Vysoké Učení Technické v Brně

Fakulta Informacních Technológií

Počítačové komunikácie a siete

Samuel Líška (xliska20)

30. Apríl 2020

1. Úvod

Navrhnite a implementuje sieťový analyzátor, ktorý bufe schopný na určitom sieťovom rozhraní zachytávať a filtrovať TCP/UDP packety.

Užívateľ zadá pomocou argument-i rozhranie na ktorom sa bude program vykonávať. Ďalej môže špecifikovať typ odhytávaných packetov TCP/UDP prípadne port cez ktorý sa prijímajú.

2. Implementácia

I. Nastavenie rozhrania

Rozhranie je definované užívateľom cez argumenty(ak nie vypíše sa zoznam dostupných rozhraní). Následne je použitá funkcia pcap_lookupdev pre získanie masky a IP rozhrania

```
pcap_lookupnet(interface, &net, &mask, errbuf)
```

II. Otvorenie rozhrania pre odchytávanie packetov

Pre otvorenie rozhriania používa program funkciu z knižnice pcap konkrétne pcap_open_live:

```
handle = pcap_open_live(interface, BUFSIZ, 1, 1000, errbuf);
```

Funkcia je nastavená na "promiscuous mode" (3. parameter) t.j. zachytáva veškerú "premávku".
BUFSIZ je veľkosť bufferu definovaná v pcap.h
Týmto sa otvorí session pre sniffovanie.

III. Špecifikácia odchytávania packetov

Užívateľ môže zadať či chce odchytávať TCP alebo UDP packety(prípadne oboje). Tieto vstupy sa uložia do stringu **filter_exp** ktorú je nutné následne skompilovať do štruktúry **bfp_program fp** pomocou funkcie pcap_compile:

```
pcap_compile(handle, &fp, filter_exp, 0, net)
```

Výsledný "filter program" **fp** potom môže byť aplikovaný na pakety prúdiace cez **handle** pomocou funkcie pcap_setfilter:

```
pcap_setfilter(handle, &fp)
```

V štruktúre **fp** sa ukladá typ packetov, prípadne port.

IV. Samotné odchytávanie packetov

Na konci mainu sa nastaví cyklus cez pcap_loop ktorý odchytá počet packetov **display_packets** v predošle otvorenej session **handle** a spracováva ich cez *callback* funckiu got_packet.

```
pcap_loop(handle, display_packets, got_packet, NULL);
```

got_packet:

Na začiatku funkcie sa zadefinujú hlavičky, keďže packet sa ukladá za seba bude treba ich neskúr pričítavať ako *offset* aby sme sa dopracovali ku konkrétnemu packetu.

```
const struct sniff_ethernet *ethernet;
const struct sniff_ip *ip;
const struct sniff_tcp *tcp;
```

x

Následne sa vypočútajú konkrétne veľkosti jednotlivých častí packetu takto:(tabuľka je použitá zo stránky www.tcpdump.org link v zdrojoch)

Variable	Location (in bytes)
Sniff ethernet	X
Sniff ip	X + SIZE_ETHERNET
sniff tcp	X + SIZE_ETHERNET + IP_HEADER_LENGTH
Payload	X + SIZE_ETHERNET + IP_HEADER_LENGTH + TCP HEADER LENGTH
Payload udp	X + SIZE_ETHERNET + IP_HEADER_LENGTH + UDP HEADER LENGTH

Ďalej sa rozhodne či sa jedná o TCP respektíve UDP packet. Samotné spracovanie packetov je implementované dvoma spôsobmi.

Implementácia získania TCP packetu je silne inšpirovaná z doporučovaných zdrojov a je implementovaná ručne pomocou štruktúry **sniff_tcp**

UDP packet je implementovaný pomocou knižnice **netinet/udp.h**
Stačilo zadefinovať offset, veľkosť udp, payloadu a štruktura v knižnici mala predchystané premenné ako **d_port**, **s_port** a podobné ktoré stačilo vypísať.
Na konci funkcie got_packet sa zavolá funkcia print_payload ktorá spracuje vypočítaný náklad, pri UDP packete:

```
payload = (u_char *)(packet + SIZE_ETHERNET + size_ip + size_udp);
```

Samotná funckia nevypisuje, ale vypočítava offsety a dĺžky, následne volá funkciu print_hex ktorá najskôr vypíše veľkosť bytov v hex-a tvare následne vypisuje a každých 8 znakov spravá pre prehľadnosť medzeru ako pri vzorovom výstupe.

3. Zdroje

https://www.tcpdump.org/pcap.html?fbclid=IwAR375Isdym7Ioc7tZAujXrie5DtYKE4ddR8OGDyT5dOqDjhLYoMq7-2UIzY

https://www.devdungeon.com/content/using-libpcap-c

UNIX Nástroj man

Prednášky IPK