Universitá Tor Vergata

Corso di Laurea di Ingegneria Informatica

A.A. 2016-2017

Mobile Sniffing Tool

Progetto di Sicurezza Informatica e Internet

Autori:
Paolo Salomé
Stefano Agostini

Supervisori:
Prof. Giuseppe Italiano
Dr. Marco Querini

January 12, 2017

Contents

1 Introduzione

Nella societá odierna vi é un larga diffusione di applicazioni mobile di messagistica che espongono implicitamente gli utenti a problematiche riguardanti la privacy. Infatti fino a qualche mese fa i dati in transito di Whatsapp (e.g.) erano in chiaro e quindi accedibili facilmente da chiunque utilizzasse una qualsiasi applicazione per lo sniffing (e.g. Whireshark). Tuttavia attualmente molte delle applicazioni di messaggistica hanno rimediato interamente o parzialmente utilizzando tecniche di crittografia end-to-end. L'obiettivo del nostro progetto é la realizzazione di una applicazione per sistemi mobile Android che all'interno di una certa rete wifi catturi i dati in transito e li filtri in base ad un flag. In tal modo ci preponiamo l'obiettivo di isolare i pacchetti dati per ogni applicazione di messaggistica presa in considerazione e analizzarli.

2 Descrizione dell'Applicazione

Per realizzare uno *sniffer Android* c'erano due strade alternative da poter seguire: sfruttare le *Android VPN* oppure utilizzare l'eseguibile *C tcpdump*. L'approccio *Android VPN* consiste nella creazione di un'interfaccia *VPNService*, gestita da una applicazione in *userspace*, che una volta attivata forza tutto il traffico del *device* ad attraversarla. Tuttavia questo approccio non permette la cattura di pacchetti appartenenti a dispositivi diversi da quello ospitante l'applicazione. Per utilizzare il secondo approccio é necessario che il dispositivo abbia i permessi di *root* in quanto verrá eseguito uno *script bash tcpdump*, il quale si occuperá di catturare i pacchetti. Inoltre questa libreria permette di sfruttare la scheda di rete in uso in varie modalitá (e.g. *promiscous mode, monitor mode*) qualora il dispositivo lo consenta. La possibilitá di sfruttare a pieno le potenzialitá della scheda di rete ci ha indotto a scegliere *tcpdump*.

2.1 Tcpdump

Il comando *bash* dell'eseguibile *tcpdump* permette l'inserimento di alcuni parametri che permettono la visualizzazione dei pacchetti in vari formati. Nel caso della nostra applicazione utilizziamo i seguenti parametri:

- **-i** : seguito dal nome dell'interfaccia, per specificare dove porsi in ascolto (e.g. wlan0)
- **-XX** in alternativa a **-A**: il primo stampa l'header di ogni pacchetto e i dati in esadecimale e *ASCII* mentre il secondo non stampa l'esadecimale
- **-tttt**: stampa la data corrente davanti l'header di ogni pacchetto in formato *YYYY-MM-DD hh:mm:ss:dddddd*

Di seguito inseriamo un pacchetto di esempio stampato da bash come risultato del comando *sudo tcpdump -i wlan0 -XX -tttt* (figure 1) e del comando *sudo tcpdump -i wlan0 -A -tttt* (figure 2).

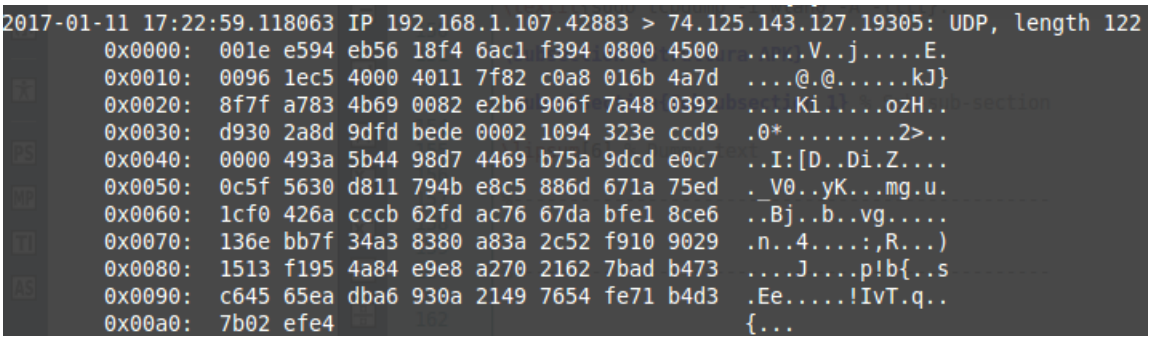


Figure 1: Con opzione -XX

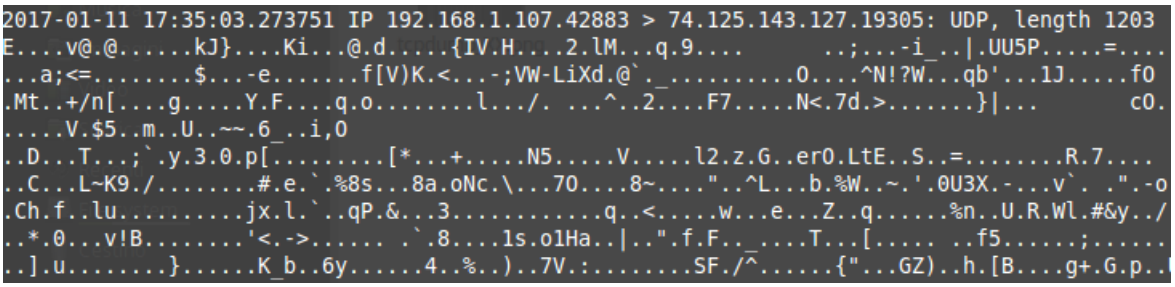


Figure 2: Con opzione -A

2.2 Struttura APK

La nostra *APK* si compone di due *Activity* e due *Service*:

