Projektová dokumentace

Packet sniffer IPK

Jakub Kuzník
xkuzni04

Úvod

Pro řešení projektu jsem zvolil jazyk C. Snažil jsem se o minimalistické řešení s důrazem na přehlednost kódu a dobrým členěním zdrojového kódu do jednotlivých souborů. Packet sniffer podporuje zobrazování Ethernet rámců s protokoly jimiž jsou ARP, ICMP, TCP a UDP a to jak pro ipv4, tak i ipv6. Rád bych ještě zmínil, že jsem se zdrojové kódy snažil vyprodukovat bez dívání na řešení jiných, ale snažil jsem se čerpat hlavně z manuálových stránek linuxu a zdrojových kódu knihoven.

Teorie

Jelikož jsem studoval obor počítačové sítě na střední škole, tak jsem nemusel příliš studovat některé teoretické základy, ale vycházel jsem především ze svých znalostí. Některé detaily, které jsem potřeboval zjistit jsem často nacházel v knize COMPUTER NETWORKING A Top-Down Approach, či přímo v dokumentaci daných knihoven.

ISO/OSI

V rámci iso/osi modelu zpracovává sniffer data na druhé síťové vrstvě tedy rámce. Ta v sobě však mohou mít zabalené hlavičky protokolů vyšších vrstev jako je ip, icmp a ipv6 na třetí síťové vrstvě a TCP a UDP na 4 vrstvě. Program sice pracuje s rámci na druhé vrstvě, ale nahlíží do hlaviček protokolů vyších vrstev a vybirá z nich relevantní informace.

Ethernet hlavička

Ethernet hlavička poskytuje na prvních 12 Bytech zdrojovou a cílovou MAC adresu a od 14 Bytu se můžou vyskytovat data, či jiná hlavička protokolu vyšší vrstvy. Pro tyto účely jsou v programu konstanty **ETH_HEAD** ta udává velikost hlavičky, neboli místo kde začíná hlavička jiného protokolu. [2,3]

IP hlavička

Ip hlavičky ipv4, ipv6 poskytují informace o zdrojové a cílové ip address, ale také určují protokol 4 vrstvy. Obě hlavičky mají pole version, podle kterého jsem rozpoznal zda-li budu pracovat s verzií 4 nebo 6. U ipv4 se tato informace nachází v sekci Upper-layer protocol a u ipv6 v sekci next header. Důležitá je také informace o tom jak jsou hlavičky velké na to slouží makra **IP_HEAD** a **IPV6_HEAD**. v místě kde končí ip hlavička totiž začíná hlavička protokolu vyšší vrstvy. [3]. Je dobré se zmínit že se z pole next-header, nebo protocol pro ipv4 dalo zjistit zda-li je packet icmp, i přesto že se nejedná o protokol vyšší vrstvy.

TCP a UDP hlavička

Protokol Tcp obsahuje spoustu informací, sniffer zobrazuje některé základní a především se zajímá o číslo zdrojového a cílového portu. Na jejichž základě se pakety mohou filtrovat. Program také vypisuje acknowledgement number a sequence number ty slouží k zajištění spolehlivosti tcp a ověření zda-li dorazily ty pakety, které měly. Velikosti tcp a udp hlaviček se program nezabírá a nijak je neověřuje, jelikož se do žádných jiných hlaviček po TCP či UDP nedívá. [4]

Udp hlavička poskytuje výrazně méně informací než tcp, zobrazují pouze informace o zdrojovém a cílovém portu. [5]

Typy arp zpráv

Protokol ARP jenž získává informace o tom jaká MAC adresa se vztahuje k jaké IP adrese a funguje na druhé vrstvě. Packet sniffer zobrazuje jaký typ arp zprávy je daný packet. To může být užitečné v odhalování potencionálních arp útoků. Dále rozlišujeme zda-li je arp dotaz mířen na ipv4 nebo ipv6, to vše se nachází v hlavičce arp rámce. [6]

