VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Počítačové komunikace a sítě – 2. projekt

Varianta ZETA: Sniffer paketů

25. dubna 2021

Jozef Makiš

Obsah

1 Úvod

Sniffer spracováva argumenty príkazoveho riadku. Sniffer paketov podporuje zachytávanie paketov TCP, UDP, ARP alebo ICMP. Tieto pakety môžu mať protokol IPv4 alebo IPv6. Knižnica, ktorá umožňuje snifferu zachytávať pakety je *pcap*. Preklad snifferu sa koná za pomoci Makefilu, pomocou príkazu *make*. Na základe argumentov dokáže zachytiť a vypísať na štandardný výstup informácie o zachytenom pakete z daného rozhrania. V prípade, že program zachytí paket, ktorý nespĺňa požiadavky tak ho nevypisuje na štandardný výstup.

1.1 Spustenie programu

Sniffer sa prekladá pomocou príkazu *make*. Pre funkčný výpis dostupných rozhrani sa sniffer spúšťa: *./ipk-sniffer alebo ./ipk-sniffe -i*. Po výbere rozhrania je možne pristúpiť k sputeniu zachytávania príkazov pr. *./ipk-sniffer -i ens33*. Sniffer umožňuje filtrovať pakety na základe argumentu -p, za ktorým následuje číslo portu na ktorom chceme zachytávať pakety. Pomocou parametru -n je možne udať počet zachytených paketov. Pomocou parametrou -t, -u, –tcp, –udp, –arp alebo –icmp je možné si navoliť pakety, ktoré chceme zachytávať. Tieto argumenty je možné kombinovať medzi sebou.

2 Sniffer paketov

2.1 Popis

Paket sniffer alebo analyzátor paketov je softvér, ktorý slúži na sledovanie internetovej premávky. Tento analyzátor paketov spracováva toky dát paketov, ktoré prúduia medzi počítačmi na sieti. Pakety, ktoré spracovávajú majú svojho príjmatela ako i odosielateľa. V prípade, že chceme nahliadnuť do paketu hlbšie je potrebné byť v takzvanom „promiscuous“ mode. Zo siete je možné vďaka filtrácií získať iba pakety, ktoré nám vyhojú.

2.2 Využitie

Na čo sa tento program používa ? Pomocou snifferu je možné vyhľadávať problémy siete, analyzovať tok dát alebo odhaliť možného narušiteľa na sieti. Ak narušitel používa sniffer, nazýva sa to pasívny útok. Sniffre využívajú pre zisk rôznych citlivých dát ako sú hesla a údaje k bankovým účtom ako je uvedené v [4]. Použitie ma aj v antiviruse, kedy môže odhaliť možné škodlivé data.

3 Popis implementácie

3.1 Jazyk implementácie

Sniffer je implementovaný v jazyku C++. Vďaka knižniciam, ktoré tento jazyk ponúka bolo možné implementovať jednoduché zachytávanie paketov. To to zachytávanie paketov je popísané v dokumentácií knižnice viz [5].

3.2 Spracovanie argumentov

Spracovanie argumentov programu zabezpečuje knižnica *getopt* viz [2]. Hlavná funkcia spracovania argumentov je *getopt_long*, ktorá spracováva argumenty príkazovej riadky, ktoré môžu byť aj napríklad celé slová. Slová s dvomi čiarkami, ktoré sa môžu vyskytnúť ako argumenty príkazovej riadky sú spracované vďaka ukazatelu na pole s názvami dlhých možností príkazovej riadky. Po spracovaní argumentov sa nastavia vlajky programu pre pakety, ktoré chceme zachytávať. Poprípade počet paketov alebo filter pre pakety.

