libpcap

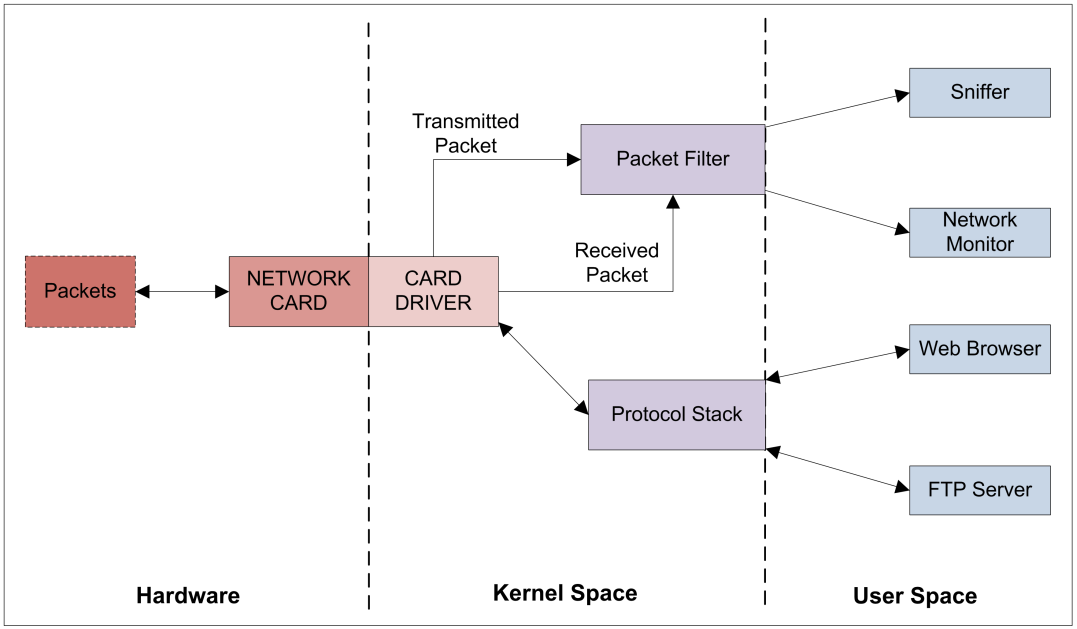
Tématicky zaměřený vývoj aplikací v jazyce C
skupina Systémové programování – Linux

Jiří Novosad

Fakulta informatiky
Masarykova univerzita
novosad@fi.muni.cz

Brno, 17. 11. 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`
- 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/

