VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA INFORMAČNÝCH TECHNOLÓGIÍ

IPK projekt
SNIFFER PACKETOV

Adrián Horváth

xhorva14

24.4.2022

OBSAH

ÚVOD

Toto je dokumentácia k druhému projektu z predmetu IPK. Zadanie ZETA, analyzátor (sniffer) sieťových paketov. Sniffer dokáže zachytávať IPv4(ICMPv4, TCP, UDP), IPv6(ICMPv6) a ARP pakety, v prípade potreby len tie, ktoré prechádzajú zadaným portom. Jeho výstupom je vypísanie zachyteného paketu, čas jeho zachytenia vo formáte RFC3339 a oboch adries(v prípade ARP paketov MAC adries) a portov(v prípade TCP a UDP paketov), ktorými paket prešiel.

IMPLEMENTÁCIA

Projekt je implementovaný v jazyku C++. Základ projektu je postavený na knižnici `pcap`, ktorej funkcie sa starajú o všetku prácu s paketmi a ich odchytávaním.

VSTUPNÉ ARGUMENTY

Sniffer je možné spúšťať s viacerými vstupnými argumentami(prepínačmi). Na ich poradí nezáleží.

Prvým z nich je prepínač `-i/--interface` *`rozhranie`*, je to povinný argument ktorý programu hovorí o tom na akom rozhraní sa budú analyzovať pakety. Ak tento argument nie je zadaný alebo je zadaný bez rozhrania program vypíše všetky dostupné rozhrania na aktuálnom zariadení a skončí. Rovnako sa program chová ak nie je zadaný žiadny vstupný argument.

Argument `-p` *`port`* slúži na filtrovanie paketov podľa portu. Tento argument nie je povinný a ak nebude zadaný, analyzované budú pakety na všetkých portoch na danom zariadení.

Argument `-n` *`pocet_paketov`* určuje počet paketov, ktoré budú vypísane na `stdout`, opäť nie je povinný a ak nebude uvedený vypíše sa jeden paket.

Nasledujúca skupina prepínačov slúži na nastavenie filtra paketov podľa ich protokolov. Ak nie je zadaný žiadny. analyzované budú všetky pakety, naopak ak je zadaný aspoň jeden, budú analyzované práve pakety daného typu. Sú to nasledujúce:
`-t/--tcp`, `-u/--udp`, `--icmp`, `--arp`

PRÁCA SNIFFERU, POUŽITÉ FUNKCIE

Vo funkcii `main()` je po definícii pár potrebných premenných volaná funkcia `parseArguments()`, ktorá má na starosti prácu so vstupnými argumentami.

Hneď po skončení tejto funkcie je kontrola či nebol zadaný prepínač `-i/-interface` *`rozhranie`* zadaný bez rozhrania, ak áno zavolá sa funkcia `PrintDevices()`, ktorá vypíše dostupné zariadenia na `stdout`.

Ďalej je volaná funkcia `pcap_open_live()`, ktorá je základom celej implementácie, otvára dane rozhranie a vracia jeho popisovač.

Teraz potrebujeme získať masku a IP siete na to nám poslúži funkcia `pcap_lookupnet()`.

Potom je volaná ďalšia funkcia a to `assembleFilter()`. Ta sa stará o nastavenie filtra na základe triedy `Arguments`. Filter je uložený ako obyčajný reťazec, ktorý ale zodpovedá požadovanej syntaxi funkcií `pcap_compile()`, ktorá je volaná neskôr.

`Pcap_compile()` slúži na skompilovanie filtra z reťazca do potrebného formátu `struct bpf_program`. Ak kompilácia prebehne úspešne(funkcia nevráti `PCAP_ERROR`) prichádza na rad funkcia na nastavenie filtru `pcap_setfilter()`.

Posledná potrebná funkcia `pcap_loop()` je volaná až teraz, zachytáva pakety podľa nastaveného filtra, a prebehne toľkokrát (toľko paketov zachytí), koľko jej je zadané. Jej tretim parametrom je funkcia `pSniffer()` .

Táto funkcia je veľmi dôležitá a komplexná, prebehne zakaždým ako je zachytený vyhovujúci paket. Najprv zavolá funkciu `PrintTime()`, ktorá vypíše aktuálny čas (čiže čas kedy bol paket zachytený) v požadovanom formáte. Prvý „switch“ rozhodne podľa premennej `ether_type`, ktorá sa nachádza v štruktúre `ether_header` či sa jedná o paket IPv4, IPv6 alebo priamo ARP. V vnútri „casu“ pre IPv4 sa nachádza ďalší switch, ktorý rozhodne o tom k akému protokolu daný paket patrí, pre korektný prístup k údajom ako sú IP adresy a čísla portov, cez ktorý sa k nám paket dostal. V prípade IPv6 sa mi podarilo nasimulovať (teda aj implementovať) iba ICMPv6 pakety. Tie na výpis adresy volajú funkciu Ipv6Expander(). Ak je načítaný paket protokolu ARP sú vypísane obe jeho MAC adresy. Na výpis paketu ako takého sa v každom prípade používa funkcia `PrintPacket()`.

PRÍKLADY SPUSTENIA

```
$ sudo ./ipk-sniffer
$ sudo ./ipk-sniffer -i
$ sudo ./ipk-sniffer -i eth0
$ sudo ./ipk-sniffer -i eth0 -n 10
$ sudo ./ipk-sniffer -i eth0 -p 23 --tcp -n 2
$ sudo ./ipk-sniffer -i eth0 --tcp --udp
$ sudo ./ipk-sniffer -i eth0 --udp
```

ZDROJE

1. https://linuxos.sk/clanok/packet-capturing-s-libpcap-1/
2. https://linuxos.sk/clanok/packet-capturing-s-libpcap-2/
3. https://linux.die.net/man/3/pcap
4. https://stackoverflow.com/questions/54325137/c-rfc3339-timestamp-with-milliseconds-using-stdchrono
5. https://stackoverflow.com/questions/3727421/expand-an-ipv6-address-so-i-can-print-it-to-stdout
6. https://www.programcreek.com/cpp/?code=mq1n%2FNoMercy%2FNoMercy-master%2FSource%2FClient%2FNM_Engine%2FINetworkScanner.cpp