Vysoké učení technické v Brně
Fakulta informačních technologií

Počítačové komunikace a sítě

2. Projekt: Varianta ZETA

Sniffer Paketů

23. dubna 2022 Pavel Heřmann (xherma34)

1 Úvod

Sniffer paketů neboli analyzátor paketů je program, který slouží k zachycování paketů v počítačové síti. Paket je blok dat v počítačových sítích, který nám umožňuje přenášet data i při výpadcích některých spojů. Obsahuje metadata a uživatelská data, neboli tzv. payload.

2 Implementace programu

2.1 Základní informace

Za úkol bylo implementovat sniffer paketů, který zachytává pakety s protokoly udp, tcp a icmp na síti ethernet. Projekt je implementován v jazyce C a využívá hlavně knihovnu pcap.h a sadu knihoven z netinet.

2.2 Implementace

Program se skládá ze dvou hlavních částí: načítání, zpracování argumentů a navázání relace, parsování packetu a výpis požadovaných hodnot.

2.2.1 Zpracování argumentů a navázání relace

Pro načítání a zpracování argumentů jsem implementoval funkci `argparse`, která bere vstupní argumenty z `argv[]`, který následně ve for cyklu projíždí, nastavuje booleanovské hodnoty, číslo portu, název interface a číslo portu. Používá také pomocné funkce jako např. `isnumber`, která zjišťuje zda je string číslo, dále funkci `devicesOnly` která při spuštění programu bez argumentů nebo pouze s argumentem -i vypíše všechna dostupná zařízení a program ukončí.

Následně je volána funkce `setfilter`, která z nastavených hodnoty vytváří filtr ve tvaru validním pro funkci `pcap_setfilter` z knihovny pcap. Do funkce se pošle array do kterého se podle příslušné podmínky kopíruje hodnota, popřípadě pokud je již hodnota do arraye nakopírována, tak string v arrayi konkatenuje v příslušném tvaru.

Dále se otevře relace pro chytání paketů pomocí funkce knihovny pcap `pcap_open_live`, pakliže je spojení neúspěšné, tak se program ukončí a na errorový výstup je vypsán důvod. Následuje kontrola hodnoty filtru. Pokud je hodnota ve filtru různá od 'none' dochází ke kompilaci filtru v relaciu a následného nastavení filtru, toto zařizují funkce `pcap_compile` a `pcap_setfilter`.

Poslední krok je následně zavolání funkce `pcap_loop` která nám zachytí paket, který je následně poslán do funkce `callbackfunc`.

2.2.2 Parsování paketů

O parsování se stará již zmíněná funkce `callbackfunc`, která si jako argument bere otevřenou relaci, hlavičku paketu a samotný paket. Na začátku se inicializuje array `iptype`, který je později využíván pro pomocné funkce na rozeznání mezi ipv4 a ipv6, po inicializaci je mu přidělena hodnota 'none' pro zjednodušení práce.

Vytvoří se hlavička etherHeader typu `ether_header` a volá se funkce `printtime`, která nám vytiskne čas v požadovaném tvaru a funkce `printmacadresses` která si bere jako argument ethernetHeader a z něj si získává mac adresy pro tisknutí. Dále se vytiskne velikost rámce.

Z ethernetové hlavičky se získá `ether_type`, která obsahuje informaci, zda se jedná o packet ipv4, ipv6 nebo arp. V příslušném if statementu se vytvoří struktura pro uchování ip hlavičky a do proměnné `iptype` se uloží informace o který typ se jedná. Následně se volá funkce `printipaddresses`, který podle `iptype` zpracuje ip hlavičku a vytiskne ip adresy.

Z příslušné pozice ip hlavičky se vytáhne informace zda je protokol tcp, udp nebo icmp. Pro každý případ je implementována funkce pro tisknutí portů. Jako argument se bere ethernetová hlavička, proměnná `iptype`, packet a velikost ip hlavičky (pro ipv4 získaná a pro ipv6 fixní 40 bytů).

Tisknutí samotných dat (payload) zařizuje funkce `printpayload`, která si bere jako argument samotný paket a poté velikost dat. Tisknutí dat je poté prováděno ve for cyklu, který tiskne dle požadovaného tvaru.

2.2.3 Implementační detaily

printtime: Do proměnné unixTime je uložený čas pomocí funkce `localtime`, ten je převeden funkcí `strftime` do požadovaného tvaru a uložen v proměnné `time`. Požadované milisekundy jsou převedeny na array a konkatenovány s proměnnou `time`. V poslední části je získán offset časového pásma a ten je také konkatenován s proměnnou `time`, která je následně tisknuta.

printmacaddresses: Z argumentu se vezme ethernetová hlavička, která obsahuje src a dst adresu. Pomocí funkce `ether_ntoa` je adresa převedena do čitelné podoby(písmena a čísla oddělená dvojtečkou) a tisknuta.

printipaddresses: Nazačátku je inicializována hlavička jak pro ipv6 tak pro ipv4. Podle rozhodující proměnné je potom vybrán typ tisknutí. Z ipv4 hlavičky se získají adresy a funkcí `inet_ntoa` jsou převedeny na string, který je následně vytisknut.

Ipv6 hlavička potřebuje pro tisknutí pole, takže je inicializováno src a dst pole o velikosti INET6_ADDRSTRLEN, což je největší možná velikost ipv6 pro string. Funkcí `inet_ntop` je převedena na string a uložena do inicializovaných polí, která se následně tisknou.

print[udp/tcp]ports: Všechny funkce tohoto typu fungují stejně, v argumentu dostanou informaci zda je paket ipv4 nebo ipv6 a následně je vytvořena struktura pro získání hlavišky protokolu, ta je funkcí `ntohs` převedena na tisknutelný integer.

printpayload Do proměnné buffer se vezme packet. Ten je následně po hexabytech projížděn. Uvnitř hlavního for cyklu jsou tisknuty pozice a také obsahuije další dva for cykly. V prvním vnořeném for cyklu se po hexabytech printují hexa hodnoty jednotlivých znaků a ve druhém vnořeném for cyklu jsou hodnoty tisknuty jako znaky pokud pro hodnotu znak existuje, jinak je tisknuta tečka.

2.3 Testování

2.3.1 Testovací ICMP paket

```
**********NEW PACKET**********
timestamp: 2022-04-23T21:52:02.131+02:00
src MAC: 0:0:0:0:0:0
dst MAC: 0:0:0:0:0:0
frame length: 98 bytes
src IP: 127.0.0.1
dst IP: 127.0.0.1
0x0000: 00 00 00 00 00 00 00 00 00 00 00 00 08 00 45 00 ..............E.
0x0010: 00 54 d3 3a 00 00 40 01 a9 6c 7f 00 00 01 7f 00 .T.:..@..l......
0x0020: 00 01 00 00 89 55 00 1b 00 01 e2 58 64 62 00 00 .....U.....Xdb..
0x0030: 00 00 6f 00 02 00 00 00 00 00 10 11 12 13 14 15 ..o.............
0x0040: 16 17 18 19 1a 1b 1c 1d 1e 1f 20 21 22 23 24 25 .......... !"#$%
0x0050: 26 27 28 29 2a 2b 2c 2d 2e 2f 30 31 32 33 34 35 &'()*+,-./012345
0x0060: 36 37                                           67
```

Obrázek 1: ICMP paket z programu

```
0000  00 00 00 00 00 00 00 00  00 00 00 00 08 00 45 00   ........ ......E·
0010  00 54 d3 39 40 00 40 01  69 6d 7f 00 00 01 7f 00   ·T·9@·@· im······
0020  00 01 08 00 81 55 00 1b  00 01 e2 58 64 62 00 00   ·····U·· ···Xdb··
0030  00 00 6f 00 02 00 00 00  00 00 10 11 12 13 14 15   ··o····· ········
0040  16 17 18 19 1a 1b 1c 1d  1e 1f 20 21 22 23 24 25   ········ ·· !"#$%
0050  26 27 28 29 2a 2b 2c 2d  2e 2f 30 31 32 33 34 35   &'()*+,- ./012345
0060  36 37                                             67
```

Obrázek 2: ICMP paket z programu wireshark

```
9 2.133337557   127.0.0.1          127.0.0.1          ICMP       98 Echo (ping) request  id=0x001b, seq=1/256, ttl=64 (reply in 1…
```

Obrázek 3: Metadata paketu z programu wireshark

2.3.2 Testovací TCP paket

```
0x0000: 60 63 4c 61 f1 6f 24 4b fe 77 62 ba 08 00 45 00 `cLa.o$K.wb...E.
0x0010: 00 81 7d 55 40 00 40 11 ce ca c0 a8 00 6b d5 a3 ..}U@.@......k..
0x0020: 57 95 eb d3 c3 65 00 6d ee ca 90 e6 57 2f 1e 70 W....e.m....W/.p
0x0030: 24 08 00 0b 3b aa be de 00 03 d4 1c 5b 08 bf ef $...;.......[...
0x0040: f2 45 a0 29 7a 85 89 ce d5 6a 0e f8 64 a4 89 76 .E.)z....j..d..v
0x0050: 52 36 72 5c 2d 56 21 22 4c 18 7e 20 bc 2e 6d d7 R6r\-V!"L.~ ..m.
0x0060: 0a 93 d0 99 b5 1c 41 9f b6 40 67 f3 74 09 86 72 ......A..@g.t..r
0x0070: 55 13 c2 2c 2a 0b 26 27 76 56 0c 5e c6 cc 11 64 U..,*.&'vV.^...d
0x0080: 72 f2 f1 6c 5b 9d 94 bd 63 e4 36 65 d0 15 80    r..l[...c.6e...
```

Obrázek 4: TCP paket z programu

```
0000  60 63 4c 61 f1 6f 24 4b  fe 77 62 ba 08 00 45 00   `cLa·o$K ·wb···E·
0010  00 81 7d 55 40 00 40 11  ce ca c0 a8 00 6b d5 a3   ··}U@·@· ·····k··
0020  57 95 eb d3 c3 65 00 6d  ee ca 90 e6 57 2f 1e 70   W·····e·m ····W/·p
0030  24 08 00 0b 3b aa be de  00 03 d4 1c 5b 08 bf ef   $···;··· ····[···
0040  f2 45 a0 29 7a 85 89 ce  d5 6a 0e f8 64 a4 89 76   ·E·)z··· ·j··d··v
0050  52 36 72 5c 2d 56 21 22  4c 18 7e 20 bc 2e 6d d7   R6r\-V!" L·~ ·.m·
0060  0a 93 d0 99 b5 1c 41 9f  b6 40 67 f3 74 09 86 72   ·····A· ·@g·t··r
0070  55 13 c2 2c 2a 0b 26 27  76 56 0c 5e c6 cc 11 64   U··,*·&' vV·^···d
0080  72 f2 f1 6c 5b 9d 94 bd  63 e4 36 65 d0 15 80      r··l[··· c·6e···
```

Obrázek 5: TCP paket z programu wireshark