VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta informačních technologií

POČÍTAČOVÉ KOMUNIKACE A SÍTĚ

2021/2022

Projekt 2

Varianta ZETA – sniffer packetů

Dominik Pop (xpopdo00) Brno, 24. dubna 2022

Obsah

Úvod

Cílem projektu bylo navrhnout a implementovat síťový analyzátor, konkrétně zachytávač packetů. Jeho úkolem je zachytávat packety na určitém rozhraní, na základě daného filtru, a vypsat o nich požadované informace.

Základní popis implementace

Analyzátor byl implementován v jazyce C za použití knihovny libpcap. K implementaci byl vytvořen pouze jeden modul obsahující strukturu pro vstupní argumenty. Funkce z knihovny libpcap jsou volány ve funkci main. O zpracování packetů se pak stará implementovaná funkce sniffer_callback, která volá další pomocné funkce, pro zpracování/vypsání informací o packetech.

Hlavní funkce

I. main

Funkce ***main*** začíná zpracováním vstupních argumentů a jejich uložením do implementované struktury ***t_Args***. O zpracování argumentů se stará funkce ***parse_arguments***, která kontroluje jejich validitu a ukládá je. Funkce ***parse_arguments*** se také stará o vyvolání funkcí ***help_function*** (výpis nápovědy) a ***print_interfaces*** (výpis všech rozhraní). Následně se ***main*** stará o získání síťové masky a IP adresy zvoleného rozhraní pomocí funkce ***pcap_lookupnet***. Poté je rozhraní otevřeno pro zachytávání packetů díky funkci ***pcap_open_live***. Dále je ve funkci ***main*** volaná implementovaná funkce ***create_filter***, která vytvoří filtr pro zachytávání packetů na základě vstupních argumentů. Tento filtr je pak zapotřebí přeložit a aplikovat na naše rozhraní pomocí funkcí ***pcap_compile*** a ***pcap_setfilter***. Když je vše správně nastaveno zavolá main funkci ***pcap_loop***, pomocí které čeká na příchozí packety, jejíž počet je specifikovaný vstupním argumentem -n (implicitně 1). Argumentem ***pcap_loop*** je funkce ***sniffer_callback*** starající se o zpracování packetů.

II. sniffer_callback

Tato funkce má na starosti získání dat z příchozích packetů a jejich výpis na standartní výstup. Začíná zpracováním ethernetové hlavičky za pomocí struktury ***ether_header***. Z této hlavičky získá zdrojovou a cílovou MAC adresu za pomocí funkce ***ether_ntoa_override.*** Tato funkce byla přepsána pro účely projektu, aby vypisovala tzv. vedoucí nuly. Následně je z hlavičky získána délka packetu v bytech a timestamp ve formátu RFC3339. K dodržení správného formátu byla použita funkce ***strftime***. Po zpracování ethernetové hlavičky přechází ***sniffer_callback*** na zpracování internetové hlavičky. Před tím je za potřebí rozhodnout, jestli se jedná o IPv4 nebo IPv6. Pro hlavičku typu IPv4 je použita struktura ***ip*** a pro získání zdrojové i cílové IP adresy funkce ***inet_ntoa***. Pro IPv6 je použita struktura ***ip6_hdr*** a funkce ***inet_ntop***. Po získání informací z internetové hlavičky probíhá první výpis na standartní výstup, o který se stará implementovaná funkce ***print_basic_info***, která vypíše timestamp, délku v Bytech, MAC adresy a o jaký protokol se jedná. Na základě protokolu packetu se pak vypisují další informace, jako IP adresy, nebo čísla portů. O jejich výpis se starají funkce individuální pro jednotlivé protokoly ***process_tcp/udp*** (v případě rozšíření by byli implementovány funkce ***process_arp/icmp***). Holá nezpracovaná data jsou pak na výstup vypsány pomocí funkcí ***print_payload*** a ***print_hex_ascii***. Tyto funkce byly vypůjčeny z odkazu odcitovaného ve zdrojích a jejich autorem je Tim Carstens.

Spuštění

Program byl implementován a testován na Linuxovém prostředí, je tudíž s tímto prostředím kompatibilní a kompatibilita s jinými prostředími není zaručena. K přeložení programu je zapotřebí překladač gcc a nástroj GNU Make.

Přeložení programu pomocí nástroje Make:

```
$ make
```

Spuštění programu pak probíhá následovně:

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

Pokud spuštění předchozím příkladem není možné, je zapotřebí přidělit příkazu rootovská práva pomocí **sudo** (sudo ./ipk-sniffer …).