3.3 Zachytávanie paketov

Potom, ako program príjme rozhranie, na ktorom má zachytávať pakety sa overí jeho prítomnosť pomocou funkcie *findalldevs*. Táto funkcia sa využíva aj pri získaní všetkých prítomných zariadení, na ktorých sa dajú zachytávať pakety. Potom, ako sa overí prítomnosť zariadenia následuje získanie sieťovej masky, ktorá je potrebná pri aplikácii filtra. Sniffovanie paketov začína otvorením sniffovacieho rozhrania vďaka funkcii *pcap_open_line*, ktorá ma nastavenú dĺžku čítania na 1000 ms. Pred tým ako sa aplikujé filter je najprv potrebné ho preložiť z načítaného reťazca do jazyka, ktorému rozumie program. Po tomto všetkom je možné prejsť na spracovanie samotných paketov. K tomuto je potrebná funkcia *pcap_loop*, ktorej počet spracovaní je daný buď argumentom programu alebo jedna. Dôležitým parametrom je call back funkcia, ktorá je vysvetlená v následujucej sekcií.

3.4 Callback funkcia

Funkcia *pcap_loop* v programe potrebuje vedieť, čo má robiť počas toho, ako spracováva zachytený paket. Pre určenie toho, čo má sniffer vykonávať je vytvorená funkcia *my_callback*. Callback funkcia je volaná s tromi parametrami. Prvým argumentom je užívateľský argument, ktorý sa nevyužíva. Druhým argumentom je ukazateľ na parametre paketu ako je čas a dĺžka paketu. Posledným argumentom je samotný paket. Podľa dokumentácie viz. [5] samotné dáta a ukazateľ na ich parametre niesu uvoľnované. Preto ich je potrebné prekopírovať, ale to nieje potrebné, pretože po volaní callbacku s danými datami už sniffer nepracuje.

Funkcia začína tým, že sa ako prvé určí hlavička IPv4, z ktorej sa zisťuje verzia a protokol paketu. Inicializácia ethernetovej hlačiky je potrebná, pretože zdojová adresa a cieľová adresa sa používa prí výpise ARP paketu. Inicializácia dĺžky IP hlavičky je potrebné, pretože potrebujeme jej dĺžku v bytoch pri ďalšej inicializácií hlavičiek TCP a UDP protokolov, ako sa uvádza v [1].

- **Výpis ARP paketov**: V prípade, že bol v ethernetovej hlavičke rozpoznaný ARP paket vypíše sa aktuálny čas, kedy bol paket zachytený. Zachytené adresy je potrebné prekopírovať do štruktúry *ether_addr* aby mohli byť následne prekonvertované ako hexadecimálne reťazce funkciou *ether_ntoa*. Obsah paketu je vypísaný funkciou *print_packet*.
- **Výpis TCP paketov a UDP paketov**: U TCP a UDP je ten postup približne rovnaký. Pri IPv4 sa adresa prekopíruje do štruktúry *sockaddr_in*, u IPv6 sa prekopíruje do štruktúry *sockaddr_in6*. IPv4 sa vypíše funkciou *inet_ntoa*. Keďže funkcia *inet_ntoa* nestačí na výpis IPv6 adries použil som jej obdobnú variantu *inet_ntop* uvedenú v [3]. Porty IPv4 ako i IPv6 sa vypisujú z hlavičiek paketov pomocou funkcie *ntohs*.
- **Výpis ICMP paketov**: Princíp výpisu ICMP paketov je podobný, U ICMP paketov sa nevypisuje ich port, ale iba zdrojová a cieľová adresa.

3.5 Vlastná funkcia pre výpis času

Funkcia sa využíva pri spracovaní a výpise paketu na štandardný výstup. Zabezpečuje výpis času zachytenia paketu vo formáte RFC3339. Čas je vypisovaný v milisekundách na dve desatinné miesta bez zaokrúhlenia. Čas paketu sa získava z parametru *callback* funkcie, ktorý ho obsahuje.

3.6 Vlastná funkcia pre výpis paketov

Funkcii je predaný v parametroch paket a jeho dĺžka. Obsah paketu sa vypisuje po 16 hexadecimálnych znakov na jeden riadok, po ktorom nasleduje ich obsah v ASCII podobe, kedy znaky mimo intervalu 32 až 127 sú vypisované ako bodky. Cyklus v tejto funkcií prejde cez celý obsah paketu, kedy si do pola o dĺžky 16 ukladá znaky po 16. To to sa deje vďaka operácií modulo. Posledné znaky sa vypisujú zvlášť na základe podmienky. V podmienke sa vypočíta počet znakov posledného riadku. Na základe toho sa hexadecimálnej časti doplnia prázdne znaky do konca hexadecimáleho riadku. ASCII časť sa vypíše podľa obahu posledného riadku.

