Vysoké učení technické v Brně

Fakulta informačních technologií

Počítačové komunikace a sítě

2019/2020

Projekt

Sniffer paketů

Vojtěch Jurka (xjurka08) 29.04.2020

Obsah

1 Zadání

Navrhněte a implementujte síťový analyzátor v C/C++/C#, který bude schopný na určitém síťovém rozhraním zachytávat a filtrovat pakety.

2 Implementace

Program je implementovaný v jazyce C. Používá k funkčnosti knihovnu pcap[1] a jako nástroj k překladu g++. Překládán je pomocí nástroje make (příkaz: make ipk-sniffer).

2.1 Samotný program

Program začíná zpracováváním argumentů předaných při spuštění. Pro toto je vyzužita funkce getopt_long(), která zajišťuje, aby některé argumenty bylo možné zadávat jejich dlouhou i zkrácenou verzí, např. „-t“ a „--tcp“. Pokud není zadané rozhraní, na kterém má program zachytávat pakety, vypíše se nápověda k jeho použití a následně seznam všech aktivních rozhraní.

Pokud je zařízení specifikováno, program začne provádět tzv. packet sniffing. Začne tím, že otevře dané zařízení a vytvoří a aplikuje na něj filtr, který určuje, jaký druh paketů má zachytávat. Typ zadaného filtru je dán argumenty programu. Filtrovat se může dle TCP/UDP protokolu a portu zařízení. Dále nastává získávání paketů. Pro toto je využita funkce pcap_loop(), která zachytí počet paketů zadaný argumentem programu a při každém paketu zavolá callback funkci zpracujPaket().

Dále se už jen zařízení uzavře a program končí činnost.

2.2 Pomocné funkce

V kódu programu jsou implementovány dvě pomocné funkce.

2.2.1 void zpracujPaket()

Toto je callback funkce volaná funkcí pcap_loop(). Zpracovává jednotlivé pakety. Jako argumenty přijímá strukturovanou proměnnou obsahující informace o hlavičce paketu a ukazatel na začátek samotných dat celého paketu.

Zpracovávání začíná ukládáním informací z jednotlivých hlaviček paketu do struktur. Takto se získává ethernet hlavička, dále IP hlavička a TCP/UDP hlavička. IP a TCP/UDP hlavička se kontroluje na správnou délku. Dále program získá timestamp paketu, zdrojovou a IP adresu destinace. Dále program zjišťuje, zda se jedná o TCP, či UDP paket a následně vypisuje dosud získané informace na standardní výstup.

Dále program vypisuje na standardní výstup data paketu pomocí funkce printData().

[1] https://www.tcpdump.org/manpages/pcap.3pcap.html

2.2.2 printData()

Tato funkce slouží k vypisování dat paketu na standardní výstup v patřičném formátu. Jako argumenty bere ukazatel na počáteční adresu s daty a číslo reprezentující bajtovou délku paketu.

V jednom velkém cyklu, který se řídí délkou zbývajících dat paketu funkce nejprve vypočte délku aktuálního řádku dat, který má vypisovat. Poté na standardní výstup vypisuje hexadecimální reprezentaci čísla, které symbolizuje ofset dat, které tiskne, vzhledem k začátku dat paketu. Dále funkce vypisuje maximálně 16 bajtů dat na řádek v hexadecimální bajtové reprezentaci. Dále na konec řádku vypisuje ta samá data, ale tentokrát v ASCII znakovém formátu. Pokud daný bajt nelze přeložit na čitelný ASCII znak, je na jeho místo vypsána tečka. Dále je posunut datový ofset o danou délku řádku a cyklus tiskne další řádek, dokud program nedojde na konec paketu.

3 Testování

Pro testování programu byl použit open-source nástroj Wireshark[2], který slouží právě ke sledování paketů na zařízení.

Po spuštění v referenčním obrazu systému čekal testovaný program na pakety. Jejich pohyb na síti byl docílen pomocí dalšího open-source nástroje curl[3], kterým byl poslán http request. Program tyto pakety zachytil a úspěšně vypsal (viz obrázek).

