Počítačové sítě a komunikace
2019 / 2020

Projekt 2 – Varianta ZETA: Sniffer paketů

Vypracoval: Jan Lorenc (xloren15) Datum: 12. 4. 2020

Obsah

O programu

Sniffer paketů je v obecnosti program, který poslouchá na síťových rozhraních a zachytává pakety. Po ukončení činnosti typicky vypíše uživateli seznam paketů, které zachytil, včetně relevantních informací, jako je čas, kdy byl paket zachycen, odesilatele, příjemce, data a jiné.

Tato implementace sniffer programu zachycuje pouze TCP a UDP pakety a to na právě jednom síťovém rozhraní. Po spuštění vyhledá všechna dostupná rozhraní a najde mezi nimi to, jež uživatel zadal. Pokud nenalezne, vypíše dostupná rozhraní na obrazovku. Jakmile naváže spojení s rozhraním, dá uživateli vědět, že již poslouchá. Podobně hlásí, že zrovna zachytil nějaký paket. Může se stát, že žádný paket dlouhou dobu nedorazí, proto je vhodné uživatele takto informovat o postupu, aby se nezalekl, že program spadnul, nebo nepracuje, jak má. Po zachycení požadovaného počtu paketů program oznámí, že přestal poslouchat, zpracuje pakety a následně je vypíše na standardní výstup.

Výstupní formát paketu je následující. Každý obsahuje hlavičku začínající časovou známkou zachycení paketu, poté doménové jméno a číslo portu odesilatele i příjemce. Pokud se nepodařilo získat doménové jméno, vypíše se místo něj IP adresa. Za hlavičkou je jeden prázdný řádek a data. Jeden řádek dat obsahuje číslo udávající počet bytů zpracovaných do tohoto řádku v hexadecimální soustavě, poté 16 bytů dat opět v hexadecimální soustavě a na závěr těchto 16 bytů převedených do ASCII, kde netisknutelné znaky, tedy hodnoty mimo rozsah 32-126, jsou reprezentovány tečkou.

Implementace

Program je psán v jazyce C# na platformě .NET Core 3.1 a jeho zdrojové soubory lze nalézt ve složce `src`. Kromě základu .NET Core využívá ještě nuget balík SharpPcap 5.1. Jedná se o opensource knihovnu umožňující zachytávání paketů, díky podpoře libpcap na Linuxu a Npcap (dříve WinPcap) na Windows. Jelikož se jedná o nuget balík, stáhne se automaticky při sestavení nebo lze i manuálně příkazem `dotnet restore`.

Projekt obsahuje celkem 5 zdrojových souborů:

ipk-sniffer.csproj

Standardní .net soubor obsahující samotnou konfiguraci projektu.

Program.cs

Implementuje spouštěcí metodu programu `main()`, ve které se zkontrolují vstupní parametry a následně se spustí zachytávání paketů.

Arguments.cs

Obsahuje třídu starající se o kontrolu a zpracování vstupních argumentů programu. Nedovolí uživateli zadat stejný parametr vícekrát, ani různé se stejným významem (např. -t/--tcp). Pokud uživatel zadá parametr -h/--help, vypíše nápovědu obsahující velmi stručný popis programu a podporované argumenty. Tento parametr nemusí stát samostatně. I když je vše správně nastaveno, ale je zadán tento parametr, program se nespustí, nýbrž vypíše nápovědu. Nespecifikuje-li uživatel síťové rozhraní nebo zadá neexistující, vypíše seznam dostupných rozhraní.

Přehled podporovaných argumentů:

-h / --help Zobrazí nápovědu.
-i [jméno] Jediný povinný parametr programu. Vyžaduje se jméno síťového rozhraní.
-p [číslo] Číslo portu, na kterém se zachytávají pakety. Výchozí nastavení jsou všechny porty.
-t / --tcp Určuje, zda-li se mají zachytávat pouze TCP pakety.
-u / --udp Určuje, zda-li se mají zachytávat pouze UDP pakety.
-n [číslo] Počet paketů, které se mají zachytit. Výchozí nastavení je 1 paket.

Ve výchozím nastavení se zachytávají oba typy paketů, TCP i UDP. Jsou-li specifikovány obě možnosti, opět se zachytávají oba typy.

