

 NESFIT / IPK-Projects-2024 

Sledovat14

Oblíbit7

Rozštěpit5

<> Zdrojový kód


Úkoly

Požadavky na natažení

Balíčky

Projekty


Vydání



 204c4860a ▾ IPK-Projects-2024 / Project 2 / zeta / README.md

5.3 KiB


<>





Surový


Blame


Historie











Project 2 - ZETA: Network sniffer

Assignment

1. Design and implement a network analyzer that will be able to capture and filter packets on a specific network interface (14 pts.)

2. Create a relevant manual/documentation for the project (6 pts.)

Specification

You should use any *PCAP library available for your language during implementation (e.g., libpcap).

You are expected to use the promiscuous mode of the network card.

The support for IPv4 Options, IPv6 Extension Headers or link-types other than LINKTYPE_ETHERNET is welcome but not required.

The program can be terminated at any given moment with `Ctrl + C` sequence.

Execution

```
./ipk-sniffer [-i interface | --interface interface] {-p|--port-source|--port-destination port [--tcp|-t] [--udp|-u]} [--arp] [--icmp4] [--icmp6] [--igmp] [--mld] {-n num}
```

where:

- `-i eth0` (just one interface to sniff) or `--interface` . If this parameter is not specified (and any other parameters as well), or if only `-i/--interface` is specified without a value (and any other parameters are unspecified), a list of active interfaces is printed (additional information beyond the interface list is welcome but not required).
- `-t` or `--tcp` (will display TCP segments and is optionally complemented by `-p` or `--port-*` functionality).
- `-u` or `--udp` (will display UDP datagrams and is optionally complemented by `-p` or `--port-*` functionality).
- `-p` (extends previous two parameters to filter TCP/UDP based on port number; if this parameter is not present, then no filtering by port number occurs; if the parameter is given, the given port can occur in source OR destination part of TCP/UDP headers).
- `--port-destination 23` (extends previous two parameters to filter TCP/UDP based on port number; if this parameter is not present, then no filtering by port number occurs; if the parameter is given, the given port can occur in destination part of TCP/UDP headers).
- `--port-source 23` (extends previous two parameters to filter TCP/UDP based on port number; if this parameter is not present, then no filtering by port number occurs; if the parameter is given, the given port can occur in source part of TCP/UDP headers).
- `--icmp4` (will display only ICMPv4 packets).
- `--icmp6` (will display only ICMPv6 echo request/response).
- `--arp` (will display only ARP frames).
- `--ndp` (will display only NDP packets, subset of ICMPv6).
- `--igmp` (will display only IGMP packets).
- `--mld` (will display only MLD packets, subset of ICMPv6).
- Unless protocols are explicitly specified, all (i.e., all content, regardless of protocol) are considered for printing.
- `-n 10` (specifies the number of packets to display, i.e., the "time" the program runs; if not specified, consider displaying only one packet, i.e., as if `-n 1`)
- All arguments can be in any order.

Output

Non-printable characters will be replaced by a period, you can also print any padding.

Output format (deviations such as whitespace and other interesting information from frames/packets/datagrams/segments are allowed):

timestamp: *time*

src MAC: MAC address with : as separator

dst MAC: MAC address with : as separator

frame length: *length*

src IP: IP address if any (support v4 but also v6 representation according to RFC5952)

dst IP: IP address if any (support v4 but also v6 representation according to RFC5952)

src port: port number if any

dst port: port number if any

byte_offset: byte_offset_hexa byte_offset_ASCII

whereby:

- *time* is in RFC 3339 format
- *length* is in bytes

Do not print irrelevant information, for instance UDP port number in the case of ICMPv4 message.

Execution Examples

```
./ipk-sniffer -i eth0 -p 23 --tcp -n 2
./ipk-sniffer -i eth0 --udp
./ipk-sniffer -i eth0 --igmp --mld -n 10
./ipk-sniffer -i eth0 --icmp4 --icmp6
./ipk-sniffer -i eth0 --arp --ndp
./ipk-sniffer -i eth0 -n 10
./ipk-sniffer -i eth0 -p 22 --tcp --udp --icmp4 --icmp6 --arp --ndp --igmp --mld
./ipk-sniffer -i eth0
```

Functionality Illustration

```
./ipk-sniffer -i
```

```
lo0
eth0
```

```
./ipk-sniffer -i eth0
timestamp: 2021-03-19T18:42:52.362+01:00
src MAC: 00:1c:2e:92:03:80
dst MAC: 00:1b:3f:56:8a:00
frame length: 512 bytes
src IP: 147.229.13.223
dst IP: 10.10.10.56
src port: 4093
dst port: 80
```

```
0x0000: 00 19 d1 f7 be e5 00 04 96 1d 34 20 08 00 45 00 ..... ..4 ..
```

```
0x0010: 05 a0 52 5b 40 00 36 06 5b db d9 43 16 8c 93 e5 ..R[.@.6. [...C....
0x0020: 0d 6d 00 50 0d fb 3d cd 0a ed 41 d1 a4 ff 50 18 .m.P..=. ..A...P.
0x0030: 19 20 c7 cd 00 00 99 17 f1 60 7a bc 1f 97 2e b7 . .... .`z.....
0x0040: a1 18 f4 0b 5a ff 5f ac 07 71 a8 ac 54 67 3b 39 ....Z._. .q..Tg;9
0x0050: 4e 31 c5 5c 5f b5 37 ed bd 66 ee ea b1 2b 0c 26 N1.\_.7. .f...+.&
0x0060: 98 9d b8 c8 00 80 0c 57 61 87 b0 cd 08 80 00 a1 .....W a.....
```

Illustrated command line output can be customised to provide relevant information in a more structured way. Generally speaking, Wireshark-like output is preferred.

Bibliography

- TCPDUMP/LIBPCAP public repository: <http://www.tcpdump.org/>
- Library <http://packetfactory.openwall.net/projects/libnet/>
- RFC 792 - Internet Control Message Protocol a RFC 4443 - ICMPv6
- RFC 826 - ARP
- RFC 5952 - A Recommendation for IPv6 Address Text Representation
- RFC 3339 - Date and Time on the Internet: Timestamps
- Wikipedia, the free encyclopedia: <http://en.wikipedia.org/wiki/Pcap>

Běží na Gitea Verze: 1.21.10 Strana: 69ms Šablona: 5ms

🌐 Čeština | [Licence](#) | [API](#)