

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Počítačové sítě a komunikace – 2. projekt
Varianta ZETA - sniffer paketů

Obsah

1	Úvod	2
2	Popis programu	2
2.1	Argumenty	2
2.2	Návratové kódy	2
2.3	Formát výstupu	2
3	Implementace	3
3.1	Vytvořené struktury	3
3.1.1	sniff_ip	3
3.1.2	UDP_hdr	3
3.1.3	TCP_hdr	3
3.2	Funkce	4
3.2.1	main()	4
3.2.2	callback(u_char *args, const struct pcap_pkthdr *header, const u_char *packet)	4
3.2.3	print_payload(const u_char *payload, struct pcap_pkthdr header)	4
4	Testování	5

1 Úvod

Při vypracovávání projektu mi byly oporou internetové stránky, díky kterým jsem pochopil základní informace o práci s internetovými pakety a jejich princip. Stránka devdungeon[4] mi pomohla pochopit základy a stránka tcpdump[1] pokročilejší práci s knihovnou pcap.

2 Popis programu

`ipk-sniffer` je aplikace v jazyce C umožňující zachytávat a filtrovat pakety na určitém síťovém rozhraní. Podporované typy paketů jsou TCP a UDP. Sniffer pracuje pouze s pakety, které mají verzi IP adresy IPv4. Verze IPv6 podporována není.

Program využívá knihovny `libpcap`[3], která umožňuje právě zachytávání internetových paketů a to i těch, jež nejsou určeny přímo pro zařízení, na němž sniffer běží.

2.1 Argumenty

Program je po překladu možné spustit příkazem `sudo ./ipk-sniffer -i rozhraní [-p port] [--tcp|-t] [--udp|-u] [-n num]`. Při spuštění bez argumentů je vypsána krátká nápověda, s argumentem `--help` je pak vypsána celá nápověda. Jednotlivé argumenty:

- `-i rozhraní` - Povinný argument, pokud je zadáno pouze samotné `-i`, je vypsán seznam aktivních rozhraní a program se ukončí.
- `-p port` - Volitelný argument, filtruje pakety podle zadaného čísla portu `port`. Číslo portu je v rozsahu od 0 do 65535 včetně.
- `-t` nebo `--tcp` - Volitelný argument, je-li zadán, uvažují se pouze TCP pakety
- `-u` nebo `--udp` - Podobné jako `-t`, uvažuje pouze UDP pakety
- Pokud není `-tcp` ani `-udp` specifikováno, zobrazují se TCP i UDP pakety. Totéž platí, pokud jsou zadány oba z těchto argumentů.
- `-n num` - Volitelný argument, určuje kolik paketů se má zobrazit. Pokud je zadána hodnota 0 a menší, nebo jakákoliv nečíslná hodnota, budou se pakety vypisovat do té doby, než uživatel program ukončí.

2.2 Návrátové kódy

- 0 - Běh programu proběhl v pořádku.
- 1 - Chyba při špatně zadaných argumentech.
- 2 a 3 - Chyby které mohou nastat při problému v některé funkci z knihovny `pcap.h`.

