Síťový analyzátor

Bc. Jiří Hrazdil
xhrazd06

2008-04-20

1. Úvod

Cílem projektu je vytvořit aplikaci schopnou zachytávat a filtrovat pakety na určitém síťovém rozhraní s následným výpisem základních informací o paketu v závislosti na protokolu. Implementace je založena na knihovně pcap, na které stojí známý odchytávač paketů tcpdump.

Knihovna pcap nám umožní zachytávat provoz na úrovni rámců síťové vrstvy. K získání informací z rámce neposkytuje knihovna žádné funkce, musíme použít buď vlastnoručně sestavené struktury, nebo existující struktury definované v hlavičkových souborech knihovny pro práci se sockety.

2. Návrh aplikace, popis implementace

Aplikace sestává z 5 funkcí. V hlavní funkci `main()` dochází ke zpracování parametrů zadaných na příkazové řádce a vykonání nějaké akce – buď výpisu rozhraní, na kterých je možno zachytávat, nebo je spuštěna funkce realizující samotné zachytávání. Výpis všech rozhraní je řešen ve funkci `outputAllInterfaces()`, která vypíše všechna rozhraní, na kterých je teoreticky možno zachytávat pakety (má-li aplikace dostatečná oprávnění).

Další funkcí je `capturePackets(char *iface, int port, int numberPackets, int tcpFilter, int udpFilter)`, která tvoří výkonné jádro aplikace. První parametr specifikuje rozhraní, na kterém se bude zachytávat. Druhý parametr určuje port, podle kterého se budou filtrovat pakety. Není-li zadán, je předána hodnota -1, která zaručí, že se filtrování podle čísla portu nebude realizovat. Třetí parametr určuje počet paketů, které se budou zachytávat. Jsou-li čtvrtý, resp. pátý parametr nastaveny na nenulovou hodnotu, budou se zobrazovat jen TCP, resp. UDP pakety. V případě, že je zadáno číslo portu z rozsahu 0..65535 a není zapnuto filtrování TCP nebo UDP paketů, je toto zapnuto, protože při jiných protokolech nemá smysl číslo portu uvažovat. V případě, že je zadáno číslo portu a zapnuto filtrování TCP nebo UDP, tak se toto nastavení filtrování nezmění. Provádění funkcí skončí buď po zachycení zadaného počtu paketů nebo při výskytu chyby.

U každého paketu je vypsán čas příchodu, s přesností na mikrosekundy, dále pak označení protokolu (IP pro protokoly TCP a UDP, XX v ostatních případech), adresu odesílatele a příjemce paketu (buď se vypisuje IP adresa a port – pro protokoly transportní vrstvy TCP a UDP - nebo MAC adresa pro ostatní protokoly) a na závěr obsah kompletního zachyceného ethernetového rámce.

Vyhledání všech aktivních rozhraní je řešeno funkcí `pcap_findalldevs()`, zachytávání pak pomocí `pcap_open_live()` a `pcap_next()`, přičemž všechny tři uvedené funkce jsou z knihovny pcap. Vyhodnocení případných filtrů, uvedených v příkazovém řádku, je provedeno porovnáváním hodnot v samotné aplikaci, nepoužívají se filtry knihovny pcap v podobě funkcí `pcap_compile()` a `pcap_setfilter()`.

Dále jsou v aplikace obsaženy pomocné funkce `printMac()` a `printPacketContents()`, které slouží k výpisu MAC adresy, resp. obsahu ethernetového rámce. Funkce `printPacketContents()` vypisuje řádky po 16-bajtových blocích.

Pro analýzu obsahu parametrů příkazové řádky je použito funkce `getopt()`.

Následuje výpis jednotlivých struktur, použitých v aplikaci (jsou uvedeny pouze použité datové položky).

Z hlavičky ethernetového rámce získáme jak zdrojovou, tak cílovou MAC adresu, ale i označení protokolu síťové vrstvy (v našem případě zda-li se jedná o protokol IP nebo ne).

```
struct ether_header {
    u_int8_t ether_dhost[ETH_ALEN];
    u_int8_t ether_shost[ETH_ALEN];
    u_int16_t ether_type;
}
```

Z IP hlavičky bereme protokol transportní vrstvy a zdrojovou a cílovou IP adresu.

```
struct iphdr {
    u_int8_t protocol; /* protokol */
    u_int32_t saddr; /* zdrojova adresa */
    u_int32_t daddr; /* cilova adresa */
  };
```

Z hlavičky TCP paketu bereme pouze zdrojový a cílový port. Struktura pro UDP pakety vypadá úplně stejně, proto ji nebudu uvádět.

```
struct tcphdr {
    u_int16_t source; /* zdrojový port */
    u_int16_t dest; /* cílový port */
};
```

3. Základní informace o programu

Program byl vyvíjen v prostředí OS Linux, distribuce Arch Linux, verze jádra 2.6.24. Pro generování síťového provozu byly použity aplikace wget, links, ping, telnet. Vývoj a testování probíhal přes účet superuživatele.

4. Shoda se zadáním

Program vyhovuje zadání – jedinou odlišností od původní podoby zadání je neimplementování zjišťování doménového jména k IP adresám.

5. Návod k použití

Program se spouští pomocí:

```
analyzer [-i rozhraní] [-p port] [-tcp] [-udp] [-n num]
```

Všechny parametry jsou nepovinné. Není-li zadán parametr `-i` s patřičným názvem rozhraní, program vypíše rozhraní, na kterých je možno zachytávat pakety, společně s IP adresou a maskou přiřazenou k rozhraní a ukončí se. Parametr `-p` omezuje výpis pouze na TCP a UDP pakety, jejichž zdrojový nebo cílový port se rovná zadané hodnotě. Parametry `-tcp` a `-udp` specifikují, že se budou zobrazovat pouze TCP, resp. UDP pakety. V případě, že je zadán parametr `-p` a nejsou určeny filtry `-tcp` a `-udp`, odchytávají se pouze pakety TCP a UDP. Parametr `-n` slouží ke stanovení počtu paketů, které se budou odchytávat. V případě, že není zadán, aplikace se ukončí po odchycení jednoho paketu odpovídajícího parametrům.

6. Použité knihovny, zdroje

Pro implementace jsem použil knihovnu pcap, dostupnou z webové stránky http://www.tcpdump.org/

analýzy parametrů příkazové řádky: inspirována http://www.gnu.org/software/libtool/manual/libc/Example-of-Getopt.html

výpis IP adresy a masky u rozhraní: inspirován http://yuba.stanford.edu/~casado/pcap/section1.html

převod IP adresy získané z paketu: inspirován http://www.haifux.org/lectures/107/ip-iidi-1.html

výpis rozhraní (použití pcap_findalldevs): inspirováno http://dog.tele.jp/winpcap/html/group__wpcapsamp1.html

struktura ethernetového rámce – struktura ether_header ze souboru netiket/if_ether.h

struktura ip paketu – struktura iphdr ze souboru netinet/ip.h

struktura tcp datagramu – struktura tcphdr ze souboru netinet/tcp.h

struktura udp datagramu – struktura udphdr ze souboru netinet/udp.h