

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

POČÍTAČOVÉ KOMUNIKACE A SÍTĚ 2021/2022

2. projekt do predmetu IPK Varianta ZETA: Sniffer paketov

Pavel KRATOCHVÍL - xkrato61

18.4.2022



Obsah

1	Úvod	2
2	Úvod do problematiky	2
2.1	Čo je paket?	2
2.2	Štruktúra paketu	2
2.3	Zaobalenie paketu(packet encapsulation)	2
3	Implementácia	3
4	Rozšírenia	3
5	Výstup programu	4
5.1	Štruktúra výstupu	4
6	Testovanie	5
6.1	Porovnanie s referenčným nástrojom Wireshark	5

1 Úvod

Tento dokument popisuje princíp implementácie, fungovania nástroja v rámci predmetu IPK pre odchyťovanie a analýzu paketov.

2 Úvod do problematiky

2.1 Čo je paket?

Paket (angl. packet - balíček) je základná jednotka dát prenášaná počas komunikácie v TCP/IP sieti. Paket má zo špecifikácie stanovenú štruktúru a obmedzenú dĺžku. Delením správ komunikácie do množstva paketov vieme docieľiť konzistentnú a spoľahlivú komunikáciu medzi dvomi zariadeniami.

Na prenos paketu nie je potrebná žiadna dopredu zahájená komunikácia medzi dvomi zariadeniami. Vďaka tomu môžeme vyslať do siete paket s určeným cieľom a ak existuje cesta k príjemcovi a komunikácia prebehla v poriadku, paket dorazí do cieľa bez ďalších potrebných akcií z našej strany.[1]

2.2 Štruktúra paketu

Paket je reťazec bitov rozdelených do dvoch hlavných častí:

- Riadiace dáta(metadáta)
- Uživatelské dáta(angl. payload)

Riadiace metadáta umožňujú správny pohyb paketu medzi zariadeniami. Takýto pohyb ale môže byť značne komplexný a zahŕňajúci množstvo rôznych zariadení. Preto bol vyvinutý abstraktný na vrstvách založený opis návrhu štruktúry komunikačných a počítačových sieťových protokolov s názvom OSI[2] (Open Systems Interconnection Reference Model). Tento model rozčleňuje pohyb dát do siedmych vrstiev, z ktorých každá vrstva poskytuje služby vrstve nad ňou a využíva vrstvy pod ňou.[3]

Aby každá vrstva mala potrebné informácie pre operácie s paketom s ňou spojené, pri odosielaní paketu každá vrstva zaobalí paket. Zaobalením paketu rozumieme pripojenie ďalších metadát k jeho užívateľskému obsahu.

2.3 Zaobalenie paketu(packet encapsulation)

Prvá vrstva(najvrchnejšia) obalu, ktorou sa `ipk-sniffer` zaoberá je 2. spojová vrstva OSI. Môj program umožňuje ďalšie rozbalovanie iba Ethernet paketov. Pod ňou nájdeme 3. sieťovú vrstvu. V nej môžeme ďalej analyzovať IP, ICMP a ARP rámce. Pod nimi nasleduje 4. transportná vrstva, ktorá je obmedzená na TCP a UDP pakety.

3 Implementácia

Pri implementácii projektu som používal mnoho štandardných knižníc a tiež aplikačné programové rozhranie `pcap`, ktoré mi v mnohom uľahčilo prácu. Keďže sa ale o danú oblasť zaujímam, rozhodol som sa nepoužiť niektoré vstavané funkcie zamerané na filtrovanie odchytených paketov (napr. `pcap-filter`), čo sa odrazilo aj na rozsiahlosti môjho riešenia.

Kvôli nepoužitiu vstavaného filtra a k nemu patriacich funkcií pre iterovanie v získaných paketoch, som zvolil nekonečnú slučku odchyťovania paketov `pcap_loop()`, ktorú zastaví buď dosiahnutý počet odchytených paketov špecifikovaný užívateľom (`-n X`), chyba počas behu programu (napr. chyba alokácie pamäte) alebo zastavenie za behu užívateľom (napr. `ctrl+c`).

Po nájdení nového paketu nekonečnou slučkou sa spúšťa funkcia `got_packet()`, ktorá podľa hlavičiek a obsahu rozhodne, ktorá ďalšia funkcia bude otvárať nasledujúcu vrstvu zaobalenia.

Na ukladanie informácií získaných o konkrétnom pakete využívam štruktúru v globálnej premennej `packet_storage`. Na uľahčenie práce s reťazcami využívam vlastnú implementáciu dynamického reťazca v súbore `dynamic_string.c`.

Pri implementácii som sa opieral hlavne o dokumentáciu, zdrojové kódy jednotlivých knižníc a referenčný návod ku knižnici `pcap`[4].

4 Rozšírenia

V mojom riešení som implementoval aj jedno malé rozšírenie, ktoré užívateľovi umožňuje filtrovanie paketov podľa verzie IP protokolu. Použitie filtra je možné vstupným argumentom `--ipv4` alebo `--ipv6`. Vďaka manuálnej implementácii triedenia paketov v mojom riešení by bola implementácia ďalších filtrov podľa akejkoľvek zachytenej informácie triviálna.

5 Výstup programu

5.1 Štruktúra výstupu

Výstupom môjho programu je n blokov, z ktorých každý odpovedá jednému zachytenému paketu. Blok môže obsahovať nasledujúce položky:

```
timestamp:
eth type:
protocol:
arp protocol:
arp operation:
src MAC:
dst MAC
src MAC (IPv6):
dst MAC (IPv6):
frame length:
src IP:
dst IP:
src port:
dst port:
ICMP type:
```

```
0x0000 28 d0 ea 5c 3d 25 d8 47 32 f5 d9 c5 08 00 45 00 (...\\=%.G 2.....E.
```

Ktoré položky sa zobrazia závisí na type nájdeného paketu a jeho hlavičkách (protokolu a typu obsahu). Napríklad pred IPv4 sa nam zobrazia iba polia `timestamp`, `eth type`, `protocol`, `src` a `dst MAC`, `frame length`, `src` a `dst IP`, `src` a `dst port` a `payload size`. Obsah celého rámca sa zobrazí v každom prípade.

Formát výpisu obsahu jedného riadka rámca je: `offset` (posun v rámci v hexadecimálnom formáte), 16 byte-ov dát, z ktorých každý je reprezentovaný dvojicou znakov v hexadecimálnej sústave a na konci je reprezentácia tých istých 16 byte-ov v ASCII znakoch. Netlačiteľné znaky sú nahradené bodkou.

6 Testovanie

6.1 Porovnanie s referenčným nástrojom Wireshark

Pri testovaní som si vybral ako referenčný nástroj Wireshark[5](verzia 3.6.2). Porovnanie výstupov pre jeden zachytený paket(IPv4 TCP):

```

    ▸ Frame 45: 66 bytes on wire (528 bits), 66 bytes captured (528 bits) on interface wlp0s20f3, id 0
    ▸ Ethernet II, Src: Tp-LinkT_f5:d9:c5 (d8:47:32:f5:d9:c5), Dst: IntelCor_5c:3d:25 (28:d0:ea:5c:3d:25)
    ▸ Internet Protocol Version 4, Src: 2.16.2.27, Dst: 192.168.0.105
      0100 .... = Version: 4
      .... 0101 = Header Length: 20 bytes (5)
      ▸ Differentiated Services Field: 0x00 (DSCP: CS0, ECN: Not-ECT)
      Total Length: 52
      Identification: 0x4368 (17256)
      ▸ Flags: 0x40, Don't fragment
      ...0 0000 0000 0000 = Fragment Offset: 0
      Time to Live: 57
      Protocol: TCP (6)
      Header Checksum: 0x3920 [validation disabled]
      [Header checksum status: Unverified]
      Source Address: 2.16.2.27
      Destination Address: 192.168.0.105
      [Source GeoIP: ]
    ▸ Transmission Control Protocol, Src Port: 443, Dst Port: 38752, Seq: 1, Ack: 1, Len: 0
      Source Port: 443
      Destination Port: 38752
      [Stream index: 1]
      [Conversation completeness: Incomplete (4)]
      [TCP Segment Len: 0]
      Sequence Number: 1 (relative sequence number)
      Sequence Number (raw): 137064874
      [Next Sequence Number: 1 (relative sequence number)]
      Acknowledgment Number: 1 (relative ack number)
      Acknowledgment number (raw): 2182947999
      1000 .... = Header Length: 32 bytes (8)
      ▸ Flags: 0x010 (ACK)
      Window: 244
      [Calculated window size: 244]
      [Window size scaling factor: -1 (unknown)]
      Checksum: 0x9b6e [unverified]
      [Checksum Status: Unverified]
      Urgent Pointer: 0
      ▸ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
      [Timestamps]

```

0000	28 d0 ea 5c 3d 25 d8 47	32 f5 d9 c5 08 00 45 00	(..\=%G 2....E.
0010	00 34 43 68 40 00 39 06	39 20 02 10 02 1b c0 a8	.4Ch@.9. 9
0020	00 69 01 bb 97 60 08 2b	71 aa 82 1d 24 9f 80 10	.i...`.+ q...\$...
0030	00 f4 9b 6e 00 00 01 01	08 0a 8d 1e 7d ee fa 95	...n....}...
0040	55 ce		U.

Obrázek 1: Referenčný výstup z nástroja Wireshark

```

[ OK ] Found interface: wlp0s20f3
Starting...

timestamp:      2022-04-18T01:02:22.211+02:00
eth type:       IPv4
protocol:       TCP
src MAC:        d8:47:32:f5:d9:c5
dst MAC:        28:d0:ea:5c:3d:25
frame length:   66
src IP:         2.16.2.27
dst IP:         192.168.0.105
src port:       443
dst port:       38752
payload size:   0

0x0000  28 d0 ea 5c 3d 25 d8 47  32 f5 d9 c5 08 00 45 00  (..\=%G 2....E.
0x0010  00 34 43 68 40 00 39 06  39 20 02 10 02 1b c0 a8  .4Ch@.9. 9 .....
```

Obrázek 2: Výstup z ipk-sniffer

Literatúra

- [1] Kurose, J. F.; Ross, K. W.: *Computer Networking: A Top-Down Approach*. Boston, MA: Pearson, 7 vydání, 2016, ISBN 978-0-13-359414-0.
- [2] Open Systems Interconnection – Connection-oriented Session protocol: Protocol specification. Nov 1995.
URL <https://www.itu.int/rec/T-REC-X.225-199511-I/en>
- [3] Data Encapsulation and the TCP/IP Protocol Stack. 2010.
URL <https://docs.oracle.com/cd/E19455-01/806-0916/ipov-32/index.html>
- [4] Cartens, T.: Programming with PCAP. 2002.
URL <https://www.tcpdump.org/pcap.html>
- [5] Wireshark. [online], 2022.
URL <https://www.wireshark.org/>