2.3 Formát výstupu

Sniffer vypisuje informace o paketu na standardní výstup. Zobrazované informace jsou čas paketu, odchozí IP adresa, odchozí port, cílová IP adresa, cílový port a samotný obsah paketu. Obsah paketu je vypisován po 16ti bytech na řádek a to jak v hexadecimální podobě, tak v ascii znacích, netisknutelné znaky jsou nahrazeny tečkou. Na začátku každého řádku se nachází hexadecimální hodnota počtu vypsanych bytů na předchozích řádcích. Tvar výpisu paketu je inspirován programem **Wireshark** [2]. Jednotlivé pakety jsou mezi sebou odděleny dvojřádkovou mezerou.

```

student@student-vm:/media/sf_FITVUT/ipk/proj2_packet_sniffer$ sudo ./ipk-sniffer -i enp0s3 -n 2

12:51:33.866356 10.0.2.15 : 53444 > 216.58.201.99 : 80

0x0000  52 54 00 12 35 02 08 00 27 6f 35 b5 08 00 45 00  RT..5... 'o5...E.
0x0010  00 28 56 c3 40 00 40 06 36 60 0a 00 02 0f d8 3a  .(V.@.@. 6`.....:
0x0020  c9 63 d0 c4 00 50 bf 8f 0a 12 3d 3c c4 bf 50 10  .c...P.. ..=<..P.
0x0030  f9 2f ad c7 00 00                                ./....

12:51:33.866427 10.0.2.15 : 38690 > 93.184.220.29 : 80

0x0000  52 54 00 12 35 02 08 00 27 6f 35 b5 08 00 45 00  RT..5... 'o5...E.
0x0010  00 28 f1 fe 40 00 40 06 02 ed 0a 00 02 0f 5d b8  .(..@.@. ....].
0x0020  dc 1d 97 22 00 50 dd 63 96 2e 3d 65 c9 20 50 10  ...".P.c ..=e. P.
0x0030  f9 60 45 ff 00 00                                .`E...

```

Obrázek 1: Ukázka výstupu programu

3 Implementace

Při spuštění přeloženého kódu je nejprve zkontrolována správnost argumentů, provede se jejich zpracování a uložení zadaných filtrů. Pomocí `pcap_open_live()` je pak otevřeno zařízení, na které se aplikují uložené filtry a zavolá se funkce `pcap_loop()` pro samotné zachytávání paketů. Pro každý paket se volá funkce `callback()`, starající se o výpis informací paketu a díky funkci `print_payload()` také obsahu paketu. Na konci programu je zařízení na němž bylo zachytávání prováděno uzavřeno pomocí `pcap_close()`, a program končí.

Následující sekce se věnují popisu programové implementace řešení.

3.1 Vytvořené struktury

Následující struktury jsou v kódu využity především pro zjištění IP adres a portů. Mimo jiné se používá i například struktura pro ethernetovou hlavčiku z knihovny `net/ethernet.h`.

3.1.1 sniff_ip

Tato struktura je převzata ze stránky `tcpdump` [1]. Je využívána pro zjištění zdrojové a cílové IP adresy paketu.

3.1.2 UDP_hdr

Použito pro UDP pakety. Má v sobě dvě proměnné typu `u_short`: `uh_sport` pro zdrojový port a `uh_dport` pro cílový port.

3.1.3 TCP_hdr

Použito pro TCP pakety. Stejně jako `UDP_hdr` obsahuje dvě proměnné pro ukládání zdrojového a cílového portu paketu.

3.2 Funkce

3.2.1 main()

Zpracovává argumenty, otvírá zařízení pro zachytávání paketů a aplikuje filtry. Všechna rozhraní se vypisují s použitím funkce `pcap_findalldevs()`, jež vrácí seznam všech dostupných zařízení. Z toho je pak vybráno zvolené rozhraní.

Rozhraní je otevřeno pomocí funkce `pcap_open_live(dev, 65535, 1, 200, errbuf)`, kde `dev` je konkrétní zvolené zařízení, `65535` udává maximální délku paketu, `1` znamená, že je povolen promiskuitní mód, `200` je timeout pro výpis paketového bufferu v ms a do `errbuf` se ukládají případné errorry.[3]. Následně se kompilují a aplikují filtry přes funkce `pcap_compile()` a `pcap_setfilter`, a proběhne volání `pcap_loop` pro zpracovávání paketů.

3.2.2 callback(u_char *args, const struct pcap_pkthdr *header, const u_char *packet)

Tato funkce je volána pro každý jeden paket. Stará se o kontrolu typu paketu, výpis času, IP adres, portů paketu. Obsah paketu je zpracováván a vypisován pomocí `print_payload()`, jež se zde volá.

