obrázku text

Popis byl vytvořen automaticky

Protokol k projektu

Varianta ZETA: Sniffer paketů

Václav Valenta (xvalen29) Brno, 24.04. 2022

# Obsah

Obsah

[2. Základní funkcionalita 3](#_Toc101732560)

[2.1. Překlad a spuštění 3](#_Toc101732561)

[2.2. Přepínače 3](#_Toc101732562)

[3. Implementace 3](#_Toc101732563)

[3.1. Zpracování argumentů 3](#_Toc101732564)

[3.2. Připojení rozhraní 4](#_Toc101732565)

[3.3. Filtrace 4](#_Toc101732566)

[3.4. Zpracování paketů – čas 4](#_Toc101732567)

[3.5. Zpracování paketů – typ 5](#_Toc101732568)

[3.6. Zpracování paketů – výpis dat 5](#_Toc101732569)

[4. Testování 6](#_Toc101732570)

[5. Zdroje 8](#_Toc101732571)

[5.1. Výpis aktivních rozhraní 8](#_Toc101732572)

[5.2. Timestamp 8](#_Toc101732573)

[5.3. Připojení rohzraní 8](#_Toc101732574)

[5.4. Filtrace 8](#_Toc101732575)

[5.5. MAC adresa 9](#_Toc101732576)

[5.6. Výpis dat 9](#_Toc101732577)

[5.7. Zpracování paketů 9](#_Toc101732578)

[5.8. Zpracování argumentů 10](#_Toc101732579)

# Základní funkcionalita

## Překlad a spuštění

Překlad projektu se provádí příkazem make, který vytvoří spustitelný soubor s názvem *ipk-sniffer*. Program se spouští následovně:

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

## Přepínače

Pomocí přepínače *-i* nebo *--interface* definujeme rozhraní. V případě, že rozhraní není zadáno, program vypíše seznam všech rozhraní.

Přepínač *-p* nebo *--port* slouží ke specifikaci určitého portu, na kterým se má komunikace zachytávat. Pokud port není specifikován, program bude zachytávat komunikaci na všech portech.

Další přepínače určují typ paketů, které se mají vypisovat *(--tcp | -t, --udp | -u, --arp, --icmp*). Pokud tyto přepínače nejsou uvedeny, nebo jejich kombinace je neplatná, uvažují se všechny zmíněné typy. Zadaný port se nemění.

Poslední přepínač -n určuje kolik paketů program vypíše.

# Implementace

## Zpracování argumentů

Program po spuštění začne zpacovávat zadané vstupní argumenty pomocí funkce *getopt\_long()*. Pokud zadané argumenty neodpovídají formátu popsanémum výše, program vypíše chybové hlášení a ukončí činnost.

Pro lepší přehlednost jsou všechny chybová hlášení v kodě zadefinované v enum struktuře a volané funkcí *print\_error(error\_t err)*:

void print\_error(error\_t err) {

switch (err) {

case PCAP\_DEVS\_ERR:

fprintf(stderr, "pcap\_findalldevs error");

break;

...

typedef enum error\_enum {

PCAP\_DEVS\_ERR,

INVALID\_INTERFACE\_ERR,

...

} error\_t;

## Připojení rozhraní

V případě, že rozhraní není definované, program vypíše seznam všech aktivních rozhraní a ukončí činnost:

Obsah obrázku text

Popis byl vytvořen automaticky

V opačném případě se pokusí na rozhraní připojit – v případě úspěchu program pokračuje dále.

## Filtrace

Filtry jsou parsovány ze vstupu – konkrétně se jedná o bool hodnoty *tcp, udp, icmp* a *arp*.Pokud se jedná o validní kombinaci, jsou filtry logicky spojeny a zapsány do řetězce filter\_string. V opačném případě je tento řetězec prázdný. Nakonec je připojen i filtr, který udává číslo zadaného portu.

// Add filter for port

if (port != -1) {

// Check invalid filter combination

if (arp || icmp) {

strcpy(filter\_string, "");

sprintf(port\_filter, "port %d", port);

} else {

sprintf(port\_filter, " and port %d", port);

}

}

strcat(filter\_string, port\_filter);

## Zpracování paketů – čas

Po úspěšném vytvoření a aplikování filtru je zavolána funkce *pcap\_loop*, která volá funkci *sniffing*. Tato funkce ihned vypíše čas ve formátu *yyyy-MM-dd'T'HH:mm:ss.SSSZ*. Pro dosažení požadovaného výsledku byly použity následující struktury:

time\_t rawtime;

struct tm \*info;

struct timeval tv;

Milisekundová část je vypočítána následujícím způsobem:

int ms = round(tv.tv\_usec / 1000);

if (ms >= 1000) {

ms -= 1000;

tv.tv\_sec++;

}

## Zpracování paketů – typ

Kromě času se také ihned vypíše zdrojová, cílová MAC adresa a také délka rámce.

