Vysoké učení technické v Brně
Fakulta informačních technologií

Počítačové komunikace a sítě

Projekt 2 - Varianta ZETA: Sniffer paketů

3. mája 2020 Viktória Halečková (xhalec00)

Obsah

1 Zadanie

- Navrhnite a implementujte sieťový analyzátor v C/C++/C#, ktorý bude schopný na určitom sieťovom rozhraní zachytávať a filtrovať pakety.
- Vytvorte relevantný manuál/dokumentáciu k projektu

Volanie programu: `./ipk-sniffer -i` *rozhranie* `[-p` *port*`] [--tcp|-t] [--udp|-u] [-n` *num*`]`

- `-i` *eth0* - rozhranie, na ktorom sa bude počúvať, ak tento parameter nebude uvedený, vypíše sa zoznam aktívnych rozhraní
- `-p` *23* - filtrovanie paketov na danom rozhraní podľa portu, ak tento parameter nebude uvedený, uvažujú sa všetky porty
- `-t/--tcp` - budú sa zobrazovať len tcp pakety
- `-u/--udp` - budú sa zobrazovať len udp pakety
- ak nebude `-tcp` ani `-udp` špecifikované, uvažujú sa TCP a UDP pakety zároveň
- `-n` *10* - určuje počet paketov, ktoré sa majú zobraziť, ak to nie je uvedené, uvažujte zobrazenie len pre 1 paket

Formát výstupu:

```
čas IP|FQDN : port > IP|FQDN : port

počet_vypísaných_bajtov: výpis_bajtov_hexa výpis_bajtov_ASCII
```

2 Štruktúra programu

2.1 Hlavičkové súbory a globálne premenné

Zdrojový kód je uložený v jednom súbore `ipk-sniffer.c`. Všetky libpcap programy vyžadujú, aby hlavičkový súbor `pcap.h` získal prístup k funkciám a konštantám knižnice. Záhlavia siete `Netinet` a `Arpa` poskytujú dátové štruktúry, ktoré zjednodušujú prístup k poliam záhlavia špecifických pre protokol. Zahrnuté sú štandardné hlavičky ANSI a UNIX, aby program mohol zobrazovať obsah paketov a spracovávať signály ukončenia programu. V programe sú používané tieto globálne premenné:

- `linkhdrlen` - veľkosť hlavičky odkazu, používa sa počas zachytávania a analýzy paketov
- `num_packets` - počet spracovávaných paketov
- `dev` - rozhranie, na ktorom sa bude počúvať
- `port` - filtrovanie paketov na danom rozhraní podľa portu
- `filter_exp` - nastavenie filtra

2.2 Funkcia main

Cieľom je zhromaždiť raw pakety prechádzajúce sieťou, aby bolo možné skontrolovať ich polia záhlavia a užitočného zaťaženia a určiť typ protokolu, zdrojovú adresu, cieľovú adresu a pod.

3 Práca s paketom

V terminológii programovania systému UNIX je soket koncový bod pre sieťovú komunikáciu, ktorý je identifikovaný v programe s deskriptorom soketu.

3.1 Výber rozhrania

Sieťové rozhrania sú označené jedinečnými znakovými reťazcami označovanými ako sieťové zariadenia na stránke manuálu libpcap. Napríklad v systéme Linux majú ethernetové zariadenia všeobecnú formu `ethN`, kde `N == 0, 1, 2` atď., V závislosti od toho, koľko sieťových rozhraní systém obsahuje. Prvým argumentom open_pcap_socket() je reťazec zariadenia získaný z príkazového riadka programu. Ak nie je zadané žiadne zariadenie, vypíše sa zoznam aktívnych rozhraní.

3.2 Otvorenie zariadenia pre snímanie

`pcap_open_live()` otvorí vybrané sieťové zariadenie na zachytávanie paketov a v prípade úspešnosti vráti deskriptor soketu libpcap. Prvým argumentom pre túto funkciu je sieťové zariadenie použité na zachytávanie paketov, druhý nastavuje maximálnu veľkosť každého prijatého paketu, tretí prepína promiskuitný režim, štvrtý nastavuje časový limit a posledný je ukazovateľ na medzipamäť chybových správ.

3.3 Získanie sieťovej adresy a subnet masky

`pcap_lookupnet()` vráti sieťovú adresu a masku podsiete pre socket zachytávania paketov. Na zostavenie reťazca filtra paketov je potrebná masku podsiete. Posledným argumentom tejto funkcie je ukazovateľ na medzipamäť chybových správ.

3.4 Filter

`pcap_compile()` skonvertuje argument reťazca filtrov paketov `open_pcap_live()` na filtračný program, ktorý dokáže libcap interpretovať. Prvý argument pre `pcap_compile()` je deskriptor soketu libpcap, druhý je ukazovateľ na reťazec filtra paketov, tretí je ukazovateľ na prázdnu štruktúru programu filtra libpcap, štvrtý je nepoužitý parameter nastavený na 0 a posledný je 32 bitový ukazovateľ na masku podsiete, ktorá bola získaná pomocou `pcap_lookupnet()`. Od tejto chvíle funkcie libpcap vráti 0, ak sú úspešné, a -1 pri chybe.

