Programming with Libpcap

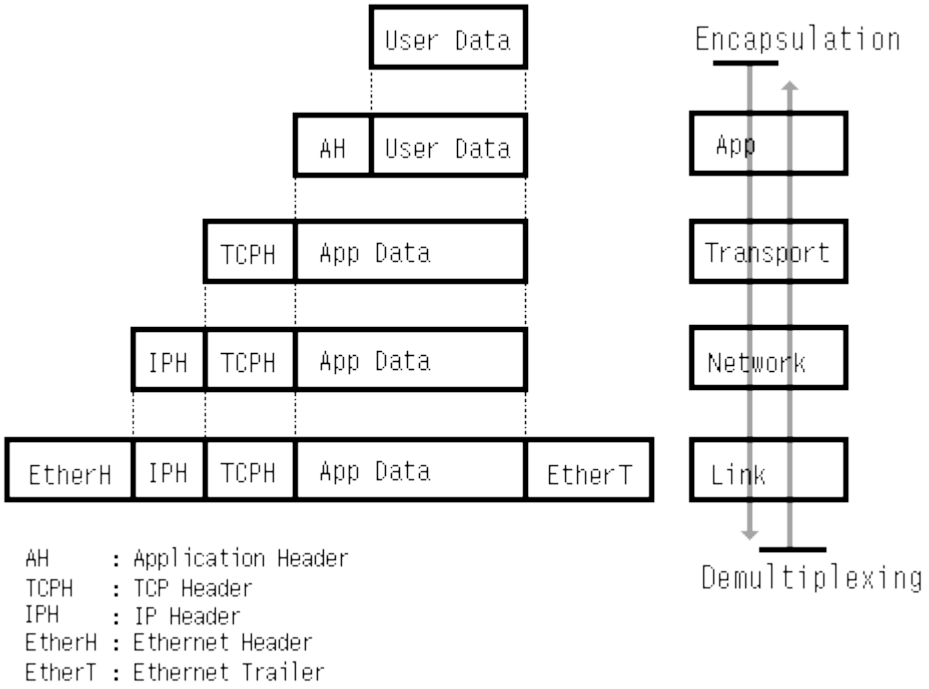
Contents

- Introduction
- Basic Concept of Packet Capturing
- Programming with Libpcap
  - Device & Network Related APIs
  - Initializing Packet Capturing APIs
  - TCP, IP, Ethernet Structures
  - Packet Read Related APIs
  - Filtering Related APIs
- Software based on Libpcap
- Reference

Introduction

- Libpcap: Portable Packet Capturing Library
  - Operating system independent
  - Provide general-purpose APIs
  - Simple and powerful user-level library
  - Compatible with Unix like System
- Other packet capturing tools
  - SOCK_PACKET, LSF, SNOOP, SINT and etc.
  - Operating System defendant
- TCPDUMP is implemented with Libpcap
- Many of commercial IDS systems utilize Libpcap to analyze packet data
- Installation
  - Unix/Linux: http://www.tcpdump.org/#latest-release
  - Windows: http://www.winpcap.org/default.htm
  - Solaris: http://www.sunfreeware.com

Basic Concept of Packet Capturing



- Packet capturing (sniffing) does not affects to data transfer
- The packet captured by libpcap is called raw packet and demultiplexing is required to analyze the packet

Programming with Libpcap

- Programming APIs-

Device & Network Related APIs (1/2)

- `char *pcap_lookupdev(char *errbuf)`
  - return a pointer to a network device suitable for use with *pcap_open_live()* and *pcap_lookupnet()*
  - return *NULL* indicates an error
  - reference: lookupdev.c

- `int pcap_lookupnet(`
  `const char *device, bpf_u_int32 *netp,`
  `bpf_u_int32 *maskp, char *errbuf)`
  - determine the *network number* and *mask* associated with the network device
  - return *-1* indicates an error
  - reference: lookupnet.c

Device & Network Related APIs (2/2)

- What if there are multiple devices?
- `int pcap_findalldevs(pcap_if_t **alldevsp, char *errbuf)`
  - constructs a list of network devices that can be opened with *pcap_create()* and *pcap_activate()* or with *pcap_open_live()*
  - alldevsp: list of network devides
  - returns *0* on success and *-1* on failure.
  - The list of devices must be freed with *pcap_freealldevs()*
- Structure of pcap_if_t
  - next: if not NULL, a pointer to the next element in the list
  - name: a pointer to a string giving a name for the device to pass to pcap_open_live()
  - description: if not NULL, a pointer to a string giving a human-read- able description of the device
  - addresses: a pointer to the first element of a list of addresses
  - flags: interface flags - PCAP_IF_LOOPBACK set if the interface is a loopback interface

Initializing Packet Capturing APIs (1/2)

- pcap_t * : Packet capture descriptor

- ```
  pcap_t *pcap_open_live(
  const char *device, int snaplen,
  int promisc, int to_ms, char *errbuf)
  ```
  - obtain a packet capture descriptor to look at packets on the network
  - snaplen: maximum number of *bytes* to capture
  - promisc: true, set the interface into *promiscuous mode*; false, only bring packets intended for you
  - to_ms: *read timeout* in milliseconds; *zero*, cause a read to wait forever to allow enough packets to arrive
  - return *NULL* indicates an error
  ```

Initializing Packet Capturing APIs (2/2)

- `pcap_t *pcap_open_offline(const char *fname, char *errbuf);`
  - open a “savefile” for reading
  - fname: the name of the file to open
  - return a *pcap_t** on success and *NULL* on failure

Packet Read Related APIs

- `const u_char *pcap_next(pcap_t *p, struct pcap_pkthdr *h)`
  - read the next packet
  - return *NULL* indicates an error
  - pcap_next.c
  - timestamp.c

- `int pcap_loop(pcap_t *p, int cnt, pcap_handler callback, u_char *user)`
  - processes packets from a live capture or “savefile‘” until *cnt* packets are processed
  - A value of -1 or 0 for *cnt* is equivalent to *infinity*
  - *callback* specifies a routine to be called

Filtering Related APIs

- `int pcap_compile(pcap_t *p, struct bpf_program *fp, char *str, int optimize, bpf_u_int32 netmask)`
  - compile the str into a filter program
  - str: filter string
  - optimize: 1, optimization on the resulting code is performed
  - netmask: specify network on which packets are being captured
  - returns *0* on success and *-1* on failure

- `int pcap_setfilter(pcap_t *p, struct bpf_program *fp)`
  - specify a filter program (after compiling filter)
  - return *-1* indicates an error
  - pcap_filter.c

Software based on Libpcap

- ntop - network top
  - a network traffic probe that shows the network usage
  - sort network traffic according to many protocols
  - http://www.ntop.org/overview.html
- snort
  - intrusion prevention and detection system
  - sniff every packet and differentiate general and intrusion by against rules
  - http://www.snort.org/
- ethereal
  - network protocol analyzer
  - http://www.ethereal.com/
- wireshark
  - http://www.wireshark.org/

Reference

- TCPDump
  - http://www.tcpdump.org/pcap.html
- The Sniffer's Guide to Raw Traffic
  - http://yuba.stanford.edu/~casado/pcap/section1.html