Brno University of Technology

Faculty of Information Technology

IPK - Počítačové komunikace a sítě 2021/2022

Varianta ZETA: Sniffer paketů

Suvorkin Artur, (xsuvor00)

Contents

Implementation

Project was implemented in C ++. The basis of the project is built on the pcap library, which functions take care of all work with packets and their interception. It is also worth mentioning libraries as **`chrono`** for writing time in required format RFC3339, **`netinet`** library, to work with the structures of supported packets.

Usage

Sniffer can be run with multiple input arguments. Order doesn't matter.

- First argument is **`-i / --interface`**, this is a required argument which program tells you on which interface the packets will be analyzed. If this argument is not specified or is specified without interface, program lists all available interfaces on the current device and exit.
- **`-p port`** argument is used to filter packets by port.
- **`-n pocet_paketov`** argument specifies the number of packets to be printed on **`stdout`**.
- Next following group of switches **`-t / --tcp, -u / --udp, --icmp`** is used to set the packet filter according to their protocols. If none of them is specified. Then all packets will be analyzed, conversely . If specified at least one, then packet of the given type will be analyzed.

```
$sudo ./ipk-sniffer -i
$sudo ./ipk-sniffer -i eth0 -p 8000
$sudo ./ipk-sniffer -i eth0 --tcp --udp
```

Main functions and principle of operation of the sniffer

Function `main()` after defining a few necessary variables call function `argumentParser()` which is working with input arguments. `interfaceLoad()` function is called, which uses functions from `pcap library pcap_findalldevs()` loads all available interfaces on current device and returns a pointer to the first interface in the list. To check whether the specified interface is valid and present `interfaceCheck()` function is defined in the list of available interfaces, which will print an error message on `stderr` if interface is invalid and program is terminated. `Filter()` function takes care of setting filter based on global variables. Filter is stored in a global variable as a regular string, but it matches the required function syntax `pcap_compile()`, `pcap_compile()` is used to compile filter from string to required format `struct bpf_program`. If compilation was successful then function `pcap_setfilter()` filter setup comes in handy. `pcap_loop()` function captures packets according to the set filter, and runs as many times as it is specified.
`packetSniffer()` function. First call `timePrint()` function, which prints current time. First switch decides according to the `ether_type` variable, which is located in the `ether_header` structure, whether it is an IPv4, IPv6 packet. Within the "time" for IPv4 there is another switch that decides which protocol the packet belongs to, for correct access to data such as IP addresses and port numbers through which the packet reached us. `packetPrint()` function to extract packet.

Resources

1. pcap_findalldevs example:

   http://embeddedguruji.blogspot.com/2014/01/pcapfindalldevs-example.html

2. C++ RFC3339 timestamp with milliseconds using std::chrono:

   https://stackoverflow.com/questions/54325137/c-rfc3339-timestamp-with-milliseconds-using-stdchrono*/

3. Expand an IPv6 address so I can print it to stdout:

   https://stackoverflow.com/questions/3727421/expand-an-ipv6-address-so-i-can-print-it-to-stdout%20*/

4. Manual page of pcap:

   http://www.tcpdump.org/pcap3_man.html

5. INTERNET PROTOCOL DARPA INTERNET PROTOCOL SPECIFICATION:

   https://tools.ietf.org/html/rfc791

6. TRANSMISSION CONTROL PROTOCOL DARPA INTERNET PROGRAM PROTOCOL SPECIFICATION:

   https://tools.ietf.org/html/rfc793