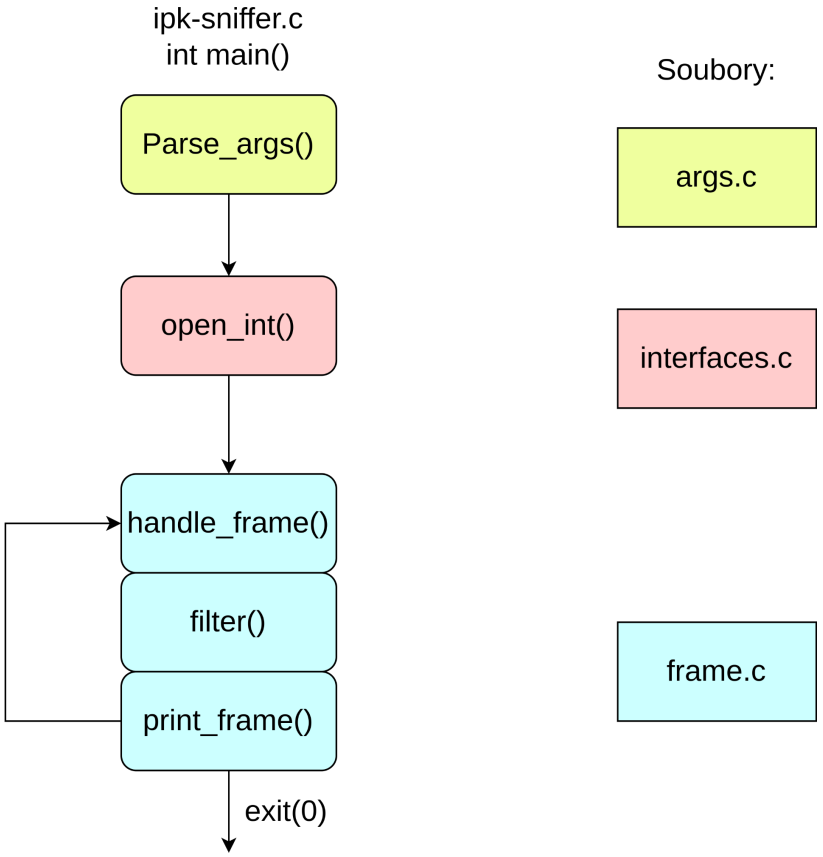
Typy icmp zpráv

ICMP protokol se nejčastěji používá na error reporting, zjistíme pomocí něj zda-li je zařízení resp rozhraní s danou ip adresou dostupné. Stejně jako u arp jsem se u icmp rozhodl zobrazit jaký typ zprávy je daný rámec a danou informaci jsem našel v hlavičce icmp rámce. Zajímavé je že icmp hlavička velmi podobná hlavičce ip. [7]

Schéma programu

Na následujícím diagramu je zobrazena základní zjednodušená algoritmická logika programu a soubory, které za dané chování zodpovídají. Princip je jednoduchý, program zpracuje argumenty, poté otevře interface, ze kterého bude “krást” rámce. A pak pakety, které projdou filtrem zobrazuje na výstup. To dělá dokud nesplní parametr -n, nebo nedojde

k chybě. Pro filtrování paketů a parsování argumentů nebyly využity žádné dostupné speciální knihovny a to kvůli jednoduchosti a lepší kontrole.



