

Uno sniffer con la libreria PCAP

Realizzato da:

- Avolio Luca *lucavo@libero.it*
- Domini Angelo *sha.man@tiscalinet.it*
- Listo Massimiliano *maslis@tiscalinet.it*
- Ventre Carmine *ventre@jumpy.it*

10/07/2001

Sniffer con PCAP

1

Che cos'è uno sniffer?

- Le comunicazioni su rete avvengono per la maggior parte mediante il protocollo TCP/IP
- TCP/IP non offre nessun meccanismo di protezione dei dati
- In altre parole i dati viaggiano in chiaro
- Posso quindi leggere tutto ciò che passa sulla rete

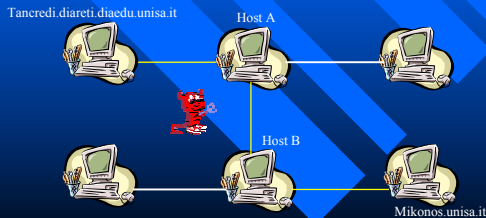
10/07/2001

Sniffer con PCAP

2

Che cos'è uno sniffer? (2)

- I dati viaggiano da origine a destinazione attraverso vari altri nodi della rete



10/07/2001

Sniffer con PCAP

3

Che cos'è uno sniffer? (3)

- Ogni nodo può leggere tali dati ponendo in modalità promiscua la propria scheda Ethernet
 - Per porre in modalità promiscua la propria scheda di rete bisogna avere i permessi di root
 - Posso imporre che tutto il traffico di una rete passi per il mio computer:
 - ✓ Meccanismo ICMP redirect
 - ✓ IP Spoofing
- } conoscenza profonda di TCP/IP e programmazione su reti

10/07/2001

Sniffer con PCAP

4

GeTThings sniffer: scelte progettuali

- Libreria PCAP
Per una semplice e veloce cattura dei pacchetti
- Xforms
Per una semplice e veloce interfaccia user-friendly

10/07/2001

Sniffer con PCAP

5

Il layering TCP/IP

- Assume importanza ricordare la stratificazione TCP/IP per capire come trattare i pacchetti che cattureremo:

Layering concettuale	
applicazione	FTP, HTTP, ecc.
trasporto	TCP, UDP
rete	IP, X25, ecc.
data link	Ethernet, PPP, SLIP, ecc.

10/07/2001

Sniffer con PCAP

6

Gtsnif: potenzialità

- GeTThings può catturare pacchetti in base all'interfaccia su cui ascoltare (eth0, eth1, lo, ecc.)



10/07/2001

Sniffer con PCAP

7

Gtsnif: potenzialità (2)

GeTThings inoltre:

- Può limitare il numero di byte da catturare per ogni pacchetto
 - Ad esempio l'utente vuole i primi 50 byte di ogni pacchetto (anche se il frame è di 1000 byte l'applicazione passerà solo i primi 50 byte all'utente)
- Può filtrare i pacchetti in base al protocollo (arp, rarp, ip, tcp, udp, icmp)
 - Ad esempio potrei voler analizzare solo il traffico arp (comunicazioni ethernet) o solo il traffico icmp (per controllare attacchi tipo ping evil)
- Può filtrare i pacchetti in base alla porta
 - Ad esempio specificando la porta 80 ascolto il traffico web di un host

10/07/2001

Sniffer con PCAP

8

Gtsnif: potenzialità (3)

GeTThings inoltre:

- Può filtrare i pacchetti e catturare solo quelli più piccoli di ... e più grandi di ...
 - Ad esempio specificando pacchetti più piccoli di 60 byte Gtsnif non passerà all'utente i pacchetti del ping (grandi all'incirca 98 byte)
- Può filtrare i pacchetti in base all'host sorgente o destinazione
 - Ad esempio mi interessa conoscere solo il traffico di un host dove opera l'utente di cui voglio conoscere la password
- Può catturare pacchetti "offline"
- Può catturare un certo numero di pacchetti
- Può scaricare i dati scambiati in un file (vedi password)