Z hlavičky `header` se zjistí čas, který je převeden a tisknut podle požadovaného formátu ze zadání. Z paketu se výpočtem zjistí pozice protokolu, pomocí něhož se rozhoduje, zda se má tisknout UDP nebo TCP paket. Volá se zde funkce pro výpis obsahu paketu `print_payload()`.

3.2.3 print_payload(const u_char *payload, struct pcap_pkthdr header)

Provádí výpis obsahu paketu ve formátu, jakým pakety vypisuje program **Wireshark**. Funkce běží v cyklu, který je ukončen po vytisknutí celého paketu. Tiskne se jednotlivě každý řádek po 16ti bytech.

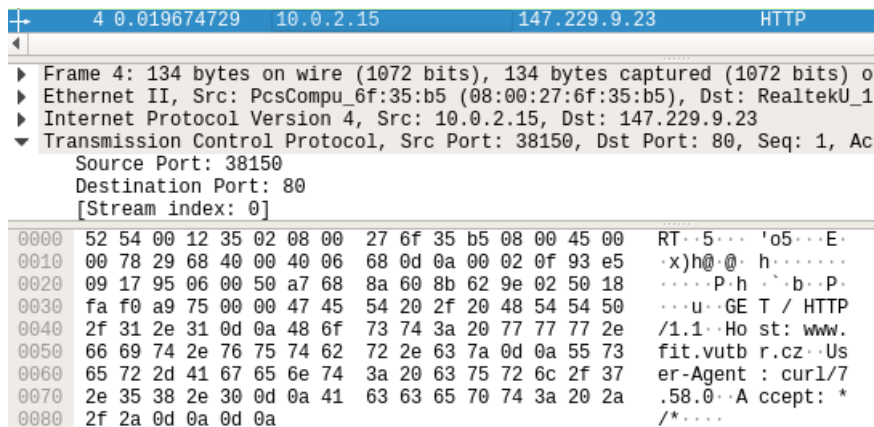
Řádek se tiskne nejdříve v hexadecimální podobě pomocí ukazatele, kterým je procházen. Pak se ten samý řádek projde druhým ukazatelem a tisknou se ascii znaky, jejichž tisknutelnost se kontroluje pomocí `isprint()`. Na začátku každého řádku mají tyto ukazatele stejnou hodnotu.

Toto se provádí pro každých 16 bytů z paketu, které představují 1 celý řádek. U posledního nekompletního řádku se provádí výpis stejně, jen je nutné doplnit správný počet mezer mezi posledním hexadecimálním tvarem a prvním ascii znakem pro lepší přehlednost. Po výtisku posledního řádku se funkce ukončí.

4 Testování

Při testování autor pracoval s programem Wireshark, který při vývoji sloužil jako etalon, proto jsou v tomto případě výpisy paketů z aplikace ipk-sniffer shodné s Wiresharkem.

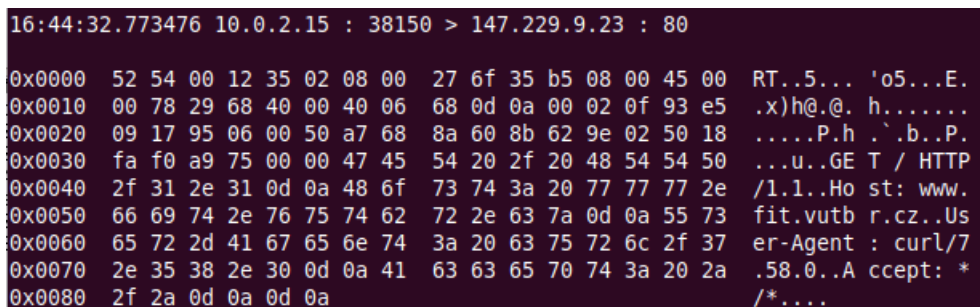
Na obrázcích výstupu z obou aplikací je vidět, že odpovídají IP adresy, porty i samotný obsah paketu:



The screenshot shows the Wireshark interface. The top bar indicates packet 4, source IP 10.0.2.15, destination IP 147.229.9.23, and protocol HTTP. The packet list pane shows details for Frame 4: 134 bytes on wire (1072 bits), 134 bytes captured (1072 bits) on Ethernet II, Src: PcsCompu_6f:35:b5 (08:00:27:6f:35:b5), Dst: RealtekU_1. The packet details pane shows Internet Protocol Version 4, Src: 10.0.2.15, Dst: 147.229.9.23, and Transmission Control Protocol, Src Port: 38150, Dst Port: 80, Seq: 1, Acc. The packet bytes pane shows the raw data in hex and ASCII.

Offset	Hex	ASCII
0000	52 54 00 12 35 02 08 00 27 6f 35 b5 08 00 45 00	RT..5... 'o5...E.
0010	00 78 29 68 40 00 40 06 68 0d 0a 00 02 0f 93 e5	.x)h@.@. h.....
0020	09 17 95 06 00 50 a7 68 8a 60 8b 62 9e 02 50 18P.h .`.b..P.
0030	fa f0 a9 75 00 00 47 45 54 20 2f 20 48 54 54 50	...u..GE T / HTTP
0040	2f 31 2e 31 0d 0a 48 6f 73 74 3a 20 77 77 77 2e	/1.1..Ho st: www.
0050	66 69 74 2e 76 75 74 62 72 2e 63 7a 0d 0a 55 73	fit.vutb r.cz..Us
0060	65 72 2d 41 67 65 6e 74 3a 20 63 75 72 6c 2f 37	er-Agent : curl/7
0070	2e 35 38 2e 30 0d 0a 41 63 63 65 70 74 3a 20 2a	.58.0..A ccept: *
0080	2f 2a 0d 0a 0d 0a	/*....

Obrázek 2: Výstup z Wiresharku



The screenshot shows the output of the ipk-sniffer application. It displays the packet details and hex data in a terminal window. The output is identical to the Wireshark screenshot.

Offset	Hex	ASCII
0x0000	52 54 00 12 35 02 08 00 27 6f 35 b5 08 00 45 00	RT..5... 'o5...E.
0x0010	00 78 29 68 40 00 40 06 68 0d 0a 00 02 0f 93 e5	.x)h@.@. h.....
0x0020	09 17 95 06 00 50 a7 68 8a 60 8b 62 9e 02 50 18P.h .`.b..P.
0x0030	fa f0 a9 75 00 00 47 45 54 20 2f 20 48 54 54 50	...u..GE T / HTTP
0x0040	2f 31 2e 31 0d 0a 48 6f 73 74 3a 20 77 77 77 2e	/1.1..Ho st: www.
0x0050	66 69 74 2e 76 75 74 62 72 2e 63 7a 0d 0a 55 73	fit.vutb r.cz..Us
0x0060	65 72 2d 41 67 65 6e 74 3a 20 63 75 72 6c 2f 37	er-Agent : curl/7
0x0070	2e 35 38 2e 30 0d 0a 41 63 63 65 70 74 3a 20 2a	.58.0..A ccept: *
0x0080	2f 2a 0d 0a 0d 0a	/*....

Obrázek 3: Výstup z ipk-snifferu

Reference

- [1] Carstens, T.: Tcpdump/Libpcap public repository. 2002. Dostupné z: <https://www.tcpdump.org/pcap.html>
- [2] Fundantion, W.: Download. Dostupné z: <https://www.wireshark.org/>
- [3] Group, T. T.: Manpage of PCAP. Dostupné z: <https://www.tcpdump.org/manpages/pcap.3pcap.html>
- [4] NanoDano: Using libpcap in C. Oct 2018. Dostupné z: <https://www.devdungeon.com/content/using-libpcap-c>