Dalším úkolem programu je zjištění typu paketu. K tomu využívá funkci *ntohs*.

void sniffing(u\_char \*user, const struct pcap\_pkthdr \*header, const u\_char \*packet) {

struct ether\_header \*eth\_header;

eth\_header = (struct ether\_header \*)packet;

...

if (ntohs(eth\_header->ether\_type) == ETHERTYPE\_ARP) {

...

} else if (ntohs(eth\_header->ether\_type) == ETHERTYPE\_IP) {

...

}

...

}

Následné zpracování se odvíjí právě podle tohoto typu. V případě typu ETHERTYPE\_IP je třeba ještě zjistit ještě příslušný typ IPv4 a obdobně v případě ETHERTYPE\_IPV6.

Na zjištění portů se používá funkce *ntohs,* na IPv4 adresy funkce *inet\_ntoa*.

## Zpracování paketů – výpis dat

Poslední věc, kterou programu ještě zbývá udělat, je vypsat data – a to jak v hexadecimální podobě, tak i v ASCII podobě. Program zavolá funkci *print\_data*.

Funkce počítá aktuální offset a podle velikosti zbývajících dat rozhoduje, zda bude pokračovat dále, nebo budou aktuálně zpracovávaná data poslední řádkou výpisu.

// Print whole lines, until the size is smaller than one line

while (size\_left > 16) {

line\_length = 16 % size\_left;

print\_hex\_ascii(char\_data, line\_length, offset);

size\_left -= line\_length;

char\_data += line\_length;

offset += 16;

}

// Print the last line

print\_hex\_ascii(char\_data, size\_left, offset);

Funkce na každý zpracovaný řádek volá funkci *print\_hex\_ascii*. Která data vypíše v obou formátech s potřebnými mezerami.

Po zpracování posledního paketu funkce zavolá funcki *pcap\_close* a ukončí běh programu.

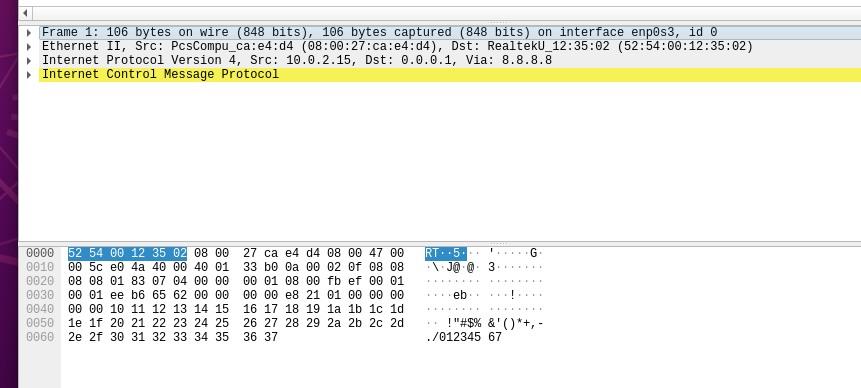
# Testování

K testování byl použit virtuální stroj a program Wireshark

## ICMP

Obsah obrázku stůl

Popis byl vytvořen automaticky



## UDP

Obsah obrázku stůl

Popis byl vytvořen automaticky

Obsah obrázku stůl

Popis byl vytvořen automaticky

## ARP

Obsah obrázku text, elektronika, klávesnice, zavřít

Popis byl vytvořen automaticky

Obsah obrázku text

Popis byl vytvořen automaticky

## TCP

Obsah obrázku text, interiér, snímek obrazovky

Popis byl vytvořen automaticky

Obsah obrázku stůl

Popis byl vytvořen automaticky

# Zdroje

## Výpis aktivních rozhraní

Tcpdump [online] <https://www.tcpdump.org/manpages/pcap\_findalldevs.3pcap.html> [viewed: 23.04.2022]

Linux Documentation [online] <https://linux.die.net/man/3/pcap\_freealldevs> [viewed: 23.04.2022]

## Timestamp

WinPcap documentation [online] <https://www.winpcap.org/docs/docs\_40\_2/html/group\_\_wpcap\_\_tut4.html> [viewed: 23.04.2022]