10/07/2001

Sniffer con PCAP

9

La libreria PCAP: un pò di storia

- Il Packet capturing nasce con l'avvento di Ethernet
- Sun implementò NIT (Network Interface Tap) per catturare pacchetti e *etherfind* per stampare gli header dei pacchetti
- A partire da *etherfind* Van Jacobson ha sviluppato *tcpdump*
- Le parti di programma originariamente prese da *etherfind* sono state successivamente riscritte da McCanne

10/07/2001

Sniffer con PCAP

10

La libreria PCAP: un pò di storia (2)

- *tcpdump* è lo sniffer più popolare nella comunità UNIX
- *tcpdump* è basato su un potente meccanismo di filtro: BSD packet filter (BPF *Berkley Packet Filter*)
- Van Jacobson, Craig Leres e Steven McCanne svilupparono PCAP (Packet Capturing) presso la Lawrence Berkeley National Laboratory
 - PCAP fu scritta per evitare che ci fossero tracce di codice proprietario in *tcpdump*
 - In essa sono implementate le potenzialità per la cattura e il filtro di *tcpdump*

10/07/2001

Sniffer con PCAP

11

La libreria PCAP: un pò di storia (3)



Craig Leres



Steve McCanne

- L'ultima versione rilasciata di PCAP è la 0.6.2
- Il codice sorgente è disponibile a:
 - <ftp://ftp.ee.lbl.gov/libpcap>
 - <ftp://ftp.sysadmin.com/pub/admin/tools/networking/libpcap>

10/07/2001

Sniffer con PCAP

12

Uno sniffer: soluzione naive

- Un primo approccio potrebbe essere:

Sniffa()

1. Setta la scheda di rete in modalità promiscua
2. Mettiti in ascolto
3. Cattura il traffico
4. Visualizza il traffico catturato

10/07/2001

Sniffer con PCAP

13

Soluzione naive: lacune

- Quale interfaccia porre in modalità promiscua?
- Quanti byte del pacchetto devo catturare (solo header o anche dati)?
- Che tipo di traffico devo catturare?
- Come passare i dati dalla rete all'applicazione?



10/07/2001

Sniffer con PCAP

14

Uno Sniffer: soluzione



Algoritmo finale per uno sniffer

10/07/2001

Sniffer con PCAP

15

Uno Sniffer: soluzione con PCAP



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

10/07/2001

Sniffer con PCAP

16

Uno Sniffer: soluzione con PCAP



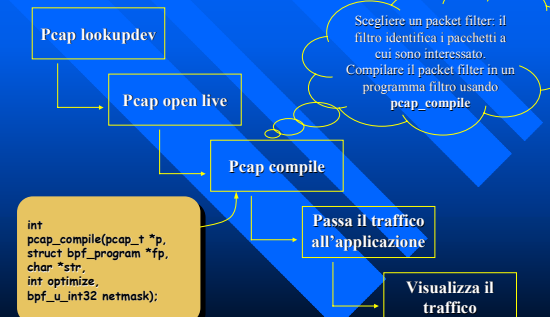
```
Pcap_t *
pcap_open_live(char *device,
int snaplen,
int promisc,
int to_ms,
char *ebuf);
```

10/07/2001

Sniffer con PCAP

17

Uno Sniffer: soluzione con PCAP



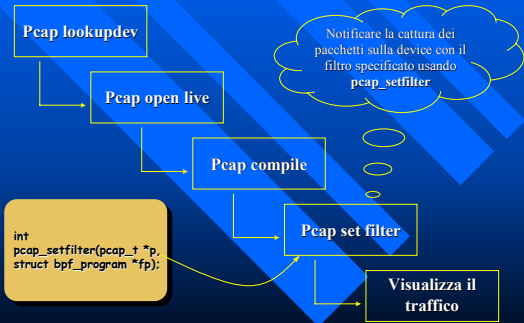
```
int
pcap_compile(pcap_t *p,
struct bpf_program *fp,
char *str,
int optimize,
bpf_u_int32 netmask);
```