Významy argumentů:

- [-i rozhraní | --interface rozhraní] -> specifikace rozhraní, na kterém se budou zachytávat packety
- {-p --port} -> specifikace čísla portu, se kterým budou packety zachytávány
- {[--tcp|-t] [--udp|-u] [--arp] [--icmp]} -> specifikace protokolů packetů, které mají být zachytávány
- {-n num} -> počet udávající kolik packetů má být zachyceno

Program má možnost zobrazení všech rozhraní pomocí příkazu:

```
$ ./ipk-sniffer {-i}
```

Program má možnost zobrazení nápovědy pomocí příkazu:

```
$ ./ipk-sniffer [-h | --help]
```

Příklady spuštění a výstupů:

1. $ ./ipk-sniffer -i enp0s3 –arp –icmp & arp -D -e <IP_adresa> enp0s3

```
student@student-vm:~/Desktop/IPK$ sudo ./ipk-sniffer -i enp0s3 --arp --icmp
***ARP***
timestamp: 2022-04-15T19:11:42.56688+02:00
src MAC: 08:00:27:ca:e4:d4
dst MAC: 52:54:00:12:35:02
frame length: 42 bytes

0x0000   52 54 00 12 35 02 08 00 27 ca e4 d4 08 06 00 01    RT..5...'.......
0x0010   08 00 06 04 00 01 08 00 27 ca e4 d4 0a 00 02 0f    ........'.......
0x0020   00 00 00 00 00 00 0a 00 02 02                      ..........
```

Obrázek 1: Příklad výstupu č.1

2. $ ./ipk-sniffer -i enp0s3 –arp –icmp & ping <IP_adresa>

```
student@student-vm:~/Desktop/IPK$ sudo ./ipk-sniffer -i enp0s3 --arp --icmp
***ICMP***
timestamp: 2022-04-15T19:10:40.66205+02:00
src MAC: 08:00:27:ca:e4:d4
dst MAC: 52:54:00:12:35:02
frame length: 98 bytes
src IP: 10.0.2.15
dst IP: 10.0.2.2

0x0000   52 54 00 12 35 02 08 00 27 ca e4 d4 08 00 45 00    RT..5...'.....E.
0x0010   00 54 9d 73 40 00 40 01 85 25 0a 00 02 0f 0a 00    .T.s@.@..%......
0x0020   02 02 08 00 ad 07 00 01 00 01 10 a7 59 62 00 00    ............Yb..
0x0030   00 00 18 1a 0a 00 00 00 00 00 10 11 12 13 14 15    ................
0x0040   16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25    .......... !"#$%
0x0050   26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35    &'()*+,-./012345
0x0060   36 37                                             67
```

Obrázek 2: Příklad výstupu č.2

3. $ ./ipk-sniffer -i enp0s3 –p 80 & otevření prohlížeče

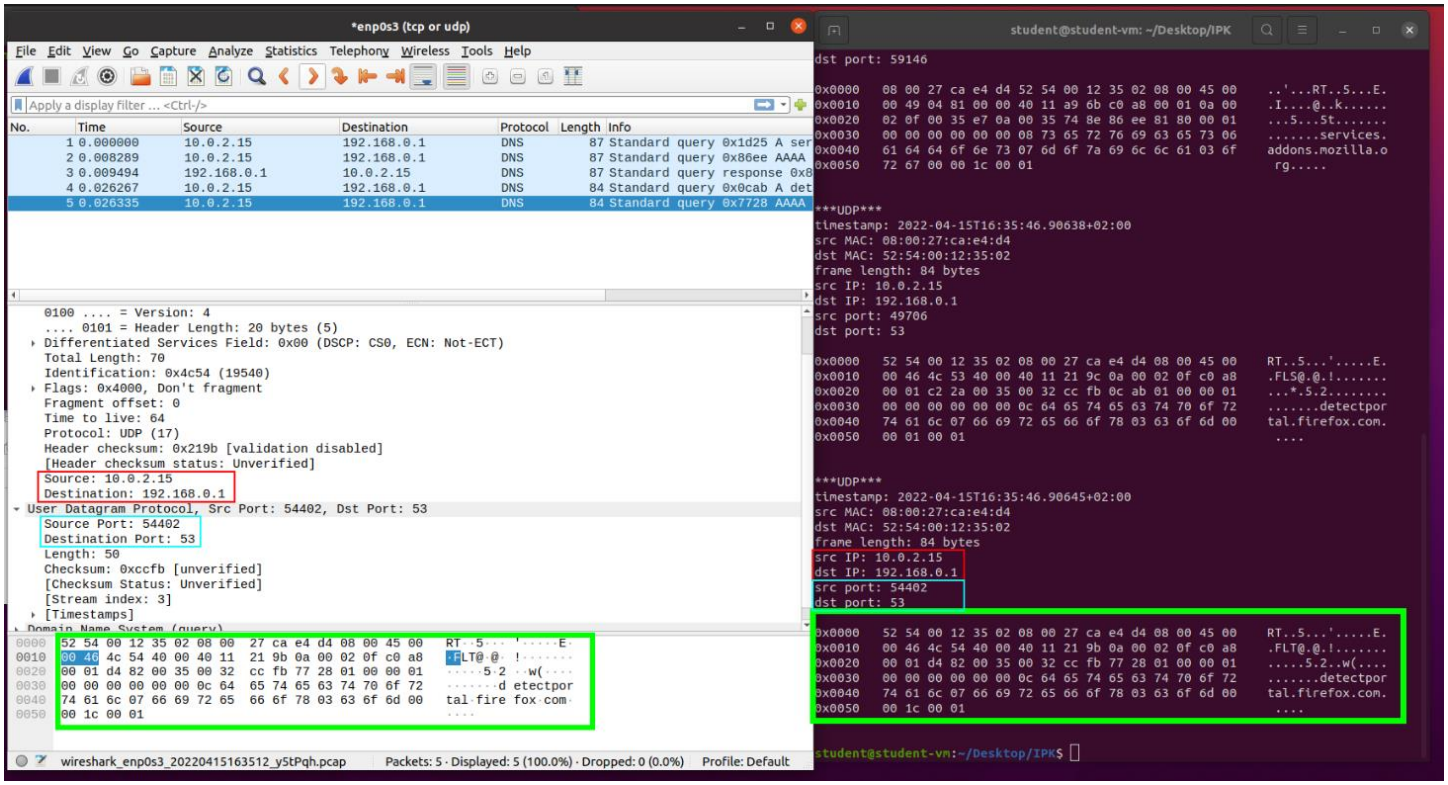
```
student@student-vm:~/Desktop/IPK$ sudo ./ipk-sniffer -i enp0s3 -p 80
***UDP***
timestamp: 2022-04-15T19:09:03.95974+02:00
src MAC: 08:00:27:ca:e4:d4
dst MAC: 52:54:00:12:35:02
frame length: 87 bytes
src IP: 10.0.2.15
dst IP: 192.168.0.1
src port: 50211
dst port: 53

0x0000   52 54 00 12 35 02 08 00 27 ca e4 d4 08 00 45 00    RT..5...'.....E.
0x0010   00 49 ec b2 40 00 40 11 81 39 0a 00 02 0f c0 a8    .I..@.@..9......
0x0020   00 01 c4 23 00 35 00 35 cc fe 56 73 01 00 00 01    ...#.5.5..Vs....
0x0030   00 00 00 00 00 00 08 73 65 72 76 69 63 65 73 06    .......services.
0x0040   61 64 64 6f 6e 73 07 6d 6f 7a 69 6c 6c 61 03 6f    addons.mozilla.o
0x0050   72 67 00 00 1c 00 01                               rg.....
```