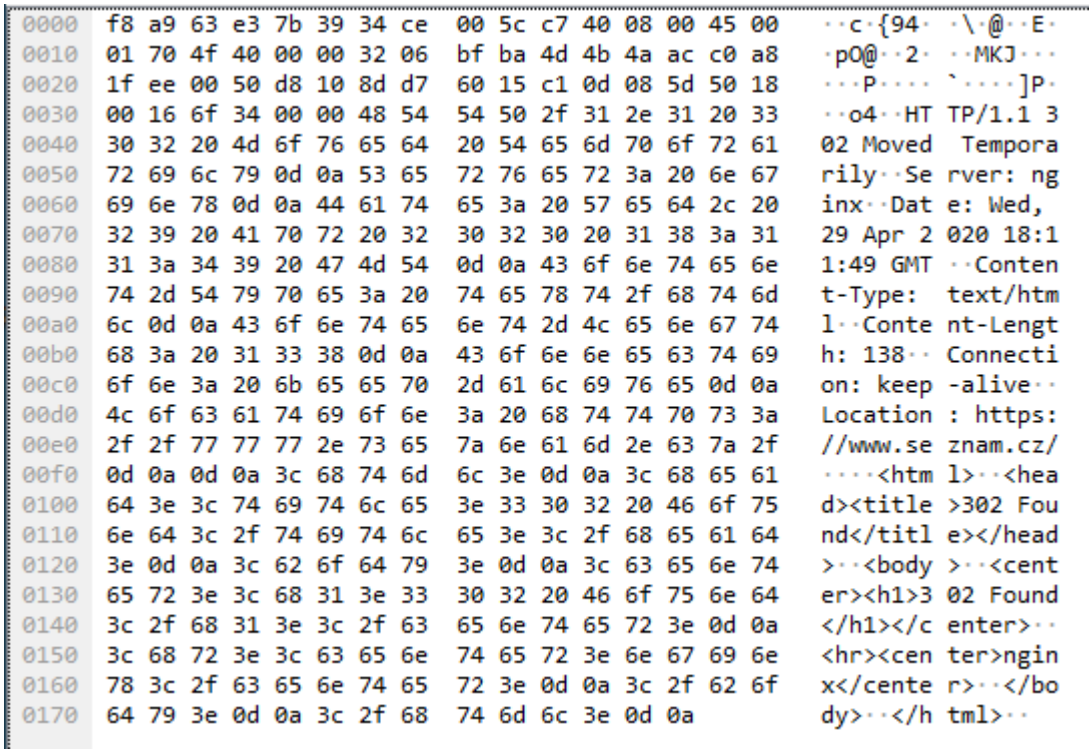
```
20:00:25.318210 www.seznam.cz : 80 > student-vm : 40032

0x0000:  08 00 27 6f 35 b5 52 54  00 12 35 02 08 00 45 00   ..'o5.RT..5...E.
0x0010:  01 70 02 9f 00 00 40 06  d1 df 4d 4b 4b b0 0a 00   .p....@...MKK...
0x0020:  02 0f 00 50 9c 60 01 4b  0e 02 ae 43 91 01 50 18   ...P.`.K...C..P.
0x0030:  ff ff ea 47 00 00 48 54  54 50 2f 31 2e 31 20 33   ...G..HTTP/1.1 3
0x0040:  30 32 20 4d 6f 76 65 64  20 54 65 6d 70 6f 72 61   02 Moved Tempora
0x0050:  72 69 6c 79 0d 0a 53 65  72 76 65 72 3a 20 6e 67   rily..Server: ng
0x0060:  69 6e 78 0d 0a 44 61 74  65 3a 20 57 65 64 2c 20   inx..Date: Wed,
0x0070:  32 39 20 41 70 72 20 32  30 32 30 20 31 38 3a 30   29 Apr 2020 18:0
0x0080:  30 3a 32 36 20 47 4d 54  0d 0a 43 6f 6e 74 65 6e   0:26 GMT..Conten
0x0090:  74 2d 54 79 70 65 3a 20  74 65 78 74 2f 68 74 6d   t-Type: text/htm
0x00a0:  6c 0d 0a 43 6f 6e 74 65  6e 74 2d 4c 65 6e 67 74   l..Content-Lengt
0x00b0:  68 3a 20 31 33 38 0d 0a  43 6f 6e 6e 65 63 74 69   h: 138..Connecti
0x00c0:  6f 6e 3a 20 6b 65 65 70  2d 61 6c 69 76 65 0d 0a   on: keep-alive..
0x00d0:  4c 6f 63 61 74 69 6f 6e  3a 20 68 74 74 70 73 3a   Location: https:
0x00e0:  2f 2f 77 77 77 2e 73 65  7a 6e 61 6d 2e 63 7a 2f   //www.seznam.cz/
0x00f0:  0d 0a 0d 0a 3c 68 74 6d  6c 3e 0d 0a 3c 68 65 61   ....<html>..<hea
0x0100:  64 3e 3c 74 69 74 6c 65  3e 33 30 32 20 46 6f 75   d><title>302 Fou
0x0110:  6e 64 3c 2f 74 69 74 6c  65 3e 3c 2f 68 65 61 64   nd</title></head
0x0120:  3e 0d 0a 3c 62 6f 64 79  3e 0d 0a 3c 63 65 6e 74   >..<body>..<cent
0x0130:  65 72 3e 3c 68 31 3e 33  30 32 20 46 6f 75 6e 64   er><h1>302 Found
0x0140:  3c 2f 68 31 3e 3c 2f 63  65 6e 74 65 72 3e 0d 0a   </h1></center>..
0x0150:  3c 68 72 3e 3c 63 65 6e  74 65 72 3e 6e 67 69 6e   <hr><center>ngin
0x0160:  78 3c 2f 63 65 6e 74 65  72 3e 0d 0a 3c 2f 62 6f   x</center>..</bo
0x0170:  64 79 3e 0d 0a 3c 2f 68  74 6d 6c 3e 0d 0a         dy>..</html>..
```

Obrázek 1: Výpis paketu HTTP

[2] https://www.wireshark.org/
[3] https://curl.haxx.se/

Dále byl tento stejný postup zopakován s nástrojem Wireshark. Výsledný paket byl také zachycen.



Obrázek 2: Výpis paketu HTTP v nástroji Wireshark

Jak je vidět, výsledné pakety jsou v obou případech kromě hlaviček (pochopitelně) naprosto identické s jednou výjimkou. Tou je skutečnost, že náš program neodděluje ASCII data v prostředku řádku mezerou, je tak učiněno s cílem lepší čitelnosti těchto textových dat. Tento test ukazuje, že oba programy pracují stejným způsobem.