Buongiorno sono Orioli Stefano e vi presento la tesi: SISTEMI FLESSIBILI PER LA CATTURA DI TRAFFICO BLUETOOTH

quello di cui vi parlerò oggi è riassunto nei seguenti punti:

inizierò con una breve introduzione alla tecnologia bluetooth low energy con un accenno ai principali campi di applicazione ed agli utilizzi più comuni, ponendo particolare attenzione alle tematiche della sicurezza che riguardano questi dispositivi. Vedremo poi il sistema nano versione 2 della società RedBear e come può essere utilizzato per creare uno Sniffer multicanale di pacchetti BLE. Infine vi parlerò di cosa è stato effettivamente realizzato e come il progetto possa evolvere in futuro.

Il bluetooth low energy è una tecnologia di comunicazione wireless a corto raggio ed a ridotto consumo energetico che opera nella banda di frequenza ISM a 2,4 GHz. La specifica Low Energy è stata introdotta nella versione 4.0 dal SIG, ovvero lo Special Interest Group, che è l’organizzazione che sovraintende lo sviluppo di questa tecnologia. In queste frequenze di trasmissione coesistono molte comunicazioni anche di altri standard trasmissivi, come per esempio il più comune Wi-FI; è quindi una banda soggetta a interferenze. Per ridurre queste interferenze e per avere quindi un canale che sia il più pulito possibile, il bluetooth low energy divide la banda trasmissiva in 40 canali, 3 dei quali utilizzati per fare Advertise con un formato dei pacchetti standard e noto a priori e 37 dedicati allo scambio di dati, con dei pacchetti personalizzati per ogni connessione. Per lo scambio dati si utilizza un canale trasmissivo diverso per ogni pacchetto che si invia, cambiandolo fino a 1600 volte al secondo. Inoltre, grazie alla funzionalità di Adaptive Frequency Hopping, questo cambio di canale è adattivo, cioè va ad escludere quei canali che sono troppo disturbati da altre comunicazioni.

Come sappiamo il bluetooth è largamente implementato nei dispositivi che utilizziamo tutti i giorni, dagli smartphone ai dispositivi dell’IoT, e ogni giorno se ne fa un uso sempre maggiore in svariati campi applicativi. Da qui nasce la necessità di creare uno strumento di test economico che sia in grado di intercettare i pacchetti inviati, da utilizzare come dispositivo di debug per accertare che venga trasmesso quello che è stato teoricamente progettato e per scovare quindi eventuali malfunzionamenti aumentando la sicurezza delle trasmissioni, potendo verificare cosa realmente viene scambiato.

Seguire una connessione bluetooth, senza conoscerne i parametri di comunicazione, richiede di essere in ascolto contemporaneamente su tutti i 37 canali dedicati. Per svolgere questo compito si possono utilizzare dispositivi costosi come due USRP B210 dal costo di 1200€ l’una, e di un PC di ultima generazione con porte USB 3.0 che riesca a memorizzare l’enorme flusso di dati generato dagli USRP. È possibile utilizzare anche un dispositivo leggermente più economico, il FrontLine BPA Low Energy, il cui costo si aggira attorno ai 900€ ma che riesce a catturare tutti i pacchetti BLE solamente avendo prima intercettato la richiesta di connessione e tramite interfacciamento con un sistema Windows, di memorizzarli ed analizzarli.

Per sviluppare uno Sniffer economico che riesca a catturare anche tutti i pacchetti di una connessione, avendo a disposizione i parametri della stessa, è stato utilizzato il prodotto di punta della RedBear, il Nano in versione 2, con un costo che si aggira attorno ai 30$. Questo dispositivo, nonostante il costo ridotto, Integra il System On a Chip della Nordic, l’nRF52832 che dispone un processore ARM a 32 bit, è uno dei migliori presenti sul mercato che supporti il protocollo BLE. Viene dotato di un ulteriore dispositivo di supporto, chiamato DAPLink, che tramite un’interfaccia USB permette di programmare facilmente il Nano, utilizzando un PC, e di dialogare con quest’ultimo tramite HW dedicato, come UART e SPI per lo scambio dati in modalità seriale.