3.5 Nastavenie filtra

`pcap_setfilter()` nainštaluje kompilovaný program filtrovania paketov do zariadenia na zachytávanie paketov. To spôsobí, že libpcap začne zhromažďovať pakety, ktoré boli vybrané pomocou filtra.

4 Loop na zachytávanie paketov

4.1 Loop funkcia

`Libpcap` poskytuje 3 funkcie na zachytenie paketov: `pcap_next()`, `pcap_dispatch()` a `pcap_loop()`. Prvá funkcia chytí jeden paket naraz, takže musí byť volaný v slučke, aby prijal viac paketov. Ostatné dve automaticky prijímajú viac paketov a volajú funkciu spätného volania dodávanú používateľom na spracovanie každého z nich.

4.2 Určenie typu datalinku

Pakety, ktoré sú zachytené vo vrstve dátových liniek, sú úplne nespracované v tom zmysle, že obsahujú hlavičky aplikované všetkými vrstvami sieťových zásobníkov, vrátane hlavičky dátových liniek. Nás zaujímajú iba údaje IP paketov, takže chceme preskočiť hlavičku dátového odkazu obsiahnutú v každom pakete. `pcap_datalink()` nám pomáha tým, že vracia číslo, ktoré zodpovedá typu dátového odkazu priradeného k soketu na snímanie paketov. Vzhľadom na typ dátového odkazu je uložená zodpovedajúca veľkosť záhlavia dátového odkazu v globálnej premennej `linkhdrlen` na použitie neskôr, keď sa analyzujú pakety IP. Medzi podporované typy dátových liniek patria `loopback(DLT_NULL)`, `Ethernet(DLT_EN10MB)`, `SLIP(DLT_SLIP)` a `PPP(DLT_PPP)`.

4.3 Zachytávanie paketov

`pcap_loop()` nastavuje počet paketov a inštaluje funkciu spätného volania.

5 Analýza a zobrazenie paketov

5.1 Funkcia analýzy

Všeobecnou technikou na analyzovanie paketov je nastaviť znakový ukazateľ na začiatok vyrovnávacej pamäte paketov a potom poslať tento ukazovateľ na hlavičku konkrétneho protokolu podľa veľkosti v záhlaví hlavičiek, ktoré mu predchádzajú v pakete. Záhlavie sa potom môže mapovať do štruktúry záhlavia IP, TCP, UDP a ICMP odliatím ukazovateľa znakov na ukazovateľ štruktúry špecifický pre protokol. Odtiaľ je možné priamo odkazovať na akékoľvek pole hlavičky protokolu pomocou ukazovateľa štruktúry protokolu.

5.2 Definícia ukazateľov protokolu

`got_packet()` začína definovaním ukazovateľov na štruktúry hlavičiek IP, TCP, UDP a ICMP. Znakové vyrovnávacie pamäte sú zahrnuté na ukladanie polí záhlavia, ktoré sa zobrazia na vyradenie.

5.3 Analýza IP hlavičky

Pointer paketov sa posúva za hlavičku dátového odkazu počtom bajtov zodpovedajúcich typu dátového odkazu určeného v `capt_loop()`. Toto umiestni pointer na začiatok hlavičky IP, kde bola premiestnená na pointer štruktúrovaného ip, aby bolo možné ľahko extrahovať ID paketu, čas života, dĺžku hlavičky IP a celkovú dĺžku paketu IP (vrátane hlavičky). Tieto hodnoty sú umiestnené do jedného znakového medzipamäte na neskoršie zobrazenie. Pretože polia záhlavia 2 a 4 bajty pre všetky internetové protokoly sú vo veľkom formáte endian, používa sa `ntohs()` a `ntohl()` na korekciu usporiadania bajtov na malých endianových systémoch. Potom je posunutý pointer paketov okolo hlavičky IP tak, aby ukazoval na užitočné zaťaženie IP. Nakoniec je určený protokol užitočného zaťaženia a prechádza sa na časť kódu určenú na prácu s týmto protokolom.

5.4 Analýza TCP a UDP hlavičky

Odovzdanie ukazovateľov paketov na pointre štruktúr tcp a udp umožňuje prístup k poliam záhlavia TCP a UDP. V oboch prípadoch sa zdrojová adresa IP a port zobrazia so symbolom > smerujúcim na cieľovú adresu IP a port. Následne je vypísaný celý paket v žiadanej forme.

6 Testovanie

Program bol testovaný na virtuálnom stroji vytvorenom z poskytnutého referenčného Linux obrazu.