Souborová struktura

Program je napsán v jazyce C. Každý soubor s příponou .c má svůj hlavičkový soubor se stejným jménem. Program má 4 hlavní c soubory, kde má každý svojí jedinečnou zodpovědnost. **ipk-sniffer.c** je soubor obsahující funkci **main().** Volá veškeré funkce z ostatních souborů. Jedinou funkcionalitu, kterou přidává je ošetření signálu z operačního systému. **args.c** je soubor ve, kterém se zpracují a vyhodnotí vstupní parametry. **interfaces.c** se stará o funkce, které se provádí nad síťovými rozhraními. Nejdůležitější je soubor **frame.c,** ten zajišťuje veškerou práci se síťovými rámci, tedy jejich parsing, filtrování, vyhodnocení a především správné zobrazení. Řešení dále obsahuje dokumentaci v souboru **manual.pdf** a soubor **README.md**. Níže je uvedena tabulka souborů

Jméno souboru	Popis
ipk-sniffer.c	Soubor s funkci main()
args.c	Má zodpovědnost za zpracování vstupních parametrů.
interfaces.c	Soubor poskytuje funkce k práci se síťovými rozhraními.
frame.c	Soubor poskytuje veškeré operace nad síťovými rámci, včetně jejich filtrace a zobrazení.

ipk-sniffer.h, interfaces.h, args.h, frame.h	Hlavičkové soubory pro stejnojmenné soubory s příponou .c
Makefile	Makefile projektu
manual.pdf	Tento soubor, obsahuje dokumentaci projektu.
README.md	Readme obsahující základní popis a instrukce pro spuštění. .

Knihovny

Přehled využívaných knihoven se nachází v souboru **ipk-sniffer.h**. Veškeré informace o jednotlivých knihovnách jsou dostupné v rámci zdrojových kódu daných knihoven či manuálových stránek. [8]

Jméno knihovny	Využití v projektu
pcap.h	Funkce pro načítání rámce z rozhraní. Pomocí **pcap_next()** se vždy načte jeden rámec a ten se dále zpracuje. Dále struktura **pcap_header**, z nichž jednoduše zjistíme základní informace o daném rámci. Funkce, která najde veškeré síťové rozhraní **pcap_findalldevs()** a vrátí jejich seznam, či **pcap_open_live()** ta otevře zadané rozhraní. Pomocí **pcap_datalink()** zjistíme zda-li dané rozhraní podporuje rámce typu ETHERNET.
arpa/inet.h	Poskytuje datové typy pro ip adresy, ale především funkce **ntohs()** či **ntohl()** ty zamezí problémům v případě odlišných počítačových architektur (little endian, big endian)
netinet/if_ether.h	Obsahuje datovou strukturu **ether_header** pro práci s hlavičkou ethernetových rámců a makra, která pomáhají s práci s daty obsaženými v dané hlavičce.
netinet/ip_icmp.h	Obsahuje datovou strukturu **icmp** pro práci s hlavičkou icmp a makra, která pomáhají s práci s daty obsaženými v dané hlavičce. Například makra pro rozpoznání typu zprávy.
netinet/ip6.h	Obsahuje datovou strukturu **ipv6** pro práci s hlavičkou ipv6
netinet/tcp.h	Obsahuje datovou strukturu **tcphdr** pro práci s TCP hlavičkou
netinet/udp.h	Obsahuje datovou strukturu **udphdr** pro práci s UDP hlavičkou
netinet/arp.h	Obsahuje datovou strukturu **arphdr** pro práci s hlavičkou arp a makra, která pomáhají s práci s daty obsaženými v dané hlavičce. Například makra pro rozpoznání typu zprávy.

Testování

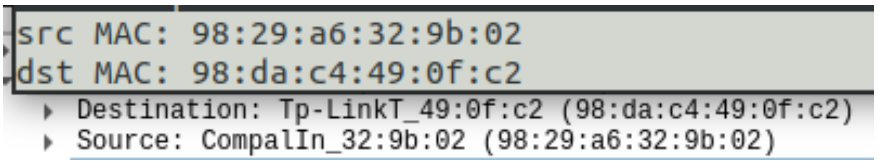
Během testování jsem zároveň testoval open-source software Wireshark. Data, která se mi podařilo načíst pomocí funkce **pcap_next()** jsme zkontroloval pomocí debugeru a pak porovnával zda-li jak můj tak wireshark výstup sedí. Veškerou síťovou komunikaci jsem testoval na živém provozu, především na VUT kolejnet sítí, která je především bohatá na ipv6 a arp rámce. Provoz byl testoval na různých rozhraních a výpisy ve wiresharku vždy odpovídaly výpisům packet snifferu, s tou výjimkou, že wireshark znal nějaké protokoly navíc, které packet sniffer přeskočil a nezobrazoval jako např LLDP. Veškeré údaje byly testovaný napříč všemi protokoly, které dokáži zobrazit. Během testování jsem rovněž kladl důraz na co jak nejrozmanitější kombinace vstupních parametrů

