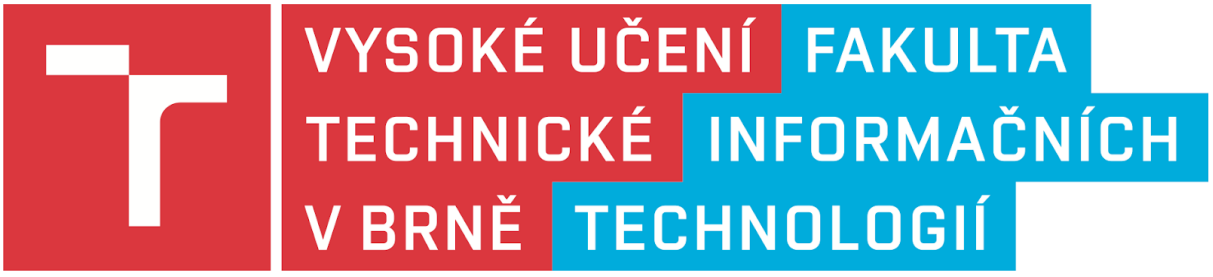

Projektová dokumentace

Implementace 2. projektu do IPK

Varianta ZETA: Sniffer paketů

Vojtěch Vašíček
xvasic31

3. května 2020

Obsah

Úvod

Cílem projektu bylo vytvořit sniffer paketů, který na základě vstupních argumentů vyfiltruje určitý počet paketů, o kterých pak vypíše informace spolu s daty jenž tyto pakety nesou.

Implementace

Projekt se skládá z několika částí které dohromady provádějí zadanou funkci. Tyto funkce představím v této kapitole.

Parsování argumentů

Na začátku projektu deklarujeme proměnné a struktury které budou využity v pozdějších částech projektu. Dále parsujeme argumenty zadané uživatelem.

Tyto argumenty zastávají určitou funkci a kromě argumentu -help mohou být zadané v jakémkoliv pořadí. Argument *-help* musí být zadán první, protože se při jeho volání předpokládá že uživatel nezná další možné parametry. Program vypíše základní funkci programu a pošle uživatele na více informovaný a lépe strukturovaný README.

Ostatní argumenty jsou využity pro specifikaci sniffovaných paketů. Tyto informace jsou později využity při kompilování filtru.

Při parsování argumentů je také kontrolováno zda byl argument již jednou v tomto volání zadán. Pokud ano, program, výpisem chybové zprávy, upozorní uživatele na zadání redundantního argumentu. Potom se program vypne s chybovým kódem 1.

U některých argumentů se očekává návazný argument, například *-i interface.* V tomto případě se také kontroluje typová správnost tohoto návazného argumentu. Pokud není argument -i zadán, vypíší se všechna dostupná rozhraní.

Argumenty se pak dále kontrolují zejména vůči sobě. Pokud je například přítomen jak argument *--tcp* tak *--udp*, zachytává můj program všechny pakety, limitován pouze dalšími argumenty. Pro toto řešení jsem se rozhodl proto, že nelze hledat pouze tcp pakety a pouze udp pakety zároveň.

Sestavení filtru

Filtr se kompiluje a užívá zejména použitím knihovny pcap.h. Nejprve se zkontroluje validita zadaného rozhraní pomocí funkce *pcap_lookupdev* která v případě chybně zadaného rozhraní vypíše chybovou zprávu a ukončí program.

Dále se pomocí *pcap_open_live* otevře zadané rozhraní pro čekání na příchozí pakety.

Poté se pomocí funkcí *pcap_compile* a *pcap_setfilter* filtr podle zadaných parametrů zkompiluje a následně nastaví filtr na námi otevřené rozhraní.

Sniffování paketu

Samotné sniffování se provádí pomocí funkce *pcap_loop* opět z knihovny pcap.h. Tato funkce čeká na procházející pakety a v případě zachycení takového paketu zavolá zadanou funkci *process_packet*.

Tato funkce nejprve zjistí současný strojový čas s přesností na milisekundy pomocí funkcí *gettimeofday* a *localtime*.

Dále funkce pomocí funkcí *in_addr* a *iphdr* zjistí výchozí ip adresu a port. Ip adresa se z důvodu častého zacyklení nepřekládá na FQDN. K tomuto zacyklení dochází kvůli zachytávání samotného paketu s ip adresou, která měla být přeložena. Tato adresa je pak znovu překládána celý cyklus se opakuje.

Poté funkce vytiskne první část hlavičky, která obsahuje právě čas a výchozí ip adresu a port. Funkce pak zjistí cílovou ip adresu a port a tyto informace také vypíše.

Dále dochází k výpisu samotných dat, jenž paket nese. Tento tisk probíhá pomocí funkce *PrintData* a to v podobě bytů v hexadecimálním zápisu.