10/07/2001

Sniffer con PCAP

18

Uno Sniffer: soluzione con PCAP

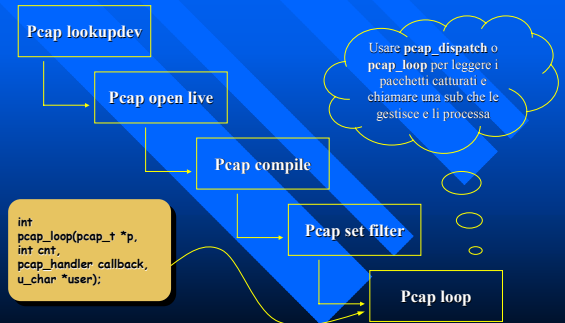


10/07/2001

Sniffer con PCAP

19

Uno Sniffer: soluzione con PCAP

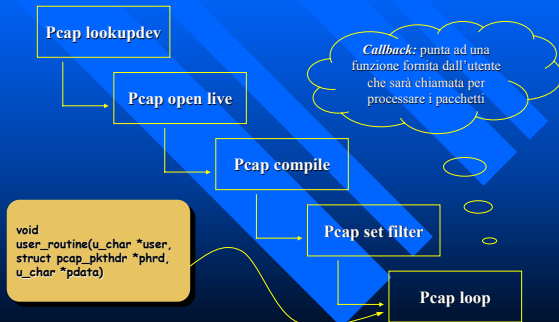


10/07/2001

Sniffer con PCAP

20

Uno Sniffer: soluzione con PCAP

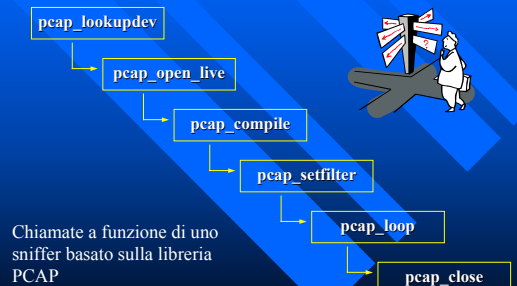


10/07/2001

Sniffer con PCAP

21

Sniffer con PCAP: panoramica

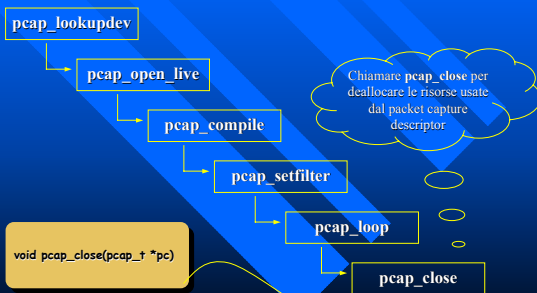


10/07/2001

Sniffer con PCAP

22

Sniffer con PCAP: panoramica



10/07/2001

Sniffer con PCAP

23

Sniffer "offline" con PCAP

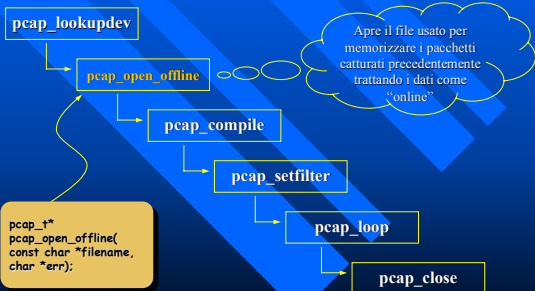


10/07/2001

Sniffer con PCAP

24

Sniffer "offline" con PCAP



10/07/2001

Sniffer con PCAP

25

Sniffer con file di output

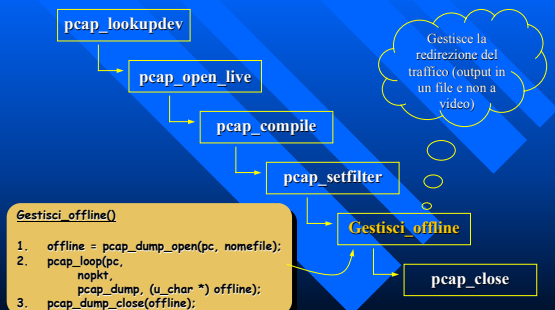


10/07/2001

Sniffer con PCAP

26

Sniffer con file di output



10/07/2001

Sniffer con PCAP

27

Gtsnif: corpo principale

```

    Se ho salvato i pkt in nomefile2... apro lo sniffing offline
    ...altrimenti apro lo sniffing on-line

    if (nomefile2) pc = pcap_open_offline(nomefile2, err);
    else pc = pcap_open_live(device, snaplen, 1, READTO, err);
    // Gestione errori per le chiamate precedenti
    if (pcap_lookupnet(device, &netp, &maskp, err) != 0) {
        printf("Pcap_Lookupnet error %s\n", err);
        return;
    }
    // Utile per compilare il filtro
    // mask è usato ora da pcap_compile
    if (pcap_compile(pc, &fp, filtro, 1, maskp) < 0) { // filtro è passato
        printf("Pcap_Compilare ERROR\n");
        return;
    }
    // Compilo con filtro e costruisco la struttura fp
  
```

10/07/2001

Sniffer con PCAP

28

Gtsnif: corpo principale (2)

```

    if (pcap_setfilter(pc, &fp) == -1) {
        printf("PCAP_setfilter ERROR\n");
