Vysoké učení technické v Brně
Fakulta informačních technologií

Sniffer packetů

Varianta ZETA

Michal Halabica (xhalab00)
IPK 2020

Obsah

1. Úvod

Cílem aplikace je sledování packetů síťového provozu. Pro sledování síťového provozu se používá síťové rozhraní, které je specifikováno při spuštění aplikace.

Aplikace podporuje výpis TCP, UDP a ICMP packetů. Zbylé druhy packetů jsou ignorovány.

2. Platforma a závislosti

Aplikace je psaná v jazyce C# a jako platforma pro tento jazyk se používá platforma .NET Core 3.1

.NET Core je open-source platforma běžící napříč platformami operačních systémů (Windows, Unix (vč. MacOS)).

2.1. Knihovny

Veškeré knihovny jsou instalovány z balíčkovacího systému NuGet, díky tomu nemusí být knihovny distribuovány při přenosu aplikace. Při překladu se všechny potřebné knihovny stáhnou automaticky.

Pro implementaci aplikace bylo využito následujících knihoven:

- CommandLineParser
  Knihovna pro zpracování parametrů příkazové řádky a jejich uložení do struktury.
- SharpPcap
  Knihovna zajišťující multiplatformní práci s knihovnami WinPcap (Windows) a LibPcap (Unix) v prostředí jazyků z rodiny .NET. Pro správný chod této knihovny je potřeba na cílovém zařízení mít nainstalovanou příslušnou knihovnu v závislosti na operačním systému.

3. Překlad

Pro přeložení aplikace lze použít nástrojů přímo ve vývojovém prostředí jako je Microsoft Visual Studio, nebo JetBrains Rider.

Vzhledem k tomu, že je aplikace psaná pod platformou .NET Core 3.1 lze aplikaci také přeložit pomocí utility „dotnet“.

Pro jednodušší překlad je u zdrojových souborů aplikace dodán i Makefile, který obsahuje následující pravidla:

- build
  Překlad aplikace. Toto pravidlo volá utilitu dotnet s následujícími parametry:
  - build – Příkaz pro utilitu, že se má provádět překlad.
  - „ipk-sniffer/“ – Cesta k projektu.
  - „-c Release“ – Překlad pro finální produkční prostředí.
  - „-o build/“ – Cesta k adresáři, kde se uloží sestavená aplikace a potřebné knihovny.
- clean
  Vyčištění výstupního adresáře. Toto pravidlo volá příkaz „rm -rf build/“.

4. Spuštění

Po úspěšném přeložení aplikace je dostupný spustitelný binární soubor v závislosti na platformě, na které byl překlad proveden.

Pro Windows: ipk-sniffer.exe

Pro Unix: ipk-sniffer

Aplikace ke svému chodu vyžaduje v unixových systémech zvýšená práva (tzn. Sudo). V operačních systémech z rodiny Windows nemusí být zvýšená práva požadována (v závislosti na konfiguraci operačního systému).

Po spuštění aplikace a úspěšnému zahájení sledování packetů aplikace čeká na příchozí packety, které má zpracovat. Zpracování může být omezeno zadáním parametrů na příkazové řádce. Parametry jsou popsány v podkapitole 4.1.

Aplikace nemá časový limit, do kdy má zachytit packet, tudíž pokud aplikace nezachytí packet, tak může běžet po nekonečně dlouhou dobu, případně lze ukončit klávesovou zkratkou CTRL+C (SIGINT).

4.1. Parametry

./ipk-sniffer -i interface [-p port1, port2, …] [--tcp|-t] [--udp|-u] [--icmp] [-n count]

Parametr	*Popis*
-i interface	Název rozhraní, na kterém budou packety sledovány. Povinný parametr, pokud není zadán, nebo byl zadán pouze parametr „-i“ bez hodnoty, tak se provede výpis všech aktivních rozhraní. Za aktivní rozhraní se považuje zařízení, které má přiřazenou platnou IPv4, nebo IPv6 adresu. Pokud bude rozhraní nalezeno, tak aplikace končí s návratovým kódem 1.
-p port1,port2…	Seznam portů, které má aplikace zpracovávat. Oproti základnímu zadání lze zadat více portů. Oddělovač je znak čárka (ASCII znak 44). Tento parametr nemá vliv na ICMP packety a to z toho důvodu, že ICMP packety pracují na 3. vrstvě (síťová vrstva), kde probíhá komunikace pomocí IP adres. Pokud bude nějaký ze zadaných portů neplatný (mimo povolený rozsah portů), tak aplikace bude ukončena s návratovým kódem 3.
--tcp|-t	Omezení zobrazení pouze TCP packetů. Lze více specifikovat výše zmíněným parametrem -p.
--udp|-u	Omezení zobrazení pouze UDP packetů. Lze více specifikovat parametrem -p.
--icmp	Omezení zobrazení pouze ICMP packetů. Tento parametr zahrnuje jak ICMPv4 packety, tak ICMPv6. Podpora ICMP packetů byla implementována nad rámec původního zadání.

