Fakulta informačních technologií
Vysoké učení technické v Brně

IPK – Počítačová komunikace a sítě
Projekt 2 - ZETA - sniffer paketů

Šimon Sedláček (xsedla1h)

30/04/2020

Obsah

Abstrakt

Toto je dokumentace k druhému projektu do předmětu IPK. Ze zadání jsem si vybral variantu ZETA - sniffer paketů. Sniffer je implementován s použitím jazyka C/C++, knihovny libpcap a z malé části několika dalších standardních unixových síťových knihoven.

1 Kompilace a spuštění snifferu

Projekt se skládá z kořenového adresáře obsahujícího tento dokument, soubor README.md a soubor **Makefile** a z podadresáře **src/**, kde jsou uloženy veškeré zdrojové texty - main.cpp, resources.cpp, resources.hpp, sniffer.cpp, sniffer.hpp.

Pro kompilaci projektu proveďte příkaz **make** v kořenovém adresáři projektu. Po přeložení se v kořenovém adresáři objeví spustitelný soubor s názvem **ipk-sniffer**. Program je poté nutné spouštět s administrátorskými privilegii, jinak mu bude odepřen přístup k síťovým rozhraním.

Program lze spouštět s následujícími parametry:

```
./ipk-sniffer -i rozhraní [-p port] [--tcp|-t] [--udp|-u] [-n num]
```

kde:

- Parametr **-i** specifikuje rozhraní, na kterém se bude poslouchat.
- Parametr **-p** specifikuje port, na kterém se bude poslouchat (a to odchozí i příchozí komunikace).
- Parametry **--tcp/-t** a **--udp/-u** umožňují zobrazit buď pouze tcp nebo udp pakety.
- Parametr **-n** specifikuje, kolik paket se zobrazí. Implicitně se zobrazí pouze jeden paket, a při zadání hodnoty 0 bude program zachytávat pakety, dokud nedostane signál k ukončení (ctrl-c).

2 Popis implementace

2.1 Rozdělení do modulů

Sniffer je rozdělen do tří modulů:

- Modul **main.cpp** - tento modul je hlavním modulem programu.
- Modul **resources.cpp** - tento modul obsahuje různé funkce pro alokaci a dealokaci zdrojů, parsování vstupních argumentů programu, definice návratových kódů. Pro samotné poslouchání nemá relevantní význam.
- Modul **sniffer.cpp** - toto je hlavní modul snifferu, jsou zde situovány veškeré funkce pro zachytávání a zpracování zachycených paketů.

2.2 Příprava zdrojů a zahájení poslechu na daném rozhraní

Při spuštění programu jsou nejprve zpracovány argumenty programu - skrze argument **-i** je nutné specifikovat jméno síťového rozhraní. Pokud tento argument není specifikován, vypíše se pouze seznam aktivních rozhraní a program se ukončí.

Pokud je specifikováno rozhraní, řízení je předáno modulu **sniffer.cpp**, konkrétně funkci sniff_init(). Tato funkce v prvé řadě specifikované rozhraní otevře pro poslech a zkontroluje, zda se jedná o ethernetové rozhraní [Car10].

Po otevření rozhraní je také zkontrolováno, zda se jedné o ethernetové rozhraní, nebo zda zda vybrané rozhraní podporuje Linux cooked-mode capture.

Následně je vytvořen filtr [20] na zachytávané pakety - a to podle toho, zda bylo snifferu předáno číslo portu, na kterém se má poslouchat, a zda byl programu předán jeden z argumentů **-t** nebo **-u**. Vytvořený filtr je zkomiplován a

předán funkci pcap_setfilter(). Tento výsledný filtr nám v další fázi zpracování zachycených paket poměrně zjednoduší práci - díky němu budou zachytávány pouze IPv4/IPv6 TCP nebo UDP pakety.

Řízení je dále předáno funkci sniff(), která vstoupením do smyčky, kde se volá knihovní funkce pcap_loop() zahájí proces zachytávání paketů. Jak už bylo zmíněno výše, jendím z parametrů funkce pcap_loop() je celkový limit počtu paketů, který chceme zachytit. Tato hodnota je implicitně nastavena na 1, avšak při specifikování hodnoty 0 se budou pakety zachytávat do té doby, než bude provádění programu přerušeno zvenčí.

