Sniffit

Giuseppe Di Francesco

10/07/2009

Abstract

Sniffit è un software scritto in C per il monitoraggio di rete. E' uno sniffer capace di catturare pacchetti live da una qualsiasi interfaccia passatagli come parametro oppure legge da un file pcap contenente pacchetti catturati in precedenza. I pacchetti catturati vengono analizzati e visualizzati a video. Ci sono due tipologie di analisi differenti; la prima consiste nell'estrapolare informazioni dai pacchetti sniffati e inserirli nel database rrdtool, in questo modo otteniamo delle statistiche sui tipi di pacchetti che passano per la rete; ad esempio il numero totale di pacchetti, distinguendoli tra tcp, udp e icmp. Sniffit si limita ad ascoltare soltanto questi tre tipi di pacchetti grazie all'uso di un filtro BPF presente nella libreria libpcap utilizzata per la cattura. La seconda tipologia di analisi consiste nell'aggregazione e la visualizzazione di pacchetti dello stesso tipo, flussi; i flussi vengono creati raggruppando pacchetti che hanno le stesse caratteristiche, come il tipo di protocollo, l'indirizzo ip sorgente, l' indirizzo ip del destinatario, la porta sorgente e la porta del destinatario.

1 ARCHITETTURA

Sniffit è l'implementazione in linguaggio C di uno sniffer, che utilizza la libPcap.

E' stato realizzato in multi-thread, in questo modo alcune funzionalità possono essere eseguite in parallello, rendendo il software scalabile. Il primo thread ad essere creato si occupa della lettura dei pacchetti (pcap_loop()), questa chiama ciclicamente la funzione process_Packet(), è proprio questa funzione che si occupa di leggere i datagrammi , decodificarli, stamparli a video e inoltrarli in una delle code; la coda viene identificata tramite una chiave hash che accomuna il tipo di protocollo, e altre informazioni del messaggio. La stessa funzione verrà usata per ottenere l'entry della tabella hash;

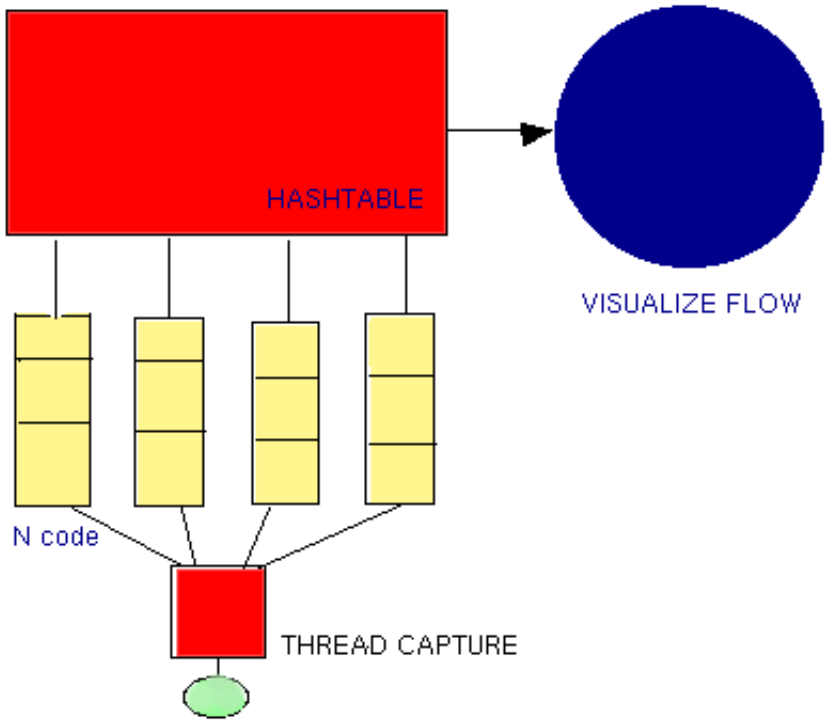
Ad ogni coda è associato un thread. Se la coda non è vuota il thread :

- legge i messaggi;
- crea il flusso;
- lo inserisce nella tabella hash;

altrimenti si sospende; sarà poi risvegliato all'inserimento del nuovo messaggio o allo scadere di un quanto di tempo.

L'ultimo thread ad essere creato ha il compito di scorrere l'intera tabella ed eliminare i flussi scaduti, i quali verrano stampati a video.

L'eliminazione dei flussi è un operazione che necessita la mutua esclusione a differenza dell'inserimento in cui la mutua esclusione è garantita dalla funzione hash e dalla dimensione della tabella; allo stesso istante lo stesso flusso può essere aggiornato, i due thread tenteranno di accedere alla stessa posizione; per la sincronizzazione ho introdotto gli spinlock; quest'ultimi utilizzano un ciclo di attesa attiva per cui sono utili per operazioni brevi, cioè operazioni che in media sono minori del tempo di due contex_switch.



