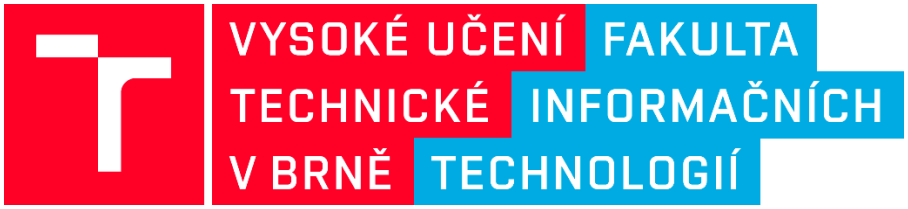

ISA – Monitorovanie DHCP komunikácie

Timotej Bučka
xbucka00

17.11.2023

Obsah

1. Úvod do problematiky

Cieľom projektu bolo vytvoriť program pre získavanie štatistiky o vyťažení žiadaných sieťových prefixov. Štatistiky sa zameriavajú na počet alokovaných adries a ich percentuálnu početnosť v porovnaní s celkovým počtom adries pre istý IP prefix. Program pri prekročení zaplnenia prefixu nad 50% možných adries informuje užívateľa o tomto fakte. Táto informácia sa zaeviduje: 1. na výstup aplikácie 2. do súboru systémových logov. Pre monitorovanie stavu zaplnenia je väčšinou využívaný DHCP server, no v tomto projekte sa pre počítanie štatistík zaplnenia monitoruje DHCP komunikácia serveru a klienta. Táto komunikácie môže byť čítaná zo súboru resp. priamo počúvať na určitom sieťovom rozhraní.

2. Teoretická stránka projektu

DHCP (Dynamic Host Configuration Protocol) je klient/server ktorý slúži na pridávanie IP adries DHCP klientom. Ako transportný protokol je v DHCP používaný BOOTP (Bootstrap Protocol).[1] BOOTP využíva UDP (User Datagram Protocol) ako transportný protokol[2], pričom UDP na transport využíva IP (Internet Protocol).[3] Na najnižšej vrstve sa využíva Ethernet.[4] Každý DHCP paket obsahuje Ethernet, IP a UDP hlavičku. V DHCP sú správy posielané medzi klientom a serverom. Typy DHCP správ sú nasledujúce[5]:

- DHCPDISCOVER – Správa klienta pre nájdenie DHCP servera
- DHCPOFFER – Odpoveď serveru na DHCPDISCOVER s konfiguračnými parametrami
- DHCPREQUEST – Správa klienta, ktorá požaduje parametre od servera alebo predlžuje dobu „lease time“
- DHCPACK – Správa servera potvrdzujúca klientsku požiadavku obsahujúca potvrdenú IP adresu
- DHCPNAK – Negatívna odpoveď servera na požiadavku
- DHCPDECLINE – Správa klienta indikujúca, že IP adresa je už používaná
- DHCPRELEASE – Správa klienta indikujúca zbavenie sa IP adresy pridelenej danému klientovi
- DHCPINFORM – Správa klienta žiadajúca si parametre lokálnej konfigurácie.

2.1. Stavba DHCP paketu[6]

V projekte sa ráta so štruktúrou DHCP paketu takou aká je definovaná v štandarde rfc2131.

[1] [6]
[2] [1]
[3] [2]
[4] [3]
[5] [4]
[6] [5]

3. Implementačná stránka projektu

3.1. Spustenie programu

Projekt je implementovaný v jazyku C++. Pre vytvorenie spustiteľného programu je nutné v adresári priečinka, kde sa nachádzajú súbory zdrojového kódu programu spustiť príkaz `make.` Po zavolaní príkazu `make` sa v priečinku vytvorí spustiteľný súbor `dhcp-stats`.

Príklad spustenia programu:

- `./dhcp-stats -i eth0 192.155.12.0/24`

Argumenty programu:

- `-i <interface-name>` – Program bude zachytávať DHCP komunikáciu z daného rozhrania. V prípade zadania nesprávneho/neznámeho rozhrania. Program končí s návratovou hodnotou `1 (INVALID_PROGRAM_ARGS)` a chybovú hlášku vypíše na štandardný chybový výstup.
- `-r <file-name>` - Program bude čítať DHCP komunikáciu z daného súboru. V prípade zadania nesprávneho typu súboru alebo neznámeho súboru skončí program návratovou hodnotou `1 (INVALID_PROGRAM_ARGS)` a chybovú hlášku vypíše na štandardný chybový výstup.
- `-h` a `--help` – Program vypíše na štandardný výstup nápovedu a končí vykonávanie.
- Každý ďalší argument programu je braný ako **IP prefix**.
  Každý IP prefix je formátu: `\d+\.\d+\.\d+\.\d+/\d{1,2}`. V prípade zadania nesprávneho prefixu program končí s návratovou hodnotou `2 (INVALID_PREFIX).`
  V prípade nezadania žiadneho prefixu program končí s návratovou hodnotou `1 (INVALID_PROGRAM_ARGS).`

  Pre spustenie programu je nutné volať program výlučne s jedným z `-r` a `-i.` Každý ďalší spôsob bude viesť na ukončenie programu s návratovou hodnotou `1 (INVALID_PROGRAM_ARGS).`

3.2. Popis funkcionality a fungovania programu `dhcp-stats`

Po spracovaní argumentov začína program čítať DHCP komunikáciu. Pri čítaní zo súboru je výstupná štatistika vypísaná na štandardný výstup a v prípade korektného behu programu program končí úspešne. V prípade monitorovania sieťového rozhrania sa štatistika vypisuje a aktualizuje pomocou funkcií knižnice `ncurses.h`. V prípade presiahnutia 50% zaplnenia jedného z IP prefixov zadaného užívateľom sa vypíše správa do súboru systémových logov v tvare: `dhcp-stats[xxxx]: prefix 192.168.1.0/28 exceeded 50% of allocations .` Na konci programu je uvoľnená pamäť zdrojov a uzavreté predtým otvorené periférie.

3.2.1. Sledovanie DHCP komunikácie

Na sledovanie DHCP komunikácie sú využité funkcie knižnice `pcap.h`. Pri súbore dochádza k inicializácii pomocou príkazu:

```
descr = pcap_open_offline(g_args.fileName.c_str(), errbuf);
```

Pri sledovaní sieťového rozhrania:

```
descr = pcap_open_live(g_args.interface.c_str(), BUFSIZ, 0, -1, errbuf);
```

Pri funkcii `pcap_open_live` je 3. parameter 0 čo znamená, že program odchytáva len tú komunikáciu, ktorá je určená zariadeniu na ktorom je spustený (zhodná MAC adresa). 4. parameter indikuje čakaciu dobu na paket. V prípade -1 čaká pokiaľ nejaký paket nebude odchytený.

Keďže DHCP využíva BOOTP, odchytávanie komunikácie sa deje na portoch 67 a 68.[7] V programe sú na to využité nasledovné volania:

```
pcap_compile(descr, &fp, "udp port 67 or udp port 68", 0, PCAP_NETMASK_UNKNOWN)
```

- vytvorí BPF (Berkeley Packet Filter) program (`fp`) z filter výrazu bez optimalizácie.

```
pcap_setfilter(descr, &fp)
```

- priradenie filtra

Na samotné zachytávanie paketov v cykle je volaná funkcia:

```
pcap_loop(descr, 0, packetCallback, NULL);
```

- argument 0 indikuje zachytávanie paketov pokiaľ nevznikne chyba pri behu alebo v prípade že je beh explicitne ukončený
- `packetCallback` je funkcia ktorá je zavolaná nad každým odchytením paketom.

3.2.2. packetCallback

```
void packetCallback(u_char *args, const struct pcap_pkthdr *pkthdr, const u_char
*packet);
```

Po príchode paketu do tejto funkcie, je smerník `packet` upravený nasledovne:

```
packet += 14 + 20 + 8;
```

- Odsadenie o 14 bajtov znamená preskočenie Ethernet hlavičky
- Odsadenie o ďalších 20 bajtov znamená preskočenie IPv4 hlavičky
- Odsadenie o ďalších 8 bajtov znamená preskočenie UDP hlavičky

Pre získanie adresy pridelenej serverom (yiaddr časť DHCP packetu) je potrebné odsadiť smerník `packet` o ďalších 16 bajtov[8]:

```
const uint8_t *yiaddr_ptr = packet + 16;
uint32_t ip = 0;
memcpy(&ip, yiaddr_ptr, 4);
ip = ntohl(ip);
```

[7] [6]
[8] [5]

Pre zistenie typu DHCP správy je nutné prechod cez časť options DHCP paketu[8]:

```
bool is_ack = false;
const uint8_t *option_ptr = packet + 240;

while (*option_ptr != 255) {
      const uint8_t *option_len_ptr = option_ptr + 1;
      if (*option_len_ptr == 0) {
           option_ptr += 2;
           continue;
      }

      const uint8_t *option_data_ptr = option_ptr + 2;

      if (*option_ptr == 53 && *option_data_ptr == 5) {
           is_ack = true;
           break;
       }

      option_ptr += *option_len_ptr + 2; // +2 to skip option_length bit and
start after option_data bits
}
```

V prípade nájdenia option-u č. 53 – DHCP Message Type, ktorý má hodnotu 5 – ACK, je možné vyhlásiť daný paket za paket typu DHCP_ACK a teda danú IP adresu zarátať do štatistík.

V prípade, ak však už bola daná IP adresa niekedy počas behu programu zarátaná do štatistík prefixu, do štatistík sa po druhý krát nepridá.

3.2.3. Trieda `IPPrefix`

Trieda `IPPrefix` reprezentuje IP prefix a je uchováva potrebné údaje o zaplnení, maximálny počet adries, broadcast adresu, masku prefixu a iné.

Prevod IP adresy z typu `string` to typu `uint32_t`:

```
int ret = inet_pton(AF_INET, this->address.c_str(), &(this->bitAddress));
if (ret != 1) {
      throw INVALID_PREFIX;   // address is not valid
}
this->bitAddress = ntohl(this->bitAddress); // get address in correct order
```

Výpočet počtu adries prefixu:

```
this->allocated = (1 << (IP4_ADDR_LEN - this->maskNumber)) - 2; // -2 because of
network and broadcast
```

- `IP4_ADDR_LEN` je definované ako 32

Vytvorenie masky prefixu:

```
this->bitMask = (1 << this->maskNumber) - 1;
this->bitMask <<= (IP4_ADDR_LEN - this->maskNumber); // shift to left
```

- pre prefix s číslom masky 28 je do premennej **`bitMask`** typu **`uint32_t`** uložená hodnota 11111111111111111111111111110000.

Pri kontrole príslušnosti IP adresy pod daný prefix dochádza k využívanie atribútov objektov triedy **`IPPrefix`**:

```
if ((ip & prefix.getBitMask()) == (prefix.getBitAddress() & prefix.getBitMask())
&& ip != prefix.getBitAddress() && ip != prefix.getBroadcastBitAddress()) {
      //ip is part of a prefix yay
}
```

3.3. Príklad výstupu programu

```
                                    ./dhcp-stats -r pcap_files/dhcp-ack-last.pcapng 192.168.1.0/28 192.168.1.0/27 192.168.1.0/22
IP-Prefix       Max-hosts       Allocated       Utilization     Over50%
192.168.1.0/28  14              14              100%            YES
192.168.1.0/27  30              20              66.7%           YES
192.168.1.0/22  1022            20              1.96%           NO
```

4. Literatúra

[1] Wireshark, "BOOTP," [Online]. Available: https://wiki.wireshark.org/BOOTP#protocol-dependencies. [Accessed 17 November 2023].

[2] Wireshark, "User_Datagram_Protocol," [Online]. Available: https://wiki.wireshark.org/User_Datagram_Protocol#protocol-dependencies. [Accessed 17 November 2023].

[3] Wireshark, "Internet_Protocol," [Online]. Available: https://wiki.wireshark.org/Internet_Protocol#protocol-dependencies. [Accessed November 17 2023].

[4] paloaltonetworks, "DHCP Messages," [Online]. Available: https://docs.paloaltonetworks.com/pan-os/9-1/pan-os-admin/networking/dhcp/dhcp-messages. [Accessed 17 November 2023].

[5] R. Droms and T. Lemon, "Dynamic Host Configuration Protocol," 1997. [Online]. Available: https://www.ietf.org/rfc/rfc2131.txt. [Accessed 17 November 2023].

[6] Wireshark, "DHCP," [Online]. Available: https://wiki.wireshark.org/DHCP. [Accessed 17 November 2023].