

libpcap

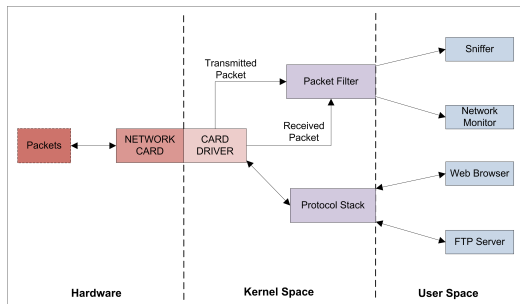
Tématicky zaměřený vývoj aplikací v jazyce C
skupina Systémové programování – Linux

Petr Velan

Fakulta informatiky
Masarykova univerzita
velan@ics.muni.cz

Brno, 19. listopadu 2014

libpcap



- packet capture knihovna (C/C++)
- zpracovává data ještě před OS (včetně NetFilteru!)
- obvykle vyžaduje práva administrátora
- dostupná i pro Windows (WinPcap)
- informace viz `man pcap`
- www.tcpdump.org/

Aplikace

Aplikace používající libpcap

tcpdump

- CLI packet analyzátor
- www.tcpdump.org/

tcpdump

- CLI packet analyzátor
- www.tcpdump.org/

úkol

- vyzkoušejte si `tcpdump -D`

tcpdump

- CLI packet analyzátor
- www.tcpdump.org/

úkol

- vyzkoušejte si `tcpdump -D`
- vyberte si interface a vyzkoušejte `tcpdump -ni eth0`

tcpdump

- CLI packet analyzátor
- www.tcpdump.org/

úkol

- vyzkoušejte si `tcpdump -D`
- vyberte si interface a vyzkoušejte `tcpdump -ni eth0`
- použijte filtr a uložte DNS komunikaci `resolveip`

Wireshark

- grafická obdoba tcpdumpu
- www.wireshark.org/

Wireshark

- grafická obdoba tcpdumpu
- www.wireshark.org/

úkol

- otevřete vzorek uložený tcpdumpem a analyzujte provoz

Další aplikace

- ngrep (network grep)
- nmap (port scanner)
- snort, suricata (IDS)
- Bro (IDS)
- ...

API

libpcap API

Specifikace zařízení

- pcapu bude třeba říct, odkud má brát data

```
#include <pcap/pcap.h>
```

```
char errbuf[PCAP_ERRBUF_SIZE];
```

```
int pcap_findalldevs(pcap_if_t **alldevsp, char *errbuf);
```

```
char *pcap_lookupdev(char *errbuf);
```

```
void pcap_freealldevs(pcap_if_t *alldevs);
```

úkol

- Vytiskněte informace o všech síťových interfezech:
 - jméno
 - popis
 - aktuálně přidělenou IP adresu (pokud je)

Capture handler

- 2 režimy: online a offline

```
#include <pcap/pcap.h>
pcap_t *pcap_open_live(const char *device, int snaplen,
    int promisc, int to_ms, char *errbuf);
pcap_t *pcap_open_offline(const char *fname, char *errbuf);
pcap_t *pcap_fopen_offline(FILE *fp, char *errbuf);
pcap_t *pcap_open_dead(int linktype, int snaplen);

int pcap_datalink(pcap_t *p);
int pcap_list_datalinks(pcap_t *p, int **dlt_buf);
void pcap_free_datalinks(int *dlt_list);

void pcap_close(pcap_t *p);
```

Zpracování paketů

- cyklus pro zpracování každého paketu
- buď is napíšete vlastní, nebo použijete cyklus pcapu

Zpracování paketů

- cyklus pro zpracování každého paketu
- buď is napíšete vlastní, nebo použijete cyklus pcapu

```
int pcap_loop(pcap_t *p, int cnt, pcap_handler callback, u_char *user);
int pcap_dispatch(pcap_t *p, int cnt, pcap_handler callback,
                 u_char *user);
void pcap_breakloop(pcap_t *);
```


Zpracování paketů

- cyklus pro zpracování každého paketu
- buď is napíšete vlastní, nebo použijete cyklus pcapu

```
int pcap_loop(pcap_t *p, int cnt, pcap_handler callback, u_char *user);
int pcap_dispatch(pcap_t *p, int cnt, pcap_handler callback,
                  u_char *user);
void pcap_breakloop(pcap_t *);

int pcap_next_ex(pcap_t *p, struct pcap_pkthdr **pkt_header,
                 const u_char **pkt_data);
const u_char *pcap_next(pcap_t *p, struct pcap_pkthdr *h);
```

Zpracování paketů

- cyklus pro zpracování každého paketu
- buď is napíšete vlastní, nebo použijete cyklus pcapu

```
int pcap_loop(pcap_t *p, int cnt, pcap_handler callback, u_char *user);
int pcap_dispatch(pcap_t *p, int cnt, pcap_handler callback,
                  u_char *user);
void pcap_breakloop(pcap_t *);

int pcap_next_ex(pcap_t *p, struct pcap_pkthdr **pkt_header,
                 const u_char **pkt_data);
const u_char *pcap_next(pcap_t *p, struct pcap_pkthdr *h);

void (*pcap_handler)(u_char *user, const struct pcap_pkthdr *h,
                    const u_char *bytes);

struct pcap_pkthdr {
    struct timeval  ts;
    bpf_u_int32    caplen;
    bpf_u_int32    len;
};
```

úkol

- Vytvořte počítadlo paketů
- Vstupním parametrem je název interfacu
- Program počítá pakety procházející interfacem
- Při ukončení (Ctrl+C) je vytištěn počet paketů

Filtry

- BPF (BSD Packet Filter) formát
- biot.com/capstats/bpf.html

Filtry

- BPF (BSD Packet Filter) formát
- biot.com/capstats/bpf.html

```
int pcap_lookupnet(const char *device, bpf_u_int32 *netp,  
                  bpf_u_int32 *maskp, char *errbuf);  
int pcap_compile(pcap_t *p, struct bpf_program *fp,  
                const char *str, int optimize, bpf_u_int32 netmask);  
int pcap_setfilter(pcap_t *p, struct bpf_program *fp);
```

Závěr

domácí úkoly a zdroje

Domácí úkol

DNS analyzátor

- Z DNS paketů (v Ethernetu) zjistěte následující informace
- čas přijetí paketu
- L2:
 - src, dst MAC adresa
- L3 (IPv4 i IPv6):
 - src, dst adresa
 - TTL
- L4 (UDP):
 - src, dst port
- DNS:
 - ID transakce
 - obsah dotazu (queries) – name, type, class
 - obsah odpovědí (answers) – name, type, class, TTL, address

Zdroje

- www.tcpdump.org/
- www.wireshark.org/
- biot.com/capstats/bpf.html