Testování základních údajů - v rámci testování základních údajů jsem si stejný rámec zobrazil jak v programu packet sniffer tak v programu wireshark a porovnaval tyto důležité údaje:

- čas (timestamp) - dole je čas z wiresharku a nahoře z packet snifferu, jde vidět, že jsou časy totožné, ale formát je trochu rozdílný. Tento údaj je velmi užitečný k identifikaci konkrétního paketu, vzhledem k testování na velkém provozu.

- Velikost rámce

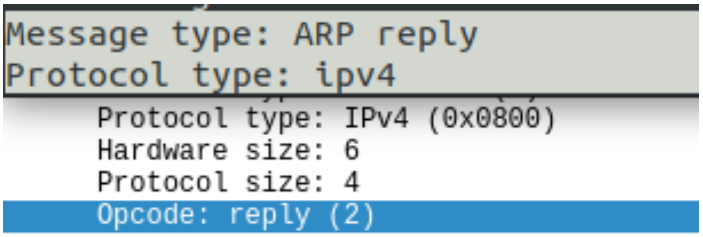
- Zdrojová a cílová MAC adresa



ARP test

V rámci arp take bylo nutné otestovat zda-li funguje korektně i pro ipv6, tedy stejně jako pro ipv4.

- Protokol, typ arp zprávy - v rámci protokolu arp bylo nutné otestovat zda-li umí packet sniffer dobře rozpoznat typ arp zprávy a ip verzi.



ICMP test

V rámci icmp take bylo nutné otestovat zda-li funguje korektně i pro ipv6, tedy stejně jako pro ipv4.

- code a message type - jak je vidět na výpisu níže tak se v rámci icmp navíc zobrazuje typ zprávy a code, které odpovídají těm ve wiresharku.

```
Message type: ICMP Echo Reply
Code: 0
▾ Internet Control Message Protocol
    Type: 0 (Echo (ping) reply)
    Code: 0
```

IPv4 IPv6 test

Veškeré protokoly, které podporují musely fungovat jak pro ipv6 tak pro ipv4.

- ipv6 adresa

```
src IP: fe80::9ada:c4ff:fe49:fc2
dst IP: fe80::545f:c652:3a54:521b
Source Address: fe80::9ada:c4ff:fe49:fc2
Destination Address: fe80::545f:c652:3a54:521b
```

- ipv4 adresa

```
src IP: 8.8.8.8
dst IP: 192.168.1.107
Source Address: 8.8.8.8
Destination Address: 192.168.1.107
```

- tcp informace

```
src IP: 192.168.1.109
dst IP: 162.159.136.234
src port: 50678
dst port: 443
seq num raw: 2693703249
ack num raw: 3874264993
Protocol: TCP
▾ Transmission Control Protocol, Src Port: 50678, Dst Port: 443
    Source Port: 50678
    Destination Port: 443
    [Stream index: 1]
    [TCP Segment Len: 0]
    Sequence Number: 1    (relative sequence number)
    Sequence Number (raw): 2693703249
    [Next Sequence Number: 1    (relative sequence number)]
    Acknowledgment Number: 2400    (relative ack number)
    Acknowledgment number (raw): 3874264993
```

- udp informace

```
src port: 62823
dst port: 1900
Protocol: UDP
▾ User Datagram Protocol, Src Port: 62823, Dst Port: 1900
    Source Port: 62823
    Destination Port: 1900
```