Sniffer.cs

Implementuje třídu starající se o samotné zachytávání paketů. Není však příliš složitá. Obsahuje pouze dvě metody. První metoda `Sniff()` nejprve otevře spojení se specifikovaným rozhraním a následně ve smyčce zachytává pakety. Při úspěšné identifikaci TCP nebo UDP paketu pro tento vytvoří instanci třídy SniffedPacket (viz. dále) a uloží ji do seznamu zachycených paketů. Dále vypíše uživateli, že se zrovna zachytil paket, aby věděl, že se něco děje. Po zachycení zadaného počtu paketů uzavře spojení s rozhraním a ukončí se. Druhou metodou je `Out()`, která pouze vypíše zachycené pakety na standardní výstup.

SniffedPacket.cs

Tato třída reprezentuje zachycený paket ve formátu pro výstup. Pro paket tedy sestaví výstupní hlavičku a data zformátuje do požadovaného tvaru (viz. popis výstupu v sekci O programu).

Překlad a spuštění

Program ke svému chodu vyžaduje pouze nainstalovaný dotnet core 3.1. Knihovna SharpPcap 5.1 se, jak již bylo zmíněno, stáhne automaticky sama. K projektu je vytvořený makefile schopný tří funkcí. Příkaz `make` nebo `make build` projekt sestaví. Spustitelný program se pak bude nacházet v adresáři `./src/bin/Debug/netcoreapp3.1` a odtud lze spustit dle zadání: `./ipk-sniffer {argumenty}`.

Příkazem `make restore` lze manuálně stáhnout/aktualizovat knihovnu SharpPcap, ale i jiné případné závislosti. Dále příkaz `make clean` vymaže sestavený program, vyčistí tedy složku `./src/bin/ Debug/netcoreapp3.1`. Tyto make příkazy však pouze obalují dotnet CLI, tedy příkazy dotnet build [projekt], dotnet restore [projekt], dotnet clean [projekt].

Příklad spuštění na referenčním stroji po rozbalení projektu:

```
make
./src/bin/Debug/netcoreapp3.1/ipk-sniffer -i enp0s3 -n 2
```

Ukázka výstupu

Takto vypadá sestavení a spuštění programu na referenčním stroji pomocí předchozích dvou příkazů:

```
root@student-vm:/media/sf_IPK/Projekt 2# make
dotnet build ./src/ipk-sniffer.csproj
Microsoft (R) Build Engine version 16.4.0+e901037fe for .NET Core
Copyright (C) Microsoft Corporation. All rights reserved.

  Restore completed in 46.75 ms for /media/sf_IPK/Projekt 2/src/ipk-sniffer.csproj.
  ipk-sniffer -> /media/sf_IPK/Projekt 2/src/bin/Debug/netcoreapp3.1/ipk-sniffer.dll

Build succeeded.
    0 Warning(s)
    0 Error(s)

Time Elapsed 00:00:01.75
root@student-vm:/media/sf_IPK/Projekt 2# ./src/bin/Debug/netcoreapp3.1/ipk-sniffer -i enp0s3 -n 2
Sniffing...
Captured UDP packet.
Captured UDP packet.
Sniffing finished.
Captured packets:

07:11:51.361 student-vm : 5353 > 224.0.0.251 : 5353

0x0000: 14 E9 14 E9 00 35 ED 50  00 00 00 00 00 02 00 00 .....5.P ........
0x0010: 00 00 00 00 05 5F 69 70  70 73 04 5F 74 63 70 05 ....._ip ps._tcp.
0x0020: 6C 6F 63 61 6C 00 00 0C  00 01 04 5F 69 70 70 C0 local... ..._ipp.
0x0030: 12 00 0C 00 01 .....

07:11:52.319 student-vm : 41525 > 192.168.0.1 : 53

0x0000: A2 35 00 35 00 30 CC F9  97 74 01 00 00 01 00 00 .5.5.0.. .t......
0x0010: 00 00 00 00 02 31 35 01  32 01 30 02 31 30 07 69 .....15. 2.0.10.i

root@student-vm:/media/sf_IPK/Projekt 2#
```

Testování

Pro testování funkčnosti jsem využil program Netcat, který byl pro vývoj projektu ideální volbou, neboť umí velmi jednoduše posílat TCP i UDP pakety na zadanou IP adresu i port. Mohl jsem jím tedy otestovat všechny funkcionality snifferu. Program se z příkazového řádku využívá následovně:

```
nc [ip-adresa] [port]      - v základu posílá TCP pakety, nemusí se tedy řešit nic víc
nc -u [ip-adresa] [port]   - příklad pro odeslání UDP paketu
```

Porovnat výsledky lze například pomocí open source programu Wireshark nebo pro pokročilé uživatele je k dispozici třeba Network Packet Capture Tool od firmy Solarwinds.

Použité zdroje

Github dokumentace knihovny SharpPcap: https://github.com/chmorgan/sharppcap