VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Síťové aplikace a správa sítí

Monitorování DHCP komunikace

20. listopadu 2023

Vladislav Khrisanov
xkhris00

Obsah

1 Introduction and problem overview

Dynamic host configuration protocol (DHCP)[1] is a client-server model protocol that allows for automatic distribution of IP addresses from the available pool between the network hosts (clients) by the DHCP server (it can either be a router or an autonomous server). There are four main stages in this process, abbreviated as DORA:

1. Server discovery
2. IP lease offer
3. IP lease request
4. IP lease acknowledgement

Since we are interested in monitoring assigned addresses (addresses are assigned upon successful completion of four specified stages) and the DHCP is a connectionless protocol that uses UDP, our goal is to catch the last stage acknowledgement message (sent from the server to the client on UDP port 67) to get this information.

2 Application design

The general data flow of the program can be described as follows:

1. Parse received addresse pools and store them for further use.
2. Establish the connection with a source (file or interface).
3. Filter the packet stream.
4. Anylise received IPv4 packets [5].
5. Update the statistics.

3 Implementation details

3.1 Receiving the network traffic

The core of the implementation is the `libpcap` [6] library, which allows you to read the network traffic either from the file or live on the specified interface of your machine. It also provides filter options, which, however, were found unstable during testing across different systems, and thus filters were implemented manually by analyzing the different packet fields directly.

3.2 Managing addresses

The address pools that were specified by the user are stored and managed using the modified version of a linked list from the algorithms class. That allows for efficient, clean, and maintainable code. The linked list element main attributes:

- network address
- number of host bits
- maximum number of hosts
- current number of hosts

In order to effectively keep track of recurring addresses during sniffing, the `std::set` container from the C++ Standard Template Library (STL) was used (otherwise, the code is completely C-compatible).

3.3 Working with IP addresses

Throughout the program, IP addresses are treated as 32-bit unsigned integers with bits arranged in network byte order. That allows us to quickly mask addresses by shifting bits or compare them just by using the == operator.

3.4 Processing received packets

This process can be divided into two main steps. First, move a pointer along the received byte sequence in accordance with the packet structure and length. Then we can either directly access the value or typecast a pointer to easily access certain parts of the bytes stream. There are many structures used for this. Such as `struct ether_header*`, `struct ip*`, `struct udphdr*`, etc. They can either be accessed through a library or written manually (such is the case with `struct udphdr*` due to portability issues).

3.5 Output

The output of the program depends on the chosen mode. When it reads from a file, it prints the statistics to the standard output. If it is in live capture mode, the displayed statistics are dynamically updated as the packets are processed with the help of the `Ncurses`[3] library. If more than half of the addresses in some address pool are used, this event is logged using the `Syslog`[2] library in both modes.

4 Program specification

4.1 Accepted packets

- The program supports untagged, 802.1Q-tagged, and double-tagged Ethernet frames [4].
- It makes sure it is a UDP over IPv4 packet.
- It checks if it is a DHCPACK message by trying to find an option type 53 with a value of 5 in DHCP options.

4.2 Accepted addresses

- The program accepts any correctly formatted IPv4 network addresses (meaning that the host portion should be 0) with a prefix length between 1 and 30.
- The user is responsible for providing semantically valid address pools that are being assigned on the DHCP server to get non-zero statistics.
- The network and broadcast addresses of given pools are not counted as host addresses in the statistics.

5 Usage

5.1 Commands

- `./dhcp-stats [-r <filename>] [-i <interface-name>] <ip-prefix> [<ip-prefix> [ ... ]]`
- `./dhcp-stats --help`

5.2 Examples

- `./dhcp-stats -i eth0 192.168.1.0/24 172.16.32.0/24 192.168.0.0/22`
- `./dhcp-stats -r traffic.pcap 192.168.1.0/24`

Literatura

[1] IETF: Dynamic Host Configuration Protocol. [online]. [vid. 2022-11-20].
Dostupné z: `https://datatracker.ietf.org/doc/html/rfc2131#ref-19`

[2] Linux: syslog(3) — Linux manual page. [online]. [vid. 2022-11-20].
Dostupné z: `https://man7.org/linux/man-pages/man3/syslog.3.html`

[3] Padala, P.: NCURSES Programming HOWTO. [online]. [vid. 2022-11-20].
Dostupné z: `https://tldp.org/HOWTO/NCURSES-Programming-HOWTO/`

[4] RFC: Definitions of Managed Objects for Bridges with Traffic Classes, Multicast Filtering and Virtual LAN Extensions. [online]. [vid. 2022-11-20].
Dostupné z: `https://www.ietf.org/rfc/rfc2674.txt`

[5] RFC: Internet Protocol. [online]. [vid. 2022-11-20].
Dostupné z: `https://datatracker.ietf.org/doc/html/rfc791`

[6] Tim Carstens, G. H.: Programming with PCAP. [online]. [vid. 2022-11-20].
Dostupné z: `https://www.tcpdump.org/pcap.html`