4 Testovanie

Svoje riešenie som porovnal s open-source nástrojom. Vybral som si Wireshark. Z testovania som vybral konkrétne ICMP, ARP, ICMPv6 a UDP IPv6 pakety. Výsledok som poroval a nenašiel som žiadne chyby alebo rozdiely vo výstupe snifferu. Obrázky z testovania sa nachádzajú nižšie.

Použítá literatúra

[1] CASADO, M.: The Sniffer's Guide to Raw Traffic. Dostupné z: `http://yuba.stanford.edu/~casado/pcap/section1.html`

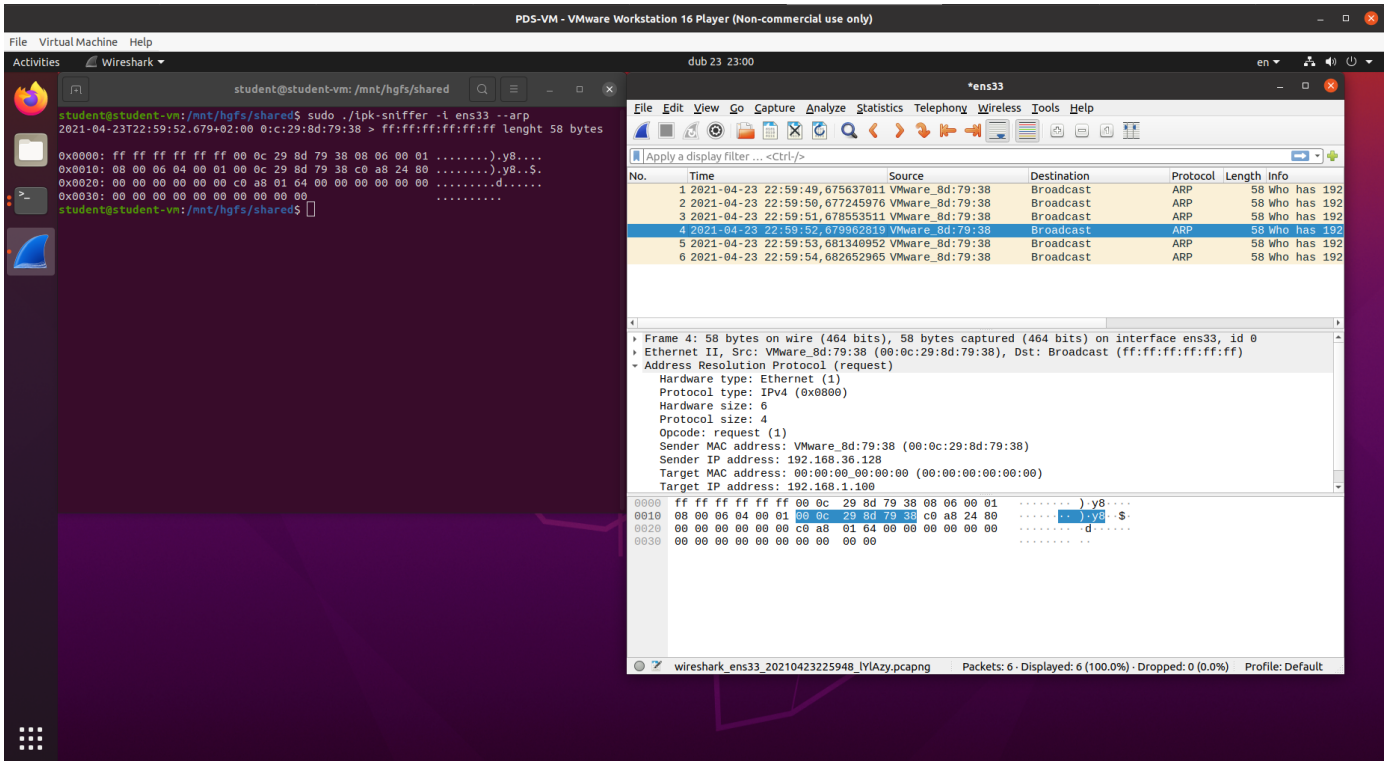
[2] DIE: *Manual page of linux*. Dostupné z: `https://linux.die.net/man/3/getopt_long`