        return;
    }
    // Calcolo la lunghezza dell'header in base al tipo di frame
    datalink = pcap_datalink(pc); // funzione che restituisce il tipo di DL
    show_datalink_type(); // lo mostro a video
    header_length = tipo_frame();
    // pcap_loop cicla per nopkt pacchetti
    if (!nomefile) { // se voglio catturare pkt on line
        gestore = (pcap_handler)packetcheck;
        if (pcap_loop(pc, nopkt, gestore, (char *) header_length)) {
            fprintf(stderr, "pcap_loop: %s\n", pcap_geterr(pc));
            exit(1);
        }
        // Chiamo packetcheck passandogli l'header_lenght calcolato prima
    } else { // se voglio catturare pkt off line...
  
```

10/07/2001

Sniffer con PCAP

29

Gtsnif: corpo principale (3)

```

    // Voglio catturare ora ma analizzare poi crea un file in cui memorizzare i pkt
    offline = pcap_dump_open(pc, nomefile);
    if (offline == NULL) {
        fprintf(stderr, "pcap_dump_open: %s\n", pcap_geterr(pc));
        exit(1);
    }
    if (pcap_loop(pc, nopkt, pcap_dump, (u_char *) offline)) {
        fprintf(stderr, "pcap_loop: %s\n", pcap_geterr(pc));
        exit(1);
    }
  
```

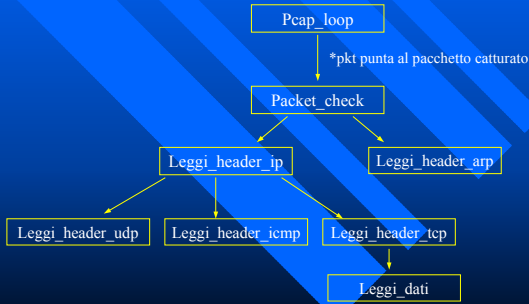
Chiamo pcap_dump come gestore dei pkt catturati e come vuole PCAP il parametro user è in questo caso offline aperto prima

10/07/2001

Sniffer con PCAP

30

Il percorso dei pacchetti catturati



10/07/2001

Sniffer con PCAP

31

Gtsnif: la funzione packetcheck

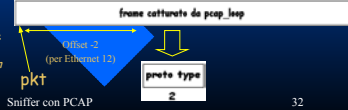
```

void packetcheck(char *user, struct pcap_pkthdr *pkthdr, u_char *pkt) {
    int packlen, caplen, hlen;
    int proto;
    hlen = (int) user;
    proto = protocol_type(pkt, hlen);
    counter++;
    packlen = pkthdr->hlen;
    caplen = pkthdr->caplen;
}
  
