Note ulteriori sullo sviluppo di IWTAN: Wireless Topology Analyzer.

Questo documento compensa quanto già illustrato nella IWTAN User Reference, fornendo informazioni sui dettagli implementativi seguiti nello sviluppo della libreria.

Leggibilità e usabilità del codice

Per mantenere il codice pulito, facilmente aggiornabile e integrabile in altri programmi, si è cercato di seguire le convenzioni di sviluppo delle librerie C, sebbene queste non siano uniformi. Ad esempio, come convenzione, si è scelto di chiamare con nomi che iniziano con un underscore tutte le funzioni ad uso interno (quelle che gli utenti non dovrebbero chiamare se non per fare debug), ad esempio quelle per processare i pacchetti. Inoltre si è scelto di far cominciare tutti i nomi delle funzioni con la stringa "`iwtan`", in modo da separare lo spazio dei nomi da quello delle funzioni di altre librerie.

I due file header definiscono anche una `#define` come `IWTAN_DATA_H`, definita a `1` la prima volta che la libreria viene importata. Questo serve per evitare che due file .c che utilizzano la medesima libreria e sono linkati tra loro diano errori (definizioni doppie) durante la compilazione.

Aspetti interessanti sull'uso del linguaggio C

Dal punto di vista della programmazione, uno dei paradigmi più interessanti utilizzati è quello dei puntatori a `struct` che definiscono il contenuto dei pacchetti e frame ai vari livelli (radiotap, wlan, ip, ecc). La chiamata della funzione `pcap_next()` infatti restituisce un puntatore ad una stringa di byte del pacchetto. Per analizzare questa stringa sarebbe bastato conoscere a priori la posizione di ogni elemento nel pacchetto. Ad esempio, in un radiotap header possiamo sapere a priori (dalla definizione dei Radiotap header) che la frequenza è contenuta nel byte di indice 17, in cui il valore della frequenza è espresso in MHz. Dunque, conoscendo la stringa `pktBody`, avremmo potuto trovare il valore in della frequenza leggendo il byte `*(pktBody + 17)`, il che avrebbe evidentemente reso inutilmente complesso il codice. Per quanto riguarda i campi composti da più di un byte, il problema si sarebbe amplificato. Per questo si è preferito sfruttare la potenza dei puntatori C, definendo delle strutture che rispecchiano la disposizione dei byte nei vari tipi di frame (si vedano le strutture definite in `iwtan_analyze.h`). I compilatori C assicurano che la memoria riservata alle strutture venga allocata in modo contiguo e nello stesso ordine di come sono state definite le strutture. Per questo, con un semplice cast di `pktBody` da `char*` a `radiotap_header*`:

```
radiotap_header* rt = (radiotap_header*) pktBody
```

sara' possibile accedere ai campi del pacchetto sfruttando la struttura, e dunque per accedere al byte della frequenza si potrà utilizzare l'espressione:

```
rt->frequency
```

Implementazione della portabilità

La portabilità della libreria su piattaforme diverse da quella su cui è stata sviluppata dovrebbe essere garantita dall'uso di tipi di dato definiti in `<types/bits.h>` la cui dimensione è fissa indipendentemente dalla piattaforma, come `uint_16` (intero a 2 byte), `uint_8` (intero a 1 byte). Per garantire il funzionamento del casting dei pacchetti tramite i puntatori a strutture, è necessario utilizzare questi tipi di dati invece dei regolari `short`, `int`, `char`, per i quali non è universalmente garantita la dimensione.

Un altro accorgimento che è stato adottato riguarda la disposizione dei byte negli interi di diversi byte, ad esempio gli `uint_16`. Infatti alcuni sistemi rappresentano in memoria questi interi con la notazione little endian, mentre altri usano la big endian. Inoltre, diversi tipi di pacchetti utilizzano diverse notazioni, ad esempio IPv4 e IPv6 utilizzano big endian per esprimere la lunghezza del payload, mentre i Radiotap header, per esprimere la frequenza, utilizzano little endian.

Per questo è stato necessario definire le funzioni `_iwtan_le_to_host()` e `_iwtan_be_to_host()` che prendono come argomento un numero di 2 byte in notazione rispettivamente little endian e big endian e ne restituiscono il valore nella notazione dell'host.

Per implementare queste funzioni si sono utilizzate le `#define __BIG_ENDIAN_BITFIELD` e `__LITTLE_ENDIAN_BITFIELD`, definite in `<asm/byteorder.h>` (vedi implementazione di `_iwtan_le_to_host()` in `iwtan_analyze.h`)

Struttura del codice

Il caso più significativo di strutturazione del codice è quello che riguarda l'analisi dei pacchetti/frame. Come sappiamo, in generale ogni frame ne contiene un altro. Un esempio possibile è quello di un radiotap header che contiene un WLAN frame, che a sua volta contiene un Datalink layer frame, che contiene un pacchetto IP, che contiene un pacchetto TCP, che a sua volta trasporta un dato HTML.