6.1 Výpis aktívnych rozhraní

Použitý príkaz: `./ipk-sniffer`

Výstup programu:

```
Active interfaces:
1. enp0s3
2. any
3. lo
4. nflog
5. nfqueue
6. usbmon1
```

6.2 Výpis tcp paketu

Použitý príkaz: `./ipk-sniffer -i enp0s3 --tcp`

Výstup programu:

```
18:06:13 10.0.2.15 : 36022 > 10.0.2.15 : 443

0x0000 17 03 03 00 85 ef e7 08 84 cc 1e 08 c0 85 bf a8 ................
0x0010 3c df 41 15 66 76 09 88 06 bc c5 de 8b d4 25 45 <.A.fv........%E
0x0020 0e 29 4a 92 9a b4 04 da 60 18 09 d0 c4 18 ac a3 .)J.....`.......
0x0030 fc 21 7f fe b6 2b 21 78 40 f3 27 b2 99 0f 3d af .!...+!x@.'...=.
0x0040 04 2d 52 a0 ff 35 70 35 39 07 43 2f 8a 00 8e 9a .-R..5p59.C/....
0x0050 7a ea cb 74 40 cc 49 dd 41 1c 6f f3 cd 65 ff f6 z..t@.I.A.o..e..
0x0060 27 fb 68 ef ce f6 b0 96 0c 0b 45 12 de 7e 7f 11 '.h.......E.. ..
0x0070 50 3b c5 9c 74 f6 9f 4e df 0d a7 3f 6e 95 fa 69 P;..t..N...?n..i
0x0080 9b 93 2c 60 a5 e8 f7 65 1f 71 ..,`...e.q
```

6.3 Výpis udp paketu

Použitý príkaz: `./ipk-sniffer -i enp0s3 -u`

Výstup programu:

```
18:11:20 10.0.2.15 : 21076 > 10.0.2.15 : 18
```

6.4 Výpis paketu na porte 443

Použitý príkaz: `./ipk-sniffer -i enp0s3 -p 443`

Výstup programu:

```
18:13:55 10.0.2.15 : 36022 > 10.0.2.15 : 443

0x0000 17 03 03 00 66 c0 99 37 3d b0 ba d7 7b c9 ca 70 ....f..7=...{..p
0x0010 45 4e 65 d5 11 43 43 11 12 7e ea 8d fc 3e b6 2d ENe..CC.. ...>.-
0x0020 08 8f d2 c8 5d 2d b0 e0 9e 08 66 e1 1e dc a5 e7 ....]-....f.....
0x0030 6e 50 68 fc 2e bd 06 44 a8 2e 41 ae db 24 69 e7 nPh....D..A..$i.
```

```
0x0040 00 4d bb bf 0c 69 d4 43 5f 37 0a 13 55 48 25 77 .M...i.C_7..UH.w
0x0050 56 75 4d 3a af 9a 4b 5c e6 d4 1c 84 80 c9 d3 c7 VuM:..K.........
0x0060 06 ed c1 1a 6e 2d e1 ab 59 ee b2 ....n-..Y..
```

6.5 Výpis 2 udp paketov

Použitý príkaz: `./ipk-sniffer -i enp0s3 -n 2 --udp`

Výstup programu:

```
18:27:05 10.0.2.15 : 21076 > 10.0.2.15 : 18

18:27:47 10.0.2.15 : 21076 > 10.0.2.15 : 18
```

7 Bibliografia

[1] pcap. Wikipedia [online]. Wikimedia Foundation, Inc., 2020 [cit. 2020-04-28].
Dostupné z: https://en.wikipedia.org/wiki/Pcap

[2] Packet analyzer. Wikipedia [online]. Wikimedia Foundation, Inc., 2020 [cit. 2020-04-28].
Dostupné z: https://en.wikipedia.org/wiki/Packet_analyzer

[3] Manpage of Tcpdump [online]. The Tcpdump Group, 2020 [cit. 2020-04-28].
Dostupné z: https://www.tcpdump.org/manpages/tcpdump.1.html

[4] Link-local address [online]. Wikimedia Foundation, Inc., 2020 [cit. 2020-04-28].
Dostupné z: https://en.wikipedia.org/wiki/Link-local_address

[5] Capturing Packets in Your C Program, with libpcap by Pankaj Tanwar [online]. 2011 [cit. 2020-04-28]. Dostupné z: https://opensourceforu.com/2011/02/capturing-packets-c-program-libpcap/

[6] Create your own packet sniffer in C by Varun Gupta [online]. 2012 [cit. 2020-04-28].
Dostupné z: http://simplestcodings.blogspot.com/2010/10/create-your-own-packet-sniffer-in-c.html

[7] Develop a Packet Sniffer with Libpcap by Vic Hargrave [online]. 2012 [cit. 2020-04-28].
Dostupné z: https://vichargrave.github.io/programming/develop-a-packet-sniffer-with-libpcap/compile-a-packet-capture-filter