

Počítačové sítě a komunikace

Projekt č. 2 – Sniffer paketů

Michal Rein (xreinm00)

22.04.2020

1 Úvod do problematiky

Zadáním je vytvoření tzv. „packet snifferu“, který na zadaném síťovém zařízení odchyťává UDP a TCP pakety, které jsou následně v případě splnění vstupních podmínek vypsány v hlavičce se základními údaji, kterými jsou čas přijetí paketu a adresa IP, včetně portu jak příjemce, tak i odesilatele, a také je vypsán obsah dat jak v hexadecimálním, tak i ASCII formátu.

2 Implementace

Aplikace byla napsána v programovacím jazyce C++. Podrobnější popisy funkcí včetně parametrů a návratových hodnot lze nalézt v hlavičkovém souboru „main.h“.

2.1 Soubory projektu

- main.cpp
 - Soubor se zdrojovým kódem.
- main.h
 - Soubor s hlavičkou ke zdrojovému kódu.
- Makefile
 - Soubor pro sestavení programu pomocí příkazu „make“.

2.2 Použité knihovny

- <pcap/pcap.h>
 - Komunikace se síťovým zařízením, filtrace a samotné zachytávání paketů.
- <netdb.h>
 - Metody pro překlad IP adres na DNS adresu.
- <arpa/inet.h>
 - Struktury a metody pro práci s IP adresami.
- <getopt.h>
 - Zpracování vstupních argumentů.
- <iostream>
 - I/O výpisy.
- <iomanip>
 - I/O manipulátory (pro výpis paketů).

2.3 Popis implementace

Po spuštění aplikace jsou funkcí *parse_args()* zpracovány vstupní argumenty a dochází k nastavení příslušných indikátorů a proměnných, které ovlivňují chování programu. Jestliže nebyla změněna hodnota proměnné *interface*, která uchovává jméno síťového rozhraní, jsou pomocí metody *pcap_lookupdev()* vyhledány a vypsány dostupná zařízení. Všechny přebytné vstupní argumenty, případně vynechání jejich hodnot vede na ukončení programu s kódem 1.

Jestliže proběhlo zpracování argumentů bez chyby, nastane pokus o navázání spojení se síťovým zařízením. Odkaz na zařízení je přiřazen proměnné *handle*.

Po úspěšném otevření síťového zařízení je volána funkce *create_filter_expr()*, která z indikátorů *protocol_tcp*, *protocol_udp* a *port* sestaví výraz pro filtr. Ten je následně funkcí *pcap_compile()* zkompilován a voláním *pcap_setfilter()* nastaven.

Hlavní smyčka pro zachycování paketů je vytvořena voláním knihovni funkce *pcap_loop()*, které je předán jako parametr počet paketů, po jehož překročení dojde k ukončení smyčky a reference na funkci *parse_packet()*, která obsluhuje každý přijatý paket, vyhovující vstupním požadavkům filtru.

V hlavičkovém souboru jsou nadefinovány struktury, které mají uchovat informace obsahující hlavičky IP a TCP, případně UDP protokolů, jimiž jsou *header_ip*, *header_tcp* a *header_udp*. Při volání *parse_packet()* je funkci předán, mimo jiné parametry, také ukazatel na začátek souvislého řetězce bytů, které reprezentují jako celek právě onen paket.

Internetový paket obsahuje obecně tyto složky:

- Hlavička Ethernetu (14 bytů)
- Hlavička IP (20-60 bytů)
- Hlavička použitého protokolu (v našem případě TCP nebo UDP)
 - UDP (8 bytů)
 - TCP (20-60 bytů)
- Data (payload)

Kvůli proměnlivé délce hlaviček IP a TCP je nutné správně dopočítat odsazení (offset) ukazatele, aby ukazoval na začátek datové části paketu. Informace o délkách proměnlivých hlaviček jsou obsaženy v hlavičkách samotných, je tedy nutné využít již zmíněné nadefinované struktury k tomu, abychom získali informace o délce a mohli správně určit adresu začátku datové části.

Vzhledem k tomu, že pro naše účely není nutné zkoumat hlavičku Ethernetu, můžeme ji přeskočit a odkázat ukazatel posunutý o 14 bytů (makrem definovaný jako *ETHERNET_HEADER_SIZE*) struktury *header_ip*. Nyní můžeme získat informace nejen o délce samotné hlavičky, ale také délku zbytku paketu, IP adresy příjemce i odesílatele, a protokol, který je k přenosu použit (struktura obsahuje i další informace, ty ale pro naši aplikaci nepotřebujeme).

Informace o protokolu je předána funkci *check_protocol()*, která vrací jednu ze tří hodnot: *UDP*, *TCP* a *unknown*. Náš program zpracovává pouze UDP a TCP pakety, jestliže je nalezen jiný typ protokolu, paket je zahozen a aplikace čeká na příjem dalšího paketu.