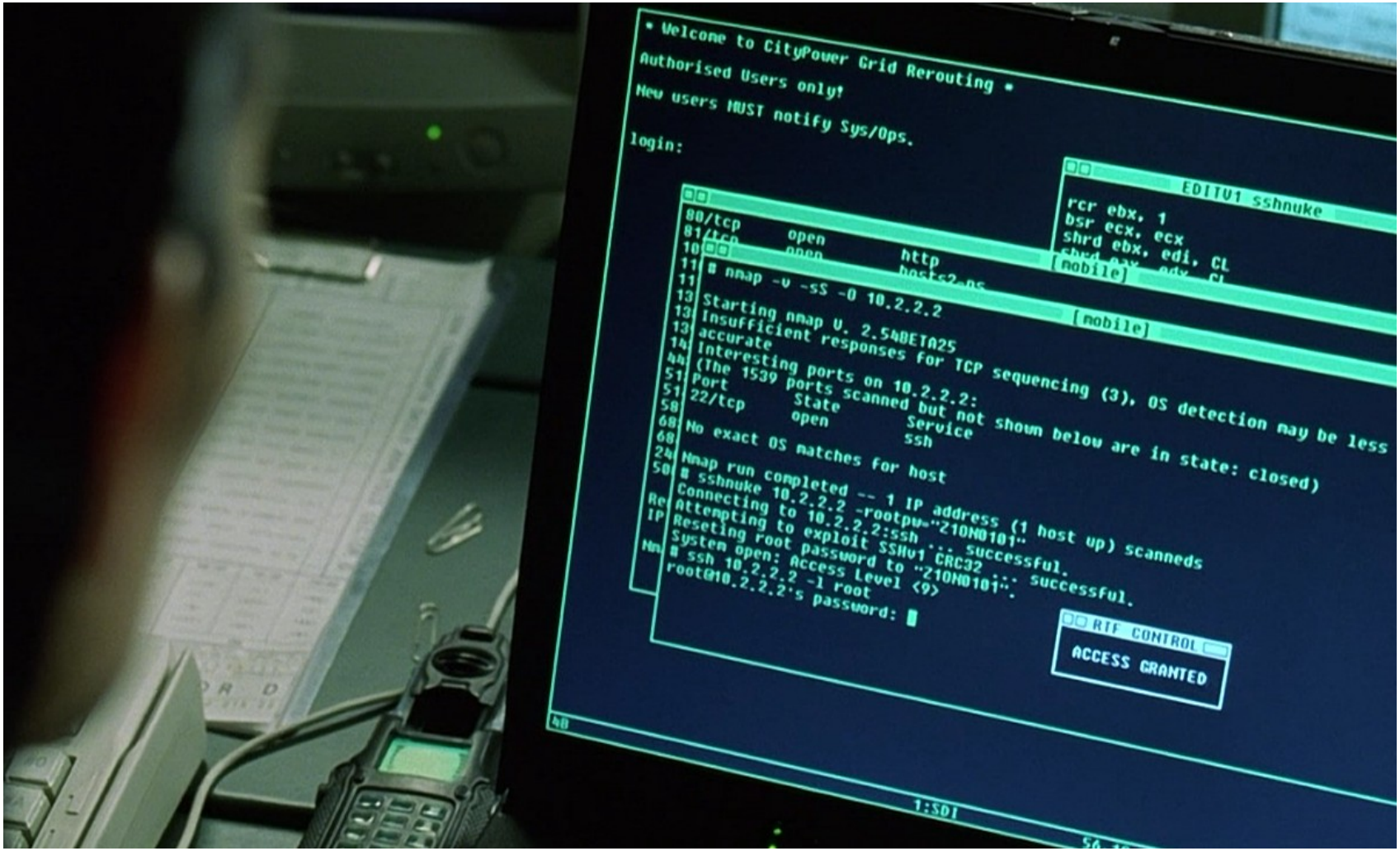
úkol

- otevřete vzorek uložený tcpdumpem a analyzujte provoz

Další aplikace

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

Nmap


* Welcome to CityPower Grid Rerouting *
Authorised Users only!
New users MUST notify Sys/Ops.
login:
EDIT V1 sshnuke
rcr ebx, 1
bsr ecx, ecx
shrd ebx, edi, CL
[mobile]
80/tcp open http
nmap -v -sS -O 10.2.2.2
Starting nmap V. 2.54BETA25
Insufficient responses for TCP sequencing (3), OS detection may be less accurate
Interesting ports on 10.2.2.2:
(The 1539 ports scanned but not shown below are in state: closed)
Port State Service
22/tcp open ssh
No exact OS matches for host
Nmap run completed -- 1 IP address (1 host up) scanneds
sshnuke 10.2.2.2 -rootpw="Z1ON0101"
Connecting to 10.2.2.2:ssh ... successful.
Attempting to exploit SSHv1 CRC32 ... successful.
Reseting root password to "Z1ON0101".
System open: Access Level <9>
ssh 10.2.2.2 -l root
root@10.2.2.2's password:
RIF CONTROL
ACCESS GRANTED
1:SDI

API

libpcap API

Specifikace zařízení

- pcapu je třeba říct, z kterého síťového zařízení má brát data

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

char errbuf[PCAP_ERRBUF_SIZE];

int pcap_findalldevs(pcap_if_t **alldevsp, char *errbuf);
void pcap_freealldevs(pcap_if_t *alldevs);
char *pcap_lookupdev(char *errbuf);
```

úkol

- Vytiskněte informace o všech síťových interfacech:
  - jméno
  - popis
  - aktuálně přidělené IP adresy (pokud jsou)

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);

char *pcap_geterr(pcap_t *p);
void pcap_perror(pcap_t *p, char *prefix);
```

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_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_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);

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_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);

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 *);

void (*pcap_handler)(u_char *user, const struct pcap_pkthdr *h,
                     const u_char *bytes);
struct pcap_pkthdr {
    struct timeval ts;      /* time stamp */
    bpf_u_int32 caplen;     /* length of portion present */
    bpf_u_int32 len;        /* length this packet (off wire) */
};
```

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);
void pcap_freecode(struct bpf_program *);

int pcap_setfilter(pcap_t *p, struct bpf_program *fp);
```

Vyzkoušejte: `tcpdump -d ip and tcp dst port 7`

Závěr

domácí úkoly a zdroje

Domácí úkol

Síťový analyzátor

- Volání: `./sniff eth0`
- Z IP paketů (v Ethernetu) zjistěte následující informace
  - čas přijetí paketu
  - L2: src, dst MAC adresa
  - L3 (IPv4 i IPv6):
    - src, dst adresa
    - TTL (IPv6: hop limit)
    - protokol (UDP/TCP)
  - L4 (UDP i TCP): src, dst port
- Informace vypisujte v reálném čase
- Zachycujte jen IPv4/IPv6 pakety (použijte správný filtr)
- Typ: `netinet/{ether,ip,ip6,tcp}.h`, `ether_ntoa()`, `inet_ntop()`, `ntohs()`

Zdroje

- www.tcpdump.org/
- www.wireshark.org/
- biot.com/capstats/bpf.html
- blog.cloudflare.com/bpf-the-forgotten-bytecode/