Spuštění programu

./ipk-sniffer [-i rozhraní | --interface rozhraní] {-p port} {[--tcp|-t] [--udp|-u] [--arp] [--icmp] } {-n num} {-h | –help}

V případě, že uživatel nezadá rozhraní, nebo spustí program bez parametru, zobrazí se seznam dostupných rozhraní na stdout. Pokud se nezadá žádný omezující protokol budou se vypisovat veškeré možné. Návod jak zkompilovat program a jednotlivé příklady spuštění jsou dostupné v README.md

- -i –interface
    - Rozhraní na, kterém se budou brát pakety
- -h –help
    - Vypíše nápovědu
- -p port
    - kde port je přirozené číslo označující číslo port, je omezení pro TCP a UDP pakety. Na výstupu se budou zobrazovat jenom pakety, které mají v source, nebo destination port dané číslo portu.
- -t –tcp
    - Zapnutí protokolu tcp - pakety tcp budou zobrazeny a nebudou přeskočeny
- -u –udp
    - Zapnutí protokolu udp - pakety udp budou zobrazeny a nebudou přeskočeny
- –arp
    - Zapnutí protokolu arp - pakety arp budou zobrazeny a nebudou přeskočeny
- –icmp
    - Zapnutí protokolu arp - pakety arp budou zobrazeny a nebudou přeskočeny

Odchylka od zadání

V rámci dodatku na foru bylo řečeno že pokud se např zadá parametr -p 443 bez parametru -t -u tak, se zobrazí jak UDP, tak TCP rámce i přesto, že tato možnost nebyla zvolená.
tedy v případě **./ipk-sniffer -i eth0 –arp -p 443** se mi zobrazí pouze arp rámce a žádné UDP či TCP i přesto, že by měly

Zdroje:

[1] Ethernet frame [online]. [cit 22.4.2022] Dostupné z: https://en.wikipedia.org/wiki/ethernet_frame odstavec ethernet II
[2] Kurose, James F. Computer networking : a top-down approach / James F. Kurose, Keith W. Ross.—6th ed. p. cm. Includes bibliographical references and index. ISBN-13: 978-0-13-285620-1 ISBN-10: 0-13-285620-4 4.4.2 ipv4 Addresing
[3] Kurose, James F. Computer networking : a top-down approach / James F. Kurose, Keith W. Ross.—6th ed. p. cm. Includes bibliographical references and index. ISBN-13: 978-0-13-285620-1 ISBN-10: 0-13-285620-4 4.4.4 ipv6
[4] Kurose, James F. Computer networking : a top-down approach / James F. Kurose, Keith W. Ross.—6th ed. p. cm. Includes bibliographical references and index. ISBN-13: 978-0-13-285620-1 ISBN-10: 0-13-285620-4 3.5.2 TCP Segment Structure
[5] Kurose, James F. Computer networking : a top-down approach / James F. Kurose, Keith W. Ross.—6th ed. p. cm. Includes bibliographical references and index. ISBN-13: 978-0-13-285620-1 ISBN-10: 0-13-285620-4 3.3.1 UDP Segment Structure
[6] Kurose, James F. Computer networking : a top-down approach / James F. Kurose, Keith W. Ross.—6th ed. p. cm. Includes bibliographical references and index. ISBN-13: 978-0-13-285620-1 ISBN-10: 0-13-285620-4 5.4.1 Link-Layer Addressing and ARP
[7] Kurose, James F. Computer networking : a top-down approach / James F. Kurose, Keith W. Ross.—6th ed. p. cm. Includes bibliographical references and index. ISBN-13: 978-0-13-285620-1 ISBN-10: 0-13-285620-4 4.4.3 Internet Control Message Protocol.
[8] <netinet/>. *The Open Group Publications Catalog* [online]. Copyright © 1997 The Open Group [cit. 22.04.2022]. Dostupné z: https://pubs.opengroup.org/onlinepubs/7908799/xns/netinetin.h.html