V případě indentifikace UDP nebo TCP paketu, je vypočítán nový posun ukazatele, a ten předán příslušné struktuře protokolu. Nyní lze identifikovat porty příjemce i odesílatele a v případě TCP hlavičky její délku (UDP má fixní velikost, která je daná makrem *UDP_HEADER_SIZE*, která činí 8 bytů). Dále jsou vypsány základní informace o paketu pomocí funkcí *print_UDP_info()*, případně *print_TCP_info()*.

Nyní již zbývá pouze vypočítat novou pozici ukazatele, který by měl již odkazovat na začátek datové části. Výpis dat spravuje funkce *print_payload()*, která na každý řádek standardního výstupu vypíše 16 bytů dat jak v hexadecimálním, tak ASCII formátu.

3 Testování

Testování probíhalo porovnáváním výstupů této aplikace s volně dostupným softwarem Wireshark. Porovnávaly se především počty odchycených paketů za daný časový interval, adresy odesílatelů a příjemců, včetně jejich portů, a také výpis dat (payload), který se u obou aplikací shoduje.

Pro ilustraci jsou zde přiloženy snímky obrazovky obou aplikací, které odchytávaly pakety od stejného okamžiku, a jejichž nasbíraná data se shodují. Filtry jsou nastaveny na odchytávání TCP paketů obsahující port 443, data byla získána znovu obnovením webové stránky <https://wis.fit.vutbr.cz/FIT/>.

```

dw0lf@ubuntu: ~/Desktop/School/IPK/Project-2/sniffer
File Edit View Search Terminal Help

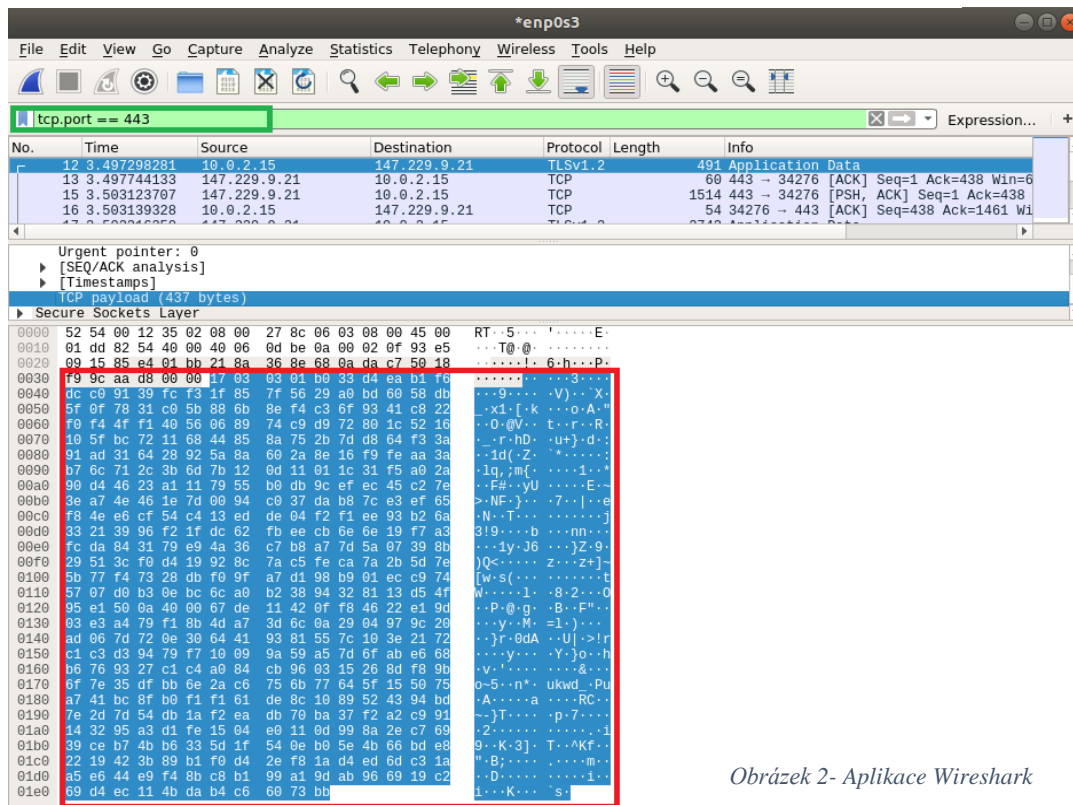
15:38:42 ubuntu : 34276 > agata.fit.vutbr.cz : 443 (TCP)
0x000000 17 03 03 01 b0 33 d4 ea b1 f6 dc c0 91 39 fc f3 .....3.....9..
0x000010 1f 85 7f 56 29 a0 bd 60 58 db 5f 0f 78 31 c0 5b ...V)..`X._.x1.[
0x000020 88 6b 8e f4 c3 6f 93 41 c8 22 f0 f4 4f f1 40 56 .k...o.A"...O.@V
0x000030 06 89 74 c9 d9 72 80 1c 52 16 10 5f bc 72 11 68 ..t...r..R...r.h
0x000040 44 85 8a 75 2b 7d d8 64 f3 3a 91 ad 31 64 28 92 D..u+).d:...1d(
0x000050 5a 8a 60 2a 8e 16 f9 fe aa 3a b7 6c 71 2c 3b 6d Z.`*.....:lq;m
0x000060 7b 12 0d 11 01 1c 31 f5 a0 2a 90 d4 46 23 a1 11 {.....1.*.F#..
0x000070 79 55 b0 db 9c ef ec 45 c2 7e 3e a7 4e 46 1e 7d yU.....E..~>.NF.}