Funkce *PrintData* tiskne pouze takzvaný “datový payload”, protože mi přišlo zbytečné tisknout znovu hlavičku, akorát v bytové podobě. Taky mi přijde že tato změna zlepšuje přehlednost výstupu, protože nezobrazuje redundantní informace.

Po zpracování daného množství paketů se uzavře rozhraní a uvolní se všechna alokovaná paměť a program se ukončí.

Makefile

Pro účely mého projektu stačil úplně základní makefile, používající překladač g++. Jedinou zvláštní věcí bylo přidání parametru *-lpcap*, který slouží k přidání externí knihovny *pcap.h,* jenž je pro tento program naprosto klíčová.

Testování

Tento program jsem testoval na referenčním virtuálním systému za pomocí známého open source softwaru Wireshark. Porovnával jsem výstupy z Wiresharku s výstupy z mého vlastního programu.

Pakety jsem získával několika způsoby. Jedním z nich bylo běžné prohlížení internetu. Dalšími možnostmi byli například příkazy *ssh, curl* nebo *ping.*

Spuštění

Přeložený program spouštíme pomocí příkazu *sudo* a běžného linuxového spouštění *./*. Dále jsou přikládány argumenty. Mezi tyto argumenty patří:

- *-i interface* : Tento argument slouží k přidání rozhraní ze kterého se mají pakety sniffovat. V případě nepřítomnosti argumentu se vypíší všechna dostupná rozhraní.
- *-p port* : Pomocí toho argumentu se zadává port na kterém se mají pakety hledat. Pokud tento parametr není zadán, pakety se hledají na všech portech.
- *-n number* : Tento argument zadává počet hledaných paketů. Když není argument určen, hledá se pouze jeden paket.
- *--tcp* nebo *-t* : Pokud je tento argument zadán, program hledá pouze tcp pakety.
- *--udp* nebo *-u* : V případě tohoto argumentu hledá program pouze udp pakety.
- -help : Díky tomuto argumentu se vypíše pomoc uživatele, která ho primárně nasměruje k přečtení přiloženého README.

Pokud jsou zadány oba argumenty *tcp* a *udp*, pracuje program jako by nebyl zadán ani jeden.

Příklady spuštění a výstupu

Zde je přiloženo několik ukázkových vstupů a výstupů z referenčního virtuálního systému.

```
student@student-vm:~/Desktop$ sudo ./ipk-sniffer
Dostupna rozhrani:
enp0s3
any
lo
nflog
nfqueue
usbmon1
student@student-vm:~/Desktop$
```

```
student@student-vm:~/Desktop$ sudo ./ipk-sniffer -help
Tento program ma za ukol sniffovat pakety podle zadanych parametru.
Pro detailnejsi popis programu a pomoc se spustenim se obratte na dodane README.
student@student-vm:~/Desktop$
```

```
student@student-vm:~/Desktop$ sudo ./ipk-sniffer -i enp0s3
17:22:8.258230 10.0.2.15 : 35407 > 192.168.2.254 : 53

0x0000:    0C 64 65 74 65 63 74 70 6F 72 74 61 6C 07 66 69         .detectportal.fi
0x0010:    72 65 66 6F 78 03 63 6F 6D 00 00 01 00 01               refox.com.....

student@student-vm:~/Desktop$
```

```
student@student-vm:~/Desktop$ sudo ./ipk-sniffer -i enp0s3 -n 2 --udp
17:24:13.90398 10.0.2.15 : 52925 > 192.168.2.254 : 53

0x0000:    05 61 31 30 38 39 04 64 73 63 64 06 61 6B 61 6D         .a1089.dscd.akam
0x0010:    61 69 03 6E 65 74 00 00 01 00 01                        ai.net.....

17:24:13.90449 10.0.2.15 : 53491 > 192.168.2.254 : 53

0x0000:    05 61 31 30 38 39 04 64 73 63 64 06 61 6B 61 6D         .a1089.dscd.akam
0x0010:    61 69 03 6E 65 74 00 00 1C 00 01                        ai.net.....

student@student-vm:~/Desktop$
```

```
student@student-vm:~/Desktop$ sudo ./ipk-sniffer -i enp0s3 -n
Nenalezena hodnota argumentu -n. Program se ukonci.
student@student-vm:~/Desktop$
```

Bibliotéka

Zde budou uvedeny zdroje, ze kterých jsem přebíral vědomosti nezbytné pro tento projekt.

- Van Jacobson, Craig Leres and Steven McCanne, all of the Lawrence Berkeley National Laboratory, University of California, Berkeley, CA, February 16, 2020[online]. Dostupné z: https://www.tcpdump.org/manpages/pcap.3pcap.html
- Tim Carstens, 2002[online]. Dostupné z: https://www.tcpdump.org/pcap.html
- Silver Moon, 20. května 2013[online]. Dostupné z: https://www.binarytides.com/packet-sniffer-code-c-linux/