[3] KERRISK, M.: *Linux manual page*. Dostupné z: `https://man7.org/linux/man-pages/man3/inet_ntop.3.html`

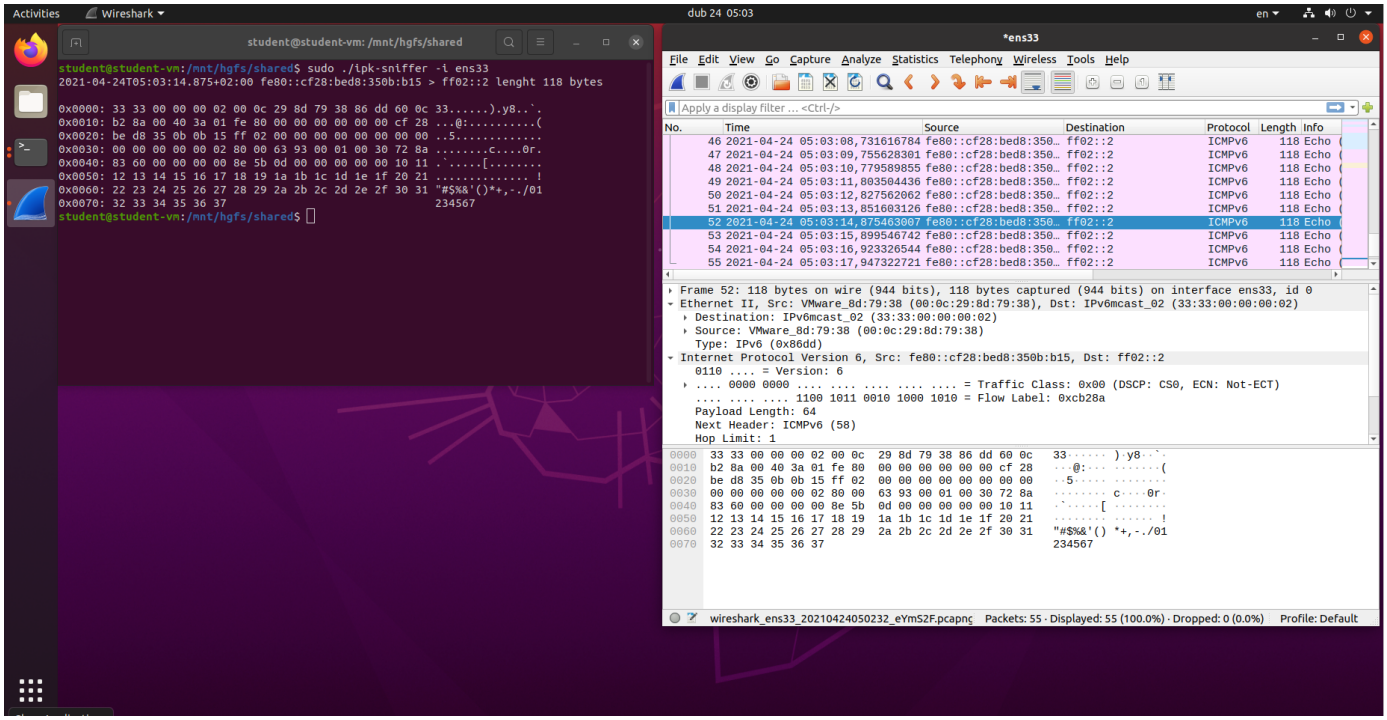
[4] OREBAUGH, A.: *Ethereal Packet Sniffing*. Rockland, MA 02370: Syngress Publishing, 2004, ISBN 1932266828.

[5] Van JACOBSON, S. M., Craig LERES: *Manual page of PCAP*. University of California, Berkeley, CA. Dostupné z: `http://www.tcpdump.org/pcap.html`

5 Obrázky z testovania



Obrázek 1: ARP paket



Obrázek 2: ICMPv6 paket