```

```

int protocol_type(u_char *pac, int offset) {
    int "ind";
    ind = (int *) &pac[offset - 2];
    return ntohs("ind");
}
  
```

Len è la lunghezza del pacchetto; caplen è la lunghezza del captured packet (la distinzione dipende dal parametro snaplen di pcap_open_live)



10/07/2001

Sniffer con PCAP

32

Gtsnif: la funzione packetcheck

Se il DL è Ethernet e il protocollo è arp (codice 0x0806 vedi RFC)...

```

if ((datalink == DLT_EN10MB) && (proto == 0x0806))
    leggi_header_arp(&pkt[hlen]);
if ((datalink == DLT_EN10MB) && (proto == 0x0800))
    leggi_header_ip(&pkt[hlen], caplen - hlen);
if ((datalink == DLT_EN10MB) && (proto == 0x0835))
    leggi_header_arp(&pkt[hlen]);
if ((datalink == DLT_PPP) && (proto == 0x0021))
    leggi_header_ip(&pkt[hlen], caplen - hlen);
if ((datalink == DLT_PPP) && (proto == 0xc021))
    // Gestisci pkt LCD (Link Control Data)
if ((datalink == DLT_PPP) && (proto == 0x8021))
    // Gestisci pkt NCD (Network Control Data)
}
  
```

Protocolli ip e arp

Stessa gestione per pacchetti PPP

10/07/2001

Sniffer con PCAP

33

Gtsnif: la funzione leggi_header_ip

- La chiamata a leggi_header_ip è del tipo: leggi_header_ip(&pkt[hlen], caplen - hlen);



&pkt[hlen] punta al primo byte del datagramma IP

byte catturati - # byte header frame = len datagramma IP

10/07/2001

Sniffer con PCAP

34

L'header IP

- Usando la struttura iphdr definita in <netinet/ip.h> posso accedere a tutti i campi dell'header IP:

4-bit version	header length	type of service		total length in byte	
		identification	flag	fragment offset	
TTL		protocol	header checksum		
source IP address					
destination IP address					
options					
Data					

10/07/2001

Sniffer con PCAP

35

Gtsnif: la funzione leggi_header_ip (2)

- Lette le informazioni dell'header IP, posso stabilire il protocollo del livello superiore:

```

switch (h_ip->protocol) {
    case 1 : leggi_header_icmp(&pkt[1len]);
              break;
    case 6 : leggi_header_tcp(&pkt[1len], cnt - len);
              break;
    case 17 : leggi_header_udp(&pkt[1len], cnt - len);
}
  
```

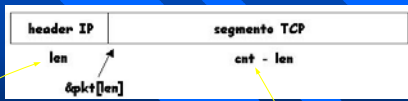
10/07/2001

Sniffer con PCAP

36

Gtsnif: i protocolli di trasporto

- La chiamata a `leggi_header_tcp` è del tipo:
`leggi_header_tcp(&pkt[len], cnt - len);`



La lunghezza dell'header IP è variabile; ma...

$Cnt = caplen - hlen = \text{len datagramma IP}$
 $Cnt - len = \text{len segmento TCP}$

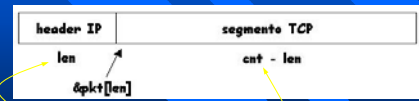
10/07/2001

Sniffer con PCAP

37

Gtsnif: i protocolli di trasporto

- La chiamata a `leggi_header_tcp` è del tipo:
`leggi_header_tcp(&pkt[len], cnt - len);`



`len = h_ip->ihl * 4; // # byte header IP`
`ihl (campo di iphdr) è il # di parole`
`a 32 bit per l'header`

$Cnt = caplen - hlen = \text{len datagramma IP}$
 $Cnt - len = \text{len segmento TCP}$

10/07/2001

Sniffer con PCAP

38

Gtsnif: i protocolli di trasporto (2)

- Posso ora utilizzando le strutture `udphdr` e `tcphdr` contenute in `netinet/tcp.h` e `netinet/udp.h` accedere a tutti i campi degli header
- Stesso discorso vale per i pacchetti ICMP, ARP e RARP (vedere `netinet/ip_icmp.h` e `net/if_arp.h`)
- Ma ho accesso non solo agli header ma anche ai dati allora...

10/07/2001

Sniffer con PCAP

39

Gtsnif: la funzione `leggi_dati`

- ... posso accedere ad essi e magari scaricare i dati in un file (password, traffico, ecc.)

```
void leggi_dati(char *pkt, int cnt) {  
    if (!cnt) return;  
    pkt[cnt] = '\0';  
    fprintf(dati, "%s", pkt);  
}
```

pkt punta al primo byte dei dati

cnt è la lunghezza della parte dati del pacchetto catturato

10/07/2001

Sniffer con PCAP

40

Gtsnif all'opera

- Vediamo Gtsnif al lavoro in uno scenario di questo tipo:



10/07/2001

Sniffer con PCAP

41

Gtsnif all'opera (2)

- L'obiettivo è scoprire la password dell'ignaro utente `angdom` alle prese con il non sicuro FTP:



Viene settata l'opzione `-d` per avere accesso ai dati dei pacchetti catturati

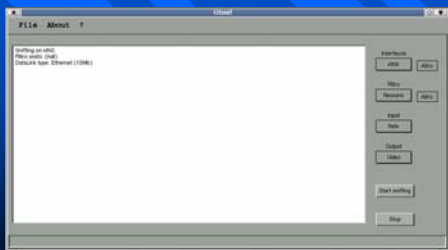
10/07/2001

Sniffer con PCAP

42

Gtsnif all'opera (3)

- Cliccando su start parte lo sniffing:



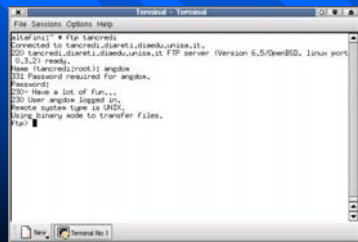
10/07/2001

Sniffer con PCAP

43

Gtsnif all'opera (4)

- Intanto l'utente angdom su altafini richiede il servizio FTP a tancredi...



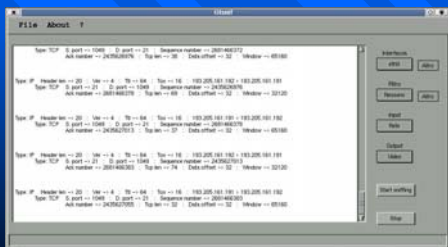
10/07/2001

Sniffer con PCAP

44

Gtsnif all'opera (5)

- ... ma Gtsnif sta catturando:



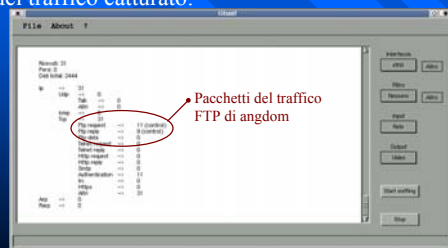
10/07/2001

Sniffer con PCAP

45

Gtsnif all'opera (6)

- Cliccando su stop Gtsnif comunica le statistiche del traffico catturato:



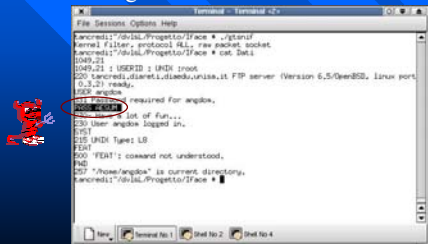
10/07/2001

Sniffer con PCAP

46

Gtsnif all'opera (7)

- Contenti dell'operato di Gtsnif si può conoscere la password di angdom consultando il file dati:



10/07/2001

Sniffer con PCAP

47