0x000080 00 94 c0 37 da b8 7c e3 ef 65 f8 4e e6 cf 54 c4 ...7..|.e.N..T.
0x000090 13 ed de 04 f2 f1 ee 93 b2 6a 33 21 39 96 f2 1f .....j3!9...
0x0000a0 dc 62 fb ee cb 6e 6e 19 f7 a3 fc da 84 31 79 e9 .b...nn.....1y.
0x0000b0 4a 36 c7 b8 a7 7d 5a 07 39 8b 29 51 3c f0 d4 19 J6....}Z.9.)Q<...
0x0000c0 92 8c 7a c5 fe ca 7a 2b 5d 7e 5b 77 f4 73 28 db ..Z...z+]-~[w.s(
0x0000d0 f0 9f a7 d1 98 b9 01 ec c9 74 57 07 d0 b3 0e bc .....tw....
0x0000e0 6c a0 b2 38 94 32 81 13 d5 4f 95 e1 50 0a 40 00 l..8.2...O..P.@
0x0000f0 67 de 11 42 0f f8 46 22 e1 9d 03 e3 a4 79 f1 8b g..B..F".....y.
0x000100 4d a7 3d 6c 0a 29 04 97 9c 20 ad 06 7d 72 0e 30 M.=l.)...}r.0
0x000110 64 41 93 81 55 7c 10 3e 21 72 c1 c3 d3 94 79 f7 dA..U|>!r....y.
0x000120 10 09 9a 59 a5 7d 6f ab e6 68 b6 76 93 27 c1 c4 ...Y.}o..h.v'..
0x000130 a0 8a cb 96 03 15 26 8d f8 9b 6f 7e 35 df bb 6e .....&...o~5..n
0x000140 2a c6 75 6b 77 64 5f 15 50 75 a7 41 bc 8f b0 f1 *.ukwd..Pu.A...
0x000150 f1 61 de 8c 10 89 52 43 94 bd 7e 2d 7d 54 db 1a .a....RC...~}T..
0x000160 f2 ea db 70 ba 37 f2 a2 c9 91 14 32 95 a3 d1 fe ...p.7.....2....
0x000170 15 04 e0 11 0d 99 8a 2e c7 69 39 ce b7 4b b6 33 .....i9..K.3
0x000180 5d 1f 54 0e b0 5e 4b 66 bd e8 22 19 42 3b 89 b1 ].T..^Kf..".B;..
0x000190 f0 d4 2e f8 1a d4 ed 6d c3 1a a5 e6 44 e9 f4 8b .....m....D...
0x0001a0 c8 b1 99 a1 9d ab 96 69 19 c2 69 d4 ec 11 4b da .....i..i...K.
0x0001b0 b4 c6 60 73 bb ..`s.

15:38:42 agata.fit.vutbr.cz : 443 > ubuntu : 34276 (TCP)

15:38:42 agata.fit.vutbr.cz : 443 > ubuntu : 34276 (TCP)
0x000000 17 03 03 10 30 d4 a8 39 9b 56 fe e5 8d be 53 58 ....0..9.V....SX
0x000010 a1 fe e9 0e 16 7d ad 95 0d 2a aa 96 96 79 12 66 .....}...*.y.f
0x000020 7c c8 6b 07 4c ff ad 58 e1 79 a3 67 3e 37 a2 38 |.k.L..X.y.g>7.8
0x000030 0f 0e f1 ea ac 7e ba 27 82 29 40 44 02 23 66 a1 .....~.'.)@D.#f.

```

Obrázek 1- Aplikace ipk-sniffer



Obrázek 2- Aplikace Wireshark

Obsah

1	Úvod do problematiky	2
2	Implementace.....	2
2.1	Soubory projektu	2
2.2	Použité knihovny	2
2.3	Popis implementace.....	3
3	Testování.....	4