Stack Overflow [online] <https://stackoverflow.com/questions/3673226/how-to-print-time-in-format-2009-08-10-181754-811> [viewed: 23.04.2022]

Sumo Logic Doc Hub [online] <https://help.sumologic.com/03Send-Data/Sources/04Reference-Information-for-Sources/Timestamps%2C-Time-Zones%2C-Time-Ranges%2C-and-Date-Formats> [viewed: 23.04.2022]

The Open Group Library [online] <https://pubs.opengroup.org/onlinepubs/9699919799/functions/strftime.html> [viewed: 23.04.2022]

C Standard Library Reference Tutorial [online] <https://www.tutorialspoint.com/c\_standard\_library/c\_function\_sprintf.htm> [viewed: 23.04.2022]

## Připojení rohzraní

Tcpdump [online] <https://www.tcpdump.org/manpages/pcap\_open\_live.3pcap.html> [viewed: 23.04.2022]

## Filtrace

Tcpdump [online] <https://www.tcpdump.org/manpages/pcap\_setfilter.3pcap.html> [viewed: 23.04.2022]

Tcpdump [online] <https://www.tcpdump.org/manpages/pcap\_compile.3pcap.html> [viewed: 23.04.2022]

Wireshark documentation [online] <https://www.wireshark.org/docs/wsug\_html\_chunked/ChCapCaptureFilterSection.html> [viewed: 23.04.2022]

Linux Documentation [online] <https://linux.die.net/man/3/pcap\_freecode> [viewed: 23.04.2022]

## MAC adresa

Capturing Our First Packet [online] <http://yuba.stanford.edu/~casado/pcap/section2.html> [viewed: 24.04.2022]

## Výpis dat

Tcpdump [online] <https://www.tcpdump.org/other/sniffex.c> [viewed: 24.04.2022]

How to code a Packet Sniffer in C with Libpcap on Linux [online] <https://www.binarytides.com/packet-sniffer-code-c-libpcap-linux-sockets> [viewed: 24.04.2022]

## Zpracování paketů

Using libpcap in C [online] <https://www.devdungeon.com/content/using-libpcap-c#pcap-loop> [viewed: 24.04.2022]

WinPcap documentation [online] <https://www.winpcap.org/docs/docs\_412/html/structpcap\_\_pkthdr.html> [viewed: 24.04.2022]

Wikipedia [online] <https://en.wikipedia.org/wiki/Ethernet\_frame> [viewed: 24.04.2022]

netinet/if\_ether.h documentation [online] <https://unix.superglobalmegacorp.com/Net2/newsrc/netinet/if\_ether.h.html> [viewed: 24.04.2022]

Tcpdump [online] <https://samy.pl/packet/MISC/tcpdump-3.7.1/ethertype.h> [viewed: 24.04.2022]

netinet/in.h documentation [online] <https://pubs.opengroup.org/onlinepubs/009695399/basedefs/netinet/in.h.html> [viewed: 24.04.2022]

ether\_arp Struct Reference [online] <http://www.ethernut.de/api/structether\_\_arp.html> [viewed: 24.04.2022]

Jeremiah Mahler - socket-examples [online] <https://github.com/jmahler/socket-examples/blob/master/packets/packets.c> [viewed: 24.04.2022]

Assigned Internet Protocol Numbers [online] <https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml> [viewed: 24.04.2022]

Sniffer example of TCP/IP packet capture using libpcap [online] <https://www.tcpdump.org/other/sniffex.c> [viewed: 24.04.2022]

Understanding the IPv6 Header [online] <https://www.microsoftpressstore.com/articles/article.aspx?p=2225063&seqNum=3> [viewed: 24.04.2022]

## Zpracování argumentů

Linux Documentation [online] <https://linux.die.net/man/3/getopt\_long> [viewed: 23.04.2022]

Stack Overflow [online] <https://stackoverflow.com/questions/19604413/getopt-optional-arguments> [viewed: 23.04.2022]

Stack Overflow [online] <https://stackoverflow.com/questions/7489093/getopt-long-proper-way-to-use-it> [viewed: 23.04.2022]

Stack Overflow [online] <https://stackoverflow.com/questions/9642732/parsing-command-line-arguments-in-c> [viewed: 23.04.2022]

getopt() function in C to parse command line arguments [online] <https://www.tutorialspoint.com/getopt-function-in-c-to-parse-command-line-arguments> [viewed: 23.04.2022]

Robbins, A. (2004). Linux programming by example (1st ed.). Prentice Hall. ISBN 9780131429642