-n count	Počet packetů, které má aplikace zpracovat, než dojde k jejímu ukončení. Lze zadat hodnoty v intervalu <1, INT_MAX>, kde INT_MAX=2 147 483 647.

Parametry [--tcp|-t], [--udp|-u] a [--icmp] lze mezi sebou kombinovat. Pokud nebyl žádný z těchto parametrů zadán, tak je to chápáno, že jsou zadány všechny tři.

4.2. Návratové kódy

Hodnota	*Popis*
0	Úspěšné a korektní ukončení aplikace.
1	Bylo zadáno neplatné rozhraní.
2	Interní chyba knihovny SharpPcap, nebo knihoven WinPcap/LibPcap (podle OS).
3	Byl zadán neplatný port (Neplatný rozsah). Povolený rozsah je <0; 65535>
99	Interní chyba aplikace.

5. Implementace

5.1. Výpis aktivních rozhraní

Pro výpis aktivních rozhraní bylo využito knihovny SharpPcap, která obsahuje funkci pro zjištění seznamu rozhraní (CaptureDeviceList.New()). Interně knihovna volá funkci pcap_findalldevs, provede datové transformace a vrátí seznam rozhraní.

Aplikace po zjištění seznamu zařízení pomocí knihovny provede pokus o připojení na dané rozhraní. Pokud nelze s rozhraním pracovat, nebo aplikace nemá dostatečná práva pro práci s rozhraním, tak je rozhraní ignorováno.

Pokud rozhraní nemá přiřazenou IPv4, nebo IPv6 adresu, tak je zařízení ignorováno.

Formát výstupu seznamu aktivních rozhraní je následující:

```
Name: nazev-rozhrani
FriendlyName: human-readable nazev rozhrani
Addresses (IPv4): seznam_ipv4_adres_oddelene_carkou
Addresses (IPv6): seznam_ipv6_adres_oddelene_carkou
Gateway addresses (IPv4): Seznam_ipv4_adres_bran_oddelene_carkou
Gateway addresses (IPv6): Seznam_ipv6_adres_bran_oddelene_carkou
```

Položky výstupu Addresses (IPv4), Addresses (IPv6), Gateway addresses (IPv4) a Gateway addresses (IPv6) nemusí být vypsány. Nejsou vypsány v případě, že rozhraní nemá daný druh adresy přiřazen.

Veškerá implementace výpisu seznamu aktivních rozhraní je implementována ve statické třídě Services.InterfaceListing.InterfaceListing v souboru (Services/InterfaceListing/InterfaceListing.cs).

5.2. Zachytávání packetů

Logika zachytávání packetů a jejich zpracování je rozdělena do následujících souborů a funkcí:

- Services/Sniffer/Sniffer.cs – Logika zachytávání packetů, filtrace packetů.

    - V tomto souboru je umístěna statická třída Sniffer, která obsahuje hlavní funkci Process(Options options) a několik pomocných funkcí.
    - Celý proces zachytávání packetů je postaven na událostech. Tzn. Když aplikace obdrží packet, tak se vyvolá událost Device_OnPacketArrival, kde v parametrech této události je obdržený packet.
    - Funkce Device_OnPacketArrival provede načtení packetu a jeho filtraci podle zadaných kritérií (buď implicitních, nebo explicitních zadaných na příkazové řádce).
    - Filtrace packetů provádí kontrolu podporovaných packetů (TCP, UDP, ICMP a ICMPv6), kontrolu na zadané porty (pokud nebyl žádný port zadán, tak filtr není aplikován), kontrolu na zadání explicitních filtrů (--tcp|-t, --udp|-u, --icmp).
    - Pokud veškeré filtrace projdou, tak se volá příslušná funkce pro výpis packetu.
    - Jakmile je packet vypsán, tak se provede kontrola v počítadlu packetů, zda již nedošlo k době, kdy se má aplikace ukončit.
- Services/Sniffer/Printer/Printer.cs – Logika výpisu jednotlivých druhů packetů.
    - Tento soubor obsahuje 2 hlavní funkce pro výpis packetů
        - PrintTcpAndUdpPackets
        - PrintICMPPackets
    - Obě funkce si prvně zjistí IP adresu odesílatele a příjemce a pokusí se je pomocí funkce PrinterHelper.TryGetHostname převést na FQDN. Pokud se nepodaří provést překlad, tak je ve výpisu ponechána IP adresa. Překlad se nemusí v některých případech zdařit.
    - Poté vypíšou řádek obsahující čas, doménový název, nebo IP adresa odesílatele, doménový název, nebo IP adresa příjemce.
        - U TCP a UDP packetů je zde ještě uveden zdrojový a cílový port.
        - U ICMP packetů je zde uveden pouze text, že se jedná o ICMP. A to z toho důvodu, že ICMP packety pracují na jiné vrstvě OSI modelu.
- Services/Sniffer/Printer/PrinterHelper.cs – Pomocné funkce pro výpis.
    - V tomto souboru je umístěna statická třída PrinterHelper, která obsahuje:
        - Funkce pro práci s DNS překladem.
            - Při implementaci DNS překladu bylo čerpáno z prvního projektu do předmětu IPK, přičemž implementace v tomto projektu ještě byla obohacena o časový limit dotazování na DNS a Cache dotazů na DNS.
            - Jak již bylo výše zmíněno, tak překlad DNS obsahuje časový limit. To je z důvodu, že dotaz na DNS servery může trvat déle. Během testování některé dotazy na DNS byly v řádech vteřin. A právě kvůli tomuto bylo hledání doménového názvu omezeno na 300ms.
            - Překlad DNS také obsahuje cache dotazů. Jedná se o jednoduchou slovníkovou strukturu, kde se jako klíč ukládá dotazovaná IP adresa (IPv4 i IPv6) a jako hodnota je přeložený název. Tato cache přinesla při testování pozitivní výsledky v případě velkého množství sledovaných packetů a současně

s tím zrychlení zpracování packetů. Pokud se však překlad nějaké IP adresy na FQDN nezdaří, tak se takový výsledek do cache neukládá.

- Funkce pro výpis obsahu packetu.
  - Nejobjemnější část celého výpisu.
  - Vypisuje data packetu v následujícím formátu:
    - `Počet bajtů bajty_v_hexa bajty_v_ascii`
  - Netisknutelné znaky (ASCII znaky, které nejsou v intervalu <33; 126>) jsou nahrazeny tečkou.
  - Každý řádek ve výpisu obsahuje 16 bajtů dat. Poté se provádí výpis na další řádek. Výpis každého řádku se provádí s doplněním bílých znaků, tak aby veškerá data byly přesně pod sebou a tím bylo zajištěno čitelnějšího výstupu.
  - V rámci jednoho řádku se také provádí každých 8 bajtů jak v hexa, tak v ascii části přidání jedné mezery. Tento výpis byl inspirován z aplikace Wireshark.

6. Testování

Aplikace byla testována na operačních systémech Linux a Windows 10. V případě Linuxu se jednalo o distribuci Ubuntu 18.04.4 LTS.

Při testování bylo využito přístupu, že na zařízení běžela jak aplikace Wireshark, tak aplikace ipk-sniffer a byly nastaveny stejné filtrační parametry. Poté bylo navozeno několik situací a bylo porovnáváno, zda výstupy souhlasí.

Byly provedeny následující testovací případy:

- ICMP ping
  - => google.dns
  - => widle.misha12.eu (osobní VPS se systémem Windows Server na koncovém bodu.)
  - => tux.misha12.eu (osobní VPS se systémem Debian 9.12)
- Připojení na vzdálenou plochu (Remote Desktop Protocol)
  - Tímto došlo k otestování UDP packetů.
- Otevření webové stránky (porty 80 a 443)
- Připojení na Samba server (port 445)
- Navázání SSH připojení (port 22)

Aplikace také byla testována tak, že bylo nastaveno velké množství očekávaných packetů a zachytávání bylo spuštěno několik hodin. Tak bylo zajištěno, že se v běžném v provozu neobjeví v aplikaci nějaké neočekávané situace. Výsledkem takového testování bylo 8 hodin sledování packetů, přibližně 500 MB výstupní soubor a žádný pád aplikace. Aplikace byla předčasně ukončena korektním způsobem a to pomocí klávesové zkratky CTRL+C.

7. Zdroje

What is .NET? An open-source developer platform.. .NET | Free. Cross-platform. Open Source. [online]. Dostupné z: https://dotnet.microsoft.com/learn/dotnet/what-is-dotnet

Cisco CCNA The OSI Model – CertificationKits.com. CertificationKits.com – Cisco Certification Lab Kits for CCNA, CCNP, CCIE, Security, Voice and more. [online]. Dostupné z: https://www.certificationkits.com/cisco-certification/cisco-ccna-640-802-exam-certification-guide/cisco-ccna-the-osi-model/

OSI Model - YUT-BMS. [online]. Dostupné z: https://sites.google.com/site/yutbms/osi-model

GitHub - chmorgan/sharppcap: Official repository - Fully managed, cross platform (Windows, Mac, Linux) .NET library for capturing packets. The world's leading software development platform · GitHub [online]. Copyright © 2020 GitHub, Inc. [cit. 03.05.2020]. Dostupné z: https://github.com/chmorgan/sharppcap

RFC 1122 - Requirements for Internet Hosts -- Communication Layers. [online]. Dostupné z: https://tools.ietf.org/html/rfc1122