Questa struttura non è deterministica (un pacchetto IP non contiene sempre TCP, e un WLAN frame non contiene sempre un pacchetto di dati), per cui il codice di tutte le funzioni di analisi dei frame sono strutturate in due fasi: nella prima estraggono i dati relativi a quel frame (ad esempio, in un pacchetto WLAN estraggono il MAC address di destinazione e sorgente), e nella seconda fase cercano di capire quale è il tipo di pacchetto contenuto e chiamano, sulla porzione di pacchetto ancora non analizzata, la funzione di analisi opportuna. Ovviamente per alcuni tipi di frame non esiste la prima o la seconda fase, ad esempio i radiotap header contengono sempre WLAN frame, mentre all'analisi dei pacchetti IP non si cerca neanche di capire quale protocollo è usato a livello di trasporto visto che ad IWTAN non interessano i dati contenuti ai livelli superiori.

Testing e sviluppi futuri

Durante lo sviluppo, ogni funzione è stata testata con dei programmi che facessero da driver per i test, in modo da testare la coerenza tra i pacchetti analizzati e i dati inseriti nel contesto. Questi test sono stati eseguiti anche in ambiente valgrind per verificare che non ci fosse memory leakage.

Per questo le funzioni create dovrebbero essere perfettamente funzionanti. Tuttavia è possibile che alcune di queste contengano degli errori non riscontrati in fase di testing.

Il testing inoltre ha tralasciato, per motivi di tempo, diversi aspetti tra cui:

- La portabilità del programma non è mai stata testata su piattaforme diverse da quella i386 a 32 bit.
- Non è stato testato il traffico e l'estrazione di indirizzi IPv6 da frame wireless.
- Non è stata testato il funzionamento della libreria con i diversi crittaggi delle trasmissioni wireless come WEP e WPA. In effetti dovrebbe essere l'utente a preoccuparsi di decrittare i frame prima di farli processare a IWTAN, tuttavia IWTAN dovrebbe implementare dei controlli in grado di capire se la chiave utilizzata per la decrittazione è corretta. Per questo si può utilizzare una feature del protocollo WLAN che prevede un campo destinato al controllo CRC. IWTAN potrebbe occuparsi di controllare questo campo per verificare la coerenza col resto del frame. Per questo è stata predisposta ma non ancora implementata la funzione `iwtan_validate_802_11()`.

Dunque uno dei principali sviluppi futuri è quello del debug dei problemi di cui sicuramente è

affetto il codice, in modo da renderlo più robusto. Oltre a questo, si possono pensare delle aggiunte alle funzionalità. Ad esempio, per come funziona adesso, la libreria aggiunge dati ma non li toglie mai. Si potrebbe pensare ad una funzione che, se invocata, elimina tutte le stazioni e gli access point di cui non si ha traccia da più di N minuti (tali elementi possono essere individuati sfruttando il campo `lastSeen`). Si possono anche aggiungere, a seconda delle esigenze, delle funzioni per il browsing dei dati, ad esempio una `iwtan_get_station_by_IP()`, che restituisce una stazione avente il dato IP address.

Materiale consultato

Il libro consultato per quanto riguarda la programmazione C è stato principalmente:

Marc J. Rochkind – **Advanced UNIX Programming (2nd edition)** – Addison-Wesley professional computing series

Per quanto riguarda le reti e la struttura dei pacchetti:

James F. Kurose, Keith W. Ross – **Reti di Calcolatori e Internet (3rd edition)** – Pearson Addison-Wesley

Altri testi e documenti trovati su web sono stati fondamentali per la comprensione di diversi aspetti.

The Wireshark Wiki – Wi-Fi - http://wiki.wireshark.org/Wi-Fi

The Wireshark Wiki - WLAN Capture Setup - http://wiki.wireshark.org/CaptureSetup/WLAN

Byte order conversion functions - http://safari.oreilly.com/1565922921/quiteconvert

AP Architecture Thoughts IEEE 802.11-04/1191r5 (Mike Moreton, STMicroelectronics) - http://www.ieee802.org/1/files/public/docs2004/liaison-11-ap-architecture-thoughts-1104.ppt

IPv6 addressing (RFC 2373) - University of Cambridge - http://www-uxsup.csx.cam.ac.uk/courses/ipv6_basics/x13.html

IEEE Standards: MAC address assignations. - http://standards.ieee.org/regauth/oui/oui.txt

The Radiotap header format - http://www.radiotap.org/Radiotap

RFC 2460 - Internet Protocol, Version 6 (IPv6) Specification - http://www.faqs.org/rfcs/rfc2460.html

RFC 791 - Internet Protocol - http://www.faqs.org/rfcs/rfc791.html