Lo sviluppo dello sniffer è stato effettuato in ambiente UNIX utilizzando Ubuntu versione 17 ed Eclipse come IDE di supporto alla scrittura del codice in linguaggio C. Sono state utilizzate anche altre periferiche a supporto dello sviluppo, come i prima citati USRP B210 e il Frontline BPA Low Energy, dispositivi molto più costosi del nano ma che si sono rivelati necessari per analizzare nel dettaglio come vengono trasmessi i pacchetti nell’etere.

I primi pacchetti ad essere catturati sono stati quelli di Advertise, ovvero quei pacchetti che un dispositivo bluetooth usa per notificare la propria presenza ai dispositivi vicini; essi sono trasmessi solo in 3 canali, il 37 38 e 39 con un formato standard e quindi sono più facili da catturare. Questi pacchetti, anche se molto contenuti come dimensioni, danno parecchie informazioni sulle intenzioni del dispositivo: notificano la disponibilità o meno ad una connessione da parte del dispositivo, l’indirizzo di chi li invia, e altre informazioni contenute nel payload che possono essere di varia natura, dal semplice nome del dispositivo, ad una panoramica dei servizi offerti. Analizzando uno dei primi pacchetti di Advertise catturati, possiamo ricavare dal primo Byte, che nell’esempio ha il valore di 40 esadecimale, che è un Advertise di tipo Indirect, ovvero che è disponibile ad una connessione con un qualsiasi dispositivo, e che trasmette un indirizzo privato. Possiamo già dedurre che sia un dispositivo in attesa di connessione per offrire servizi specifici, come ad esempio mouse o tastiera di un pc. Il secondo Byte indica la lunghezza del payload, ovvero le informazioni contenute nel pacchetto. In questo caso tale Byte vale 0C, quindi 12 Byte di lunghezza. Dopodiché si ha l’indirizzo dell’Advertiser stesso, lungo 6 Byte, che sappiamo da prima essere privato, ovvero non il reale indirizzo della periferica Bluetooth ma uno generato pseudo-casualmente, per ragioni di sicurezza. Dopodiche, si ha un Byte di un ulteriore campo length, del valore di 5, che questa volta indica la lunghezza della sezione dati del pacchetto; Il valore FF specifica il tipo dei dati comunicati, che ad esempio in questo caso sono del tipo Manufacturer Specific Data, e infine gli ultimi 4 Byte contengono i dati veri e propri.

La vera sfida è stata catturare il pacchetto di CONNECT\_REQ, ovvero quel pacchetto che da avvio ad una connessione e che contiene le informazioni necessarie per poter catturare tutti i pacchetti della stessa. I campi fondamentali di questo pacchetto, che vengono utilizzati per poter catturare i dati scambiati in fase di connessione, sono L’access Address, il campo CRCInit, il campo Interval ed infine la costante di Hop, contenuta negli ultimi 5 bit del pacchetto.

Dopo essere riusciti a catturare ed analizzare questo pacchetto, si è potuto saltare con le tempistiche corrette tra i vari canali ed intercettare così tutti i pacchetti di una connessione. Un esempio di cattura di una connessione è mostrato nella slide, in cui la comunicazione inizia nel canale 37 con l’invio del pacchetto di connection Request da parte del Central; si entra poi nello stato di connessione e quindi ci si sposta fra i vari canali, dapprima il canale 14 passando per il 28 e così via ogni 900 ms. Si nota che per ogni canale vengono inviati almeno 2 pacchetti, uno dal master che è sempre il primo a parlare in una comunicazione e la risposta dello Slave, necessaria per mantenere la connessione.

Con questo lavoro di tesi è stato quindi creato uno Sniffer multicanale economico, ma che riesce ad ascoltare anche la fase di connessione tra due dispositivi. Questo Sniffer può essere utilizzato in futuro per implementare vari attacchi volti a testare la sicurezza del Bluetooth Low Energy, come l’attacco Man in the Middle che consente ad un attaccante di rubare e/o modificare le informazioni scambiate o l’attacco di Relay del segnale a lunga distanza, che fa credere a 2 dispositivi di essere vicini e in grado di comunicare, anche se in realtà sono molto distanti fra loro.

ho concluso, grazie per l’attenzione.