Fakulta Informačních Technologií

Vysoké Učení Technické v Brně

Dokumentácia k projektu ISA

Analyzátor sieťovej prevádzky

Michael Halinár - xhalin01

19. novembra 2016

Obsah

1 Základné informácie o programe

Aplikácia dokáže filtrovať pakety na základe mac,ipv4,ipv6,udp portu alebo tcp portu a počítať dáta obsiahnuté v paketoch. Ďalej je možné pomocou filtru top10 vypísať 10 položiek ktoré preniesli v paketoch najviac dát.

2 Úvod do problematiky

Mojou úlohou bola implementácia aplikácie pre analýzu sieťovej prevádzky uloženej vo formáte libcap. Táto aplikácia počíta prenesené bajty na základe požadovaných filtrov.

Prvým krokom bolo oboznámenie sa s formátom libcap. Tento formát obsahuje globálnu hlavičku a za ňou postupnosť jednotlivých hlavičiek paketov a samotných dát paketov. Nás budú zaujímať hlavičky paketov v ktorých sa nachádza veľkosť zachyteného paketu. Z tejto veľkosti paketu následne zistíme koľko bajtov treba načítať aby sme dostali celý paket. Ak máme paket môžeme sa pustiť do parsovania jednotlivých hlavičiek rámca.

Ako prvá v pakete sa nachádza ethernetová hlavička, ktorá má na prvých šiestich bajtoch cieľovú adresu, nasleduje zdrojová adresa na ďalších šiestich bajtoch a ďalšie dva bajty slúžia na identifikáciu typu dalšej hlavičky.

V prípade že nasleduje ipv4 hlavička 20 ďalších bajtov patrí jej. V tejto hlavičke nás bude zaujímať celková dĺžka na prvých dvoch bajtoch (pre výpočet prípadnej výplne - padding), ďalej typ nasledujúceho protokolu, cieľová adresa a zdrojová adresa na posledných 8 bajtoch.

Ak za ehernetovou hlavičkou nasledovala ipv6 hlavička, nasledujúcich 40 bajtov patrí jej. Tu nás zaujímal jej 7. bajt ktorý obsahuje informáciu o nasledujúcom protokole. Posledných 32 bajtov patrí cieľovej a zdrojovej adrese.

Po ipv4 alebo ipv6 hlavičke môže nasledovať udp alebo tcp protokol. Udp protokol je dlhý 8 bajtov, pričom nás zaujíma cieľový a zdrojový port, prípadne veľkosť dát ktoré sú za ním. V prípade tcp protokolu nás zaujíma zase zdrojový a cieľový port a veľkosť dát.

Po týchto protokoloch nasledujú už iba dáta pri ktorých počítame celkovú veľkosť.

3 Návrh

Vstupný súbor pcap načítame do buffra z neho budeme oddeľovať jednotlivé hlavičky a pakety. Keď sa paket oddelí, je poslaný do cyklu ktorý ho ďalej parsuje. V prvom kroku spracuje ethernetovú hlavičku. Tá sa rozdelí do štruktúry `ethHeader`. Podľa identifikácie v ethernet protokole paket pošleme ďalej do parsovania ipv4 alebo ipv6 hlavičky. Ďalšou variantou je protokol ARP ktorý môže nasledovať za ethernetovou hlavičkou. V tomto prípade musíme brať do úvahy že ARP protokol je veľký 18 bajtov a treba ho odčítať z celkovej počítanej veľkosti. Pri parsovaní ipv4 alebo ipv6 hlavičky ich rozdeľuje do štruktúr `ipv4Header` či `ipv6Header`. Ďalej si zistíme čo pokračuje za týmito hlavičkami, môže to byť tcp alebo udp protokol,tie sa parsujú do štruktúry `udpHeader` alebo `tcpHeader`. Ďalšie dáta v pakete už nikde nerozdeľujeme iba si uchovávame ich veľkosť.

V prípade ipv4 hlavičky musíme dávať pozor aj na ethernet padding. Ten sa počíta nasledovne:

$$padding = sizeOfPacket - ipv4Header.totalLength - 14 \quad (1)$$

Kde *sizeOfPacket* je dĺžka zachyteného paketu v súbore, *ipv4Header.totalLength* je celková dĺžka ipv4 hlavičky aj s dátami a 14 je veľkosť ethernetovej hlavičky. Ak padding je kladné číslo, musíme ho odčítať z výslednej počítanej hodnoty. Týmto získame veľkosť dát bez nechcenej výplne.

4 Implementácia

Dôležitým krokom bolo načítanie vstupných argumentov do takej podoby aby sa dal ľahko porovnávať s hodnotami v paketoch. S týmto problémom veľmi pomohlo formátované čítanie `sscanf`. Ktoré načítalo vstup do veľmi priaznivej podoby a to do poľa integerov, ktoré sa jednoducho porovnávali s dátami v paketoch. Ako telo programu slúžil jeden veľký `while` cyklus ktorý spracováva pakety jeden po druhom.

Počítanie hodnôt pracovalo tak že pomocné premenné pre jednotlivé vrstvy sa každý cyklus nulujú. Ak nastáva zhoda paketu so vstupným filtrom, tak si tieto premenné uchovajú jednotlivé hodnoty a pričítajú ich do výslednej celkovej hodnoty. Pre filter top10 bolo treba vytvoriť vektor štruktúr. Ktoré obsahovali vždy 3 informácie a to adresu či už mac,ipv4,ipv6 alebo port, ďalej 2 hodnoty celkového počtu prenesených bajtov. Pri každom pakete v závislosti na tom či vektor štruktúru s danou adresou alebo portom už obsahoval, buď pripočítal hodnoty paketu alebo sa vytvorila nová štruktúra ktorá sa vložila do vektora. Toto umožnilo počítanie dát sieťovej prevádzky pre jednotlivé adresy.

5 Návod na použitie

./analyzer [-i subor] [-f typFiltra] [-v hodnotaFiltra] [-s] [-d]

-i subor - (povinný parameter) vstupný súbor vo formáte libpcap

-f typFiltra - (povinný parameter) určenie podľa ktorej položky sa počíta objem dát. Možné hodnoty: mac, ipv4, ipv6, tcp, udp

-v hodnotaFiltra - (povinný parameter) hodnota filtra. Možné hodnoty napr.: 5C:D5:96:2C:38:63 (pre mac), 192.168.1.1 (pre ipv4), 2001::1 (pre ipv6), 80 (pre tcp, udp), top10 (pre mac, ipv4, ipv6, tcp, udp)

-s - (minimálne jeden z parametrov s/d musí byť zadaný) filter sa aplikuje na zdrojové adresy (MAC, IPv4, IPv6, port)

-d - (minimálne jeden z parametrov s/d musí byť zadaný) filter sa aplikuje na cieľové adresy (MAC, IPv4, IPv6, port)

Literatúra

[1] Libcap File Format, https://wiki.wireshark.org/Development/LibpcapFileFormat

[2] Prednášky z predmetu ISA