Obrázek 3: Příklad výstupu č.3

Testování

Projekt byl testován na referenčním Linux virtuálním prostředí. K ověřování výsledků byl použit software Wireshark. K vyvolání packetů bylo použito několik metod (ping, arping, prohlížeč, curl, …). Program Wireshark má možnosti nastavení filtrů podobné vstupním argumentům implementovaného snifferu. Testování tedy probíhalo, tak že byl nastaven filtr vyhledávání (nikoliv filtr zobrazení) ve Wiresharku, podle toho byly následovně zvoleny vstupní argumenty snifferu, a oba byly spuštěny. Data nalezených packetů byly pak mezi sebou porovnávány (viz. příklad).

- $ ./ipk-sniffer -i enp0s3 -t -u -n 5 & otevření prohlížeče



Obrázek 4: Výstup testovacího příkladu

Zdroje

- *TCPDUMP* [online]. The Tcpdump Group, 2010 [cit. 2022-04-18]. Dostupné z: https://www.tcpdump.org/
- *Sniffex* [online]. 2005-07-05 [cit. 2022-04-18]. Dostupné z: https://www.tcpdump.org/other/sniffex.c
- Networking Library Functions. *ORACLE* [online]. [cit. 2022-04-18]. Dostupné z: https://docs.oracle.com/cd/E36784_01/html/E36875/index.html
- Data Structures. *Linux Kernel Documentation* [online]. 2013 [cit. 2022-04-18]. Dostupné z: https://docs.huihoo.com/doxygen/linux/kernel/3.7/annotated.html
- Protocol Numbers. *Iana* [online]. [cit. 2022-04-18]. Dostupné z: https://www.iana.org/assignments/protocol-numbers/protocol-numbers.xhtml
- Section 3: library functions - Linux man pages. *Iana* [online]. 1996 [cit. 2022-04-18]. Dostupné z: https://linux.die.net/man/3/
- *NetBSD Manual Pages* [online]. [cit. 2022-04-18]. Dostupné z: https://man.netbsd.org/
- Strftime. *Cplusplus* [online]. [cit. 2022-04-18]. Dostupné z: https://www.cplusplus.com/reference/ctime/strftime/
- MAC address:pad missing left Zeros. *Cplusplus* [online]. [cit. 2022-04-18]. Dostupné z: https://stackoverflow.com/questions/4736718/mac-addresspad-missing-left-zeros
- I'm trying to build an RFC3339 timestamp in C. How do I get the timezone offset?. *Stack Overflow* [online]. [cit. 2022-04-18]. Dostupné z: https://stackoverflow.com/questions/4736718/mac-addresspad-missing-left-zeros