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 odchytává 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 kódém.
- 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 idikátorů 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 přebytečné 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 knihovní 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*) struktuře *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 fukci *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.

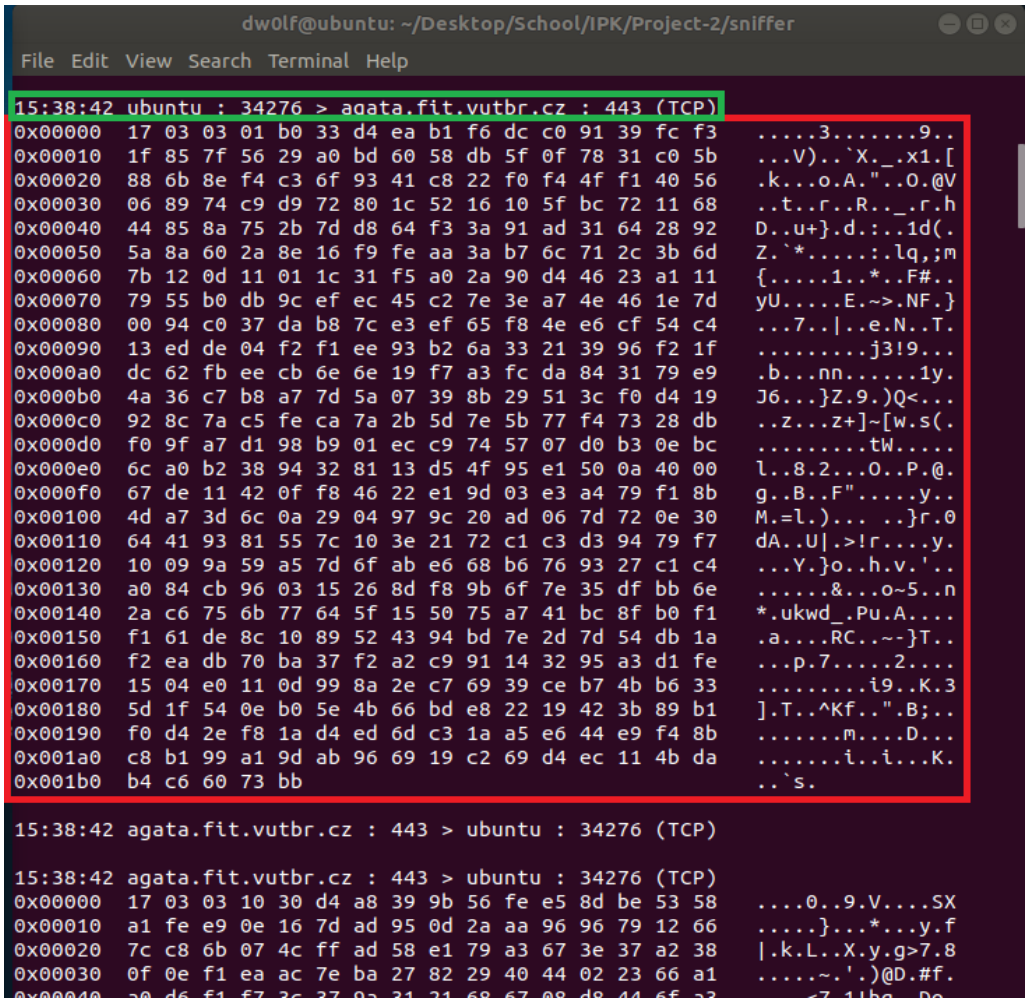
V případě indentifikace UDP nebo TCP paketu, je vypočítán nový posun ukazatele, a ten předán příslušné struktuře protokolu. Nyní lze identifikovat porty příjemce i odesilatele a v případě TCP hlavičky její délku (UDP má fixní velikost, která je daná makrem *UDP_HEADER_SIZE*, která činí 8 bytů). Dále 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 fukce *print_payload(),* která na každý řádek standardního výstupu vypíše 16 bytů dat jak v hexadecimálním, tak ASCII formátu.

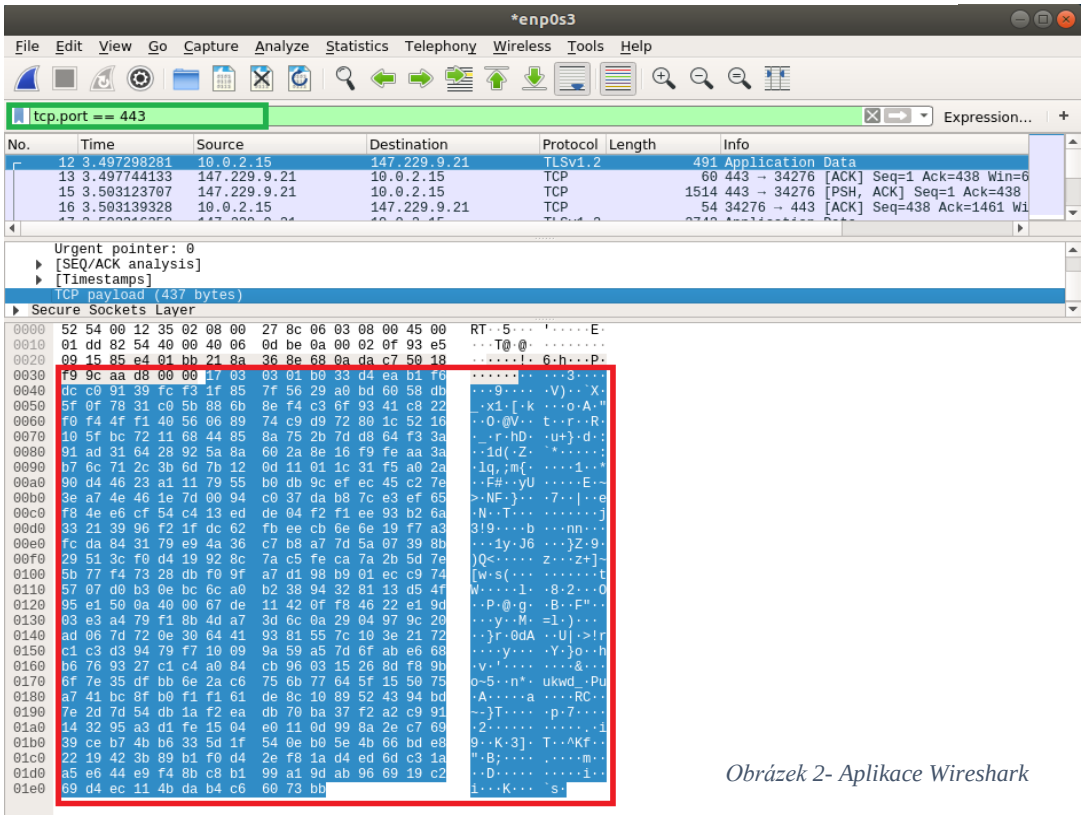
3 Testování

Testování probíhalo porovnáváním výstupů této aplikace s volně dostupným softwarem Wireshark. Porovnávaly se především počty odchycených paketů za daný časový interval, adresy odesilatelů 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/.



Obrázek 1- Aplikace ipk-sniffer



Obrázek 2- Aplikace Wireshark

Obsah