1.1 Strutture dati

Le strutture dati utilizzate sono: code e tabella hash.

Il numero di code N è parametrico come anche la creazione dei thread associati, in questo modo si aumenta la scalabilità del programma; default N=4.

La dimensione della tabella hash è multipla del numero di code, in questo modo si garantisce che l'accesso alla tabella hash è esclusivo per ogni thread; la funzione hash garantisce che ogni pacchetto dello stesso tipo (protocollo, ip sorgente, ip destinatario, porta sorgente, porta destinatario) andrà sempre nella stessa coda e quindi sarà sempre trattato dallo stesso thread che accederà in modo esclusivo alle entry della tabella hash. Questo fa si che la tabella hash è l'equivalente di più tabelle hash distribuite, e inoltre non è necessario l'uso di semafori per accedervi.

La struttura del messaggio che andrà inserito nelle code è il seguente:

```
typedef struct msg {
   int protocol;            /* tipo di protocollo di trasporto */
   char *s;                 /* indirizzo sorgente formattato */
   char *d;                 /* indirizzo destinatario formattato */
   unsigned short source;   /* porta sorgente */
   unsigned short dest;     /* porta destinatario */
   in_addr_t ip_src;        /* indirizzo ip sorgente */
   in_addr_t ip_dst;        /* indirizzo ip destinatario */
   int packet_bytes;        /* # bytes */
   time_t timestamp;        /* timestamp del pacchetto */
}msg;
```

1.2 Sniffer

Il programma analizza la linea di comando, il nome del file in cui scrivere o da cui leggere i pacchetti e l'opzione che specifica se eseguire una sessione on line oppure off line. Il software se avviato in modalità live legge i pacchetti dall'interfaccia desiderata altrimenti legge da un file pcap passatogli come parametro.

L'intercettazione dei singoli pacchetti avviene in modalità promiscua, decodificando le varie intestazioni di livello datalink, rete e trasporto.

1.3 Analisi Statistiche

Sniffit tiene traccia del numero totale di pacchetti distinguendoli tra Tcp, Udp e Icmp. Queste informazioni vanno inserite nel database rrdtool di tipo round-robin; questo database viene aggiornato periodicamente disegnando un grafico dei pacchetti intercettati (my.png) .

1.4 Flussi

Sniffit cataloga i pacchetti in flussi bidirezionali. Un flusso è rappresentato nel modo seguente:

```
typedef struct flow{
  char *s;                  /* indirizzo ip sorgente */
  char *d;                  /* indirizzo ip destinatario */
  int protocol;             /* protocollo */
  int pkt_b_s;              /* numero di byte  dal sorgente al destinatario */
  int pkt_c_s;              /* numero di pkt dal sorgente al destinatario*/
  int pkt_b_d;              /* numero di byte  dal destinatario al sorgente */
  int pkt_c_d;              /* numero di pkt dal destinatario al sorgente*/
  int dir;                  /* direzione del pkt  0 è dal sorg al dest */
  unsigned short ps;        /* porta sorgente */
  unsigned short pd;        /* porta destinatario */
  time_t timestamp_first;   /* timestamp al momento della cattura*/
  time_t timestamp_last;    /* timestamp al momento dell'aggiornamento */
}FLOW;
```

Quando un flusso è scaduto verrà rimosso dalla tabella hash e stampato a video; un flusso è scaduto quando è inattivo da 60 secondi oppure quando è attivo da troppo tempo.

Tuttavia i flussi scaduti invece di essere stampati a video potrebbero essere scritti su un file oppure mandati in remoto tramite socket, per esempio ad un web server, il quale li potrebbe elaborare e fare altri tipi di analisi, come ad esempio la loro localizzazione geografica, oppure semplicemente analizzare il traffico in remoto.

1.5 Miglioramenti

Il software potrebbe essere migliorato su alcuni aspetti; ad esempio sarebbe piu stabile se l'aggiornamento al database rrd venisse eseguito da un processo figlio duplicato tramite fork, in questo modo se la scrittura su disco non andasse a buon fine sarebbe solo quel processo a bloccarsi, lasciando indisturbato il resto del lavoro di cattura, seguendo questa linea sarebbe necessario creare una shared memory che tenesse conto del numero di pacchetti.

Inoltre la sincronizzazione nelle code potrebbe essere realizzata secondo una implementazione lock-free.

1.6 Versione 2.0

Nella versione 2.0 è stata utilizzata la libreria libpcap_1.0-ring(PF-RING) al posto della libreria libpcap.