2.3 Zpracování zachycených paketů

Ještě předtím, než začnu popisovat samotný proces zpracování jednotlivých paketů, je třeba říci, že v průběhu samotného zpracování jsou různé potřebné informace jako ip adresy a čísla portů pro každý packet ukládány do struktury *struct pck_data packet_data*, aby pak při vypisování obsahu a informací o paketu byly jednoduše k dispozici. Definice této struktury je v souboru sniffer.hpp.

Jedním z požadovaných parametrů funkce pcap_loop() je callback funkce, která každý zachycený paket zpracuje. Tato funkce se ve zdrojovém textu jmenuje process_packet(). Vzhledem k tomu, že zachytáváme pouze ethernetové pakety, můžeme pro všechny zachycené pakety zvolit jednotný přístup. Pokud se chceme dostat k obsahu hlavičky ethernetového rámce, můžeme si vytvořit proměnnou uchovávající ukazatel na strukturu, která Ethernet II hlavičku popisuje. Struktura, kterou jsem použil, je definována v hlavičkovém souboru net/ethernet.h. Do tohoto ukazatele můžeme poté přiřadit hodnotu přetypovaného ukazatele na celý zpracovávaný paket - přístup k prvkům naší struktury je pak ekvivalentní přístupu k polím hlavičky ethernetového rámce. Toto postupné "rozbalování"paketu je vzhledem ke způsobu konstrukce síťových paketů možné aplikovat i dále pro vyšší vrstvy.

Z této hlavičky zjistíme typ následujícího protokolu (IPv4 nebo IPv6) a zavoláme příslušnou funkci. Obě funkce process_ipv4() i process_ipv6() pracují na stejném principu (narozdíl od process_packet() však již dostávají původní packet "useknutý"- bez ethernetové hlavičky). Jsou vytvořeny ukazatele na struktury popisující IPv4 a IPv6 hlavičky a je do nich přiřazen ukazatel na data paketu. Do struktury packet_data() je uložena výchozí a cílová IP adresa a je zde nastaven flag indikující, zda zpracovávaný paket náležel protokolu IPv4 nebo IPv6. Z hlavičky IP protokolu je zjištěn typ následujícího protokolu (TCP nebo UDP) a packet zkrácený o IP hlavičku je předán funkci na zpracování TCP nebo případně UDP paketu.

Zpracování UDP a TCP paketů mají na starost funkce process_tcp() a process_udp(). Tyto funkce fungují na stejném principu jako funkce na zpracování IP paketů s tím rozdílem, že ve struktuře packet_data nastavují flag tcp_udp podle typu paketu, a také číslo výchozího a cílového portu.[81][80]

2.4 Výpis informací o paketu

Poté, co jsou sesbírány veškeré informace o aktuálním paketu, je struktura packet_data předána funkci print_current_packet_data(). Tato funkce převede sesbíraná data o paketu do příslušných formátů a poté tyto informace spolu s celkovým obsahem paketu vypíše na výstup.

2.4.1 Překlad IP adres na doménová jména

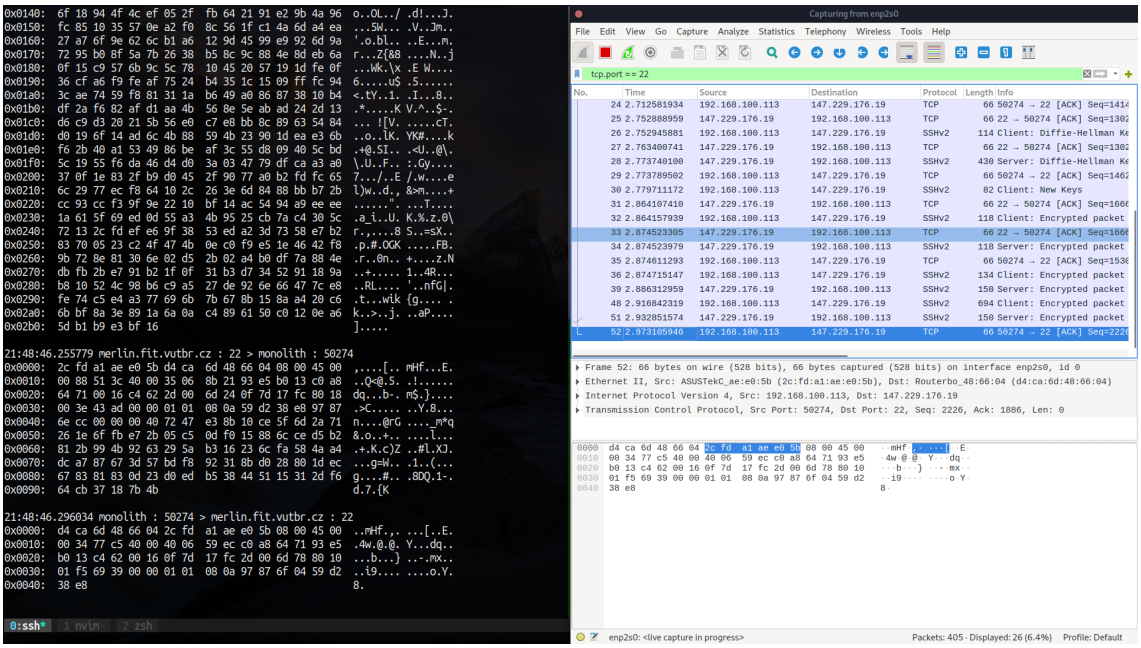
Při implementaci překladu jednotlivých ip adres na doménová jména jsem narazil na problém, kdy samotný DNS dotaz generoval pakety, u kterých se pak sniffer snažil zjistit doménové jméno. Takto se program dostal do nekonečné smyčky posílání dotazů na dotazy a úplně tím zastínil ostatní příchozí pakety.

Tento problém jsem nejprve chtěl vyřešit pouze podmíněným překladem s parametrem **-DDO_NOT_TRANSLATE_IP**, kdy se jednoduše doménová jména nepřekládají, avšak nakonec jsem zvolil jiný postup - který jsem však implementoval pouze pro IPv4 adresy, pro IPv6 adresy jsem to již nestihl (pro ty se vždy vypíše původní IPv6 adresa).

Výsledky DNS queries se pro IPv4 adresy ukládají do asociativní paměti, která se vždy před odesláním dalšího dotazu zkontroluje. Pokud je pro danou IP adresu již v paměti jméno, dotaz se neposílá a použije se uložené doménové jméno. Kompilace s parametrem **-DDO_NOT_TRANSLATE_IP** (je zakomentován v Makefile) překlad doménových jmen vypne úplně.

3 Testování

Projekt jsem testoval s pomocí ssh a nástroje curl. Tyto nástroje mi umožnily začít komunikaci na určitých konkrétních portech (například port 22 v případě ssh), případně skrze tyto porty posílat jen malé kontrolované množství paketů. Daný port jsem poté poslouchal jednak mnou implementovaným snifferem a dále s pomocí programů Wireshark a tcpdump. Tyto dva druhé nástroje mi po nastavení příslušných filtrů sloužily jako reference při kontrolování výstupu projektu.



Obr. 1: Porovnávání výstupů projektu a analyzátoru Wireshark

Reference

[20] *Manpage of PCAP-FILTER*. The Tcpdump Group. Led. 2020.

[80] *User Datagram Protocol*. RFC 768. Srp. 1980. DOI: 10.17487/RFC0768. URL: https://rfc-editor.org/rfc/rfc768.txt.

[81] *Transmission Control Protocol*. RFC 793. Zář. 1981. DOI: 10.17487/RFC0793. URL: https://rfc-editor.org/rfc/rfc793.txt.

[Car10] Tim Carstens. *Programming with pcap*. 2010. URL: https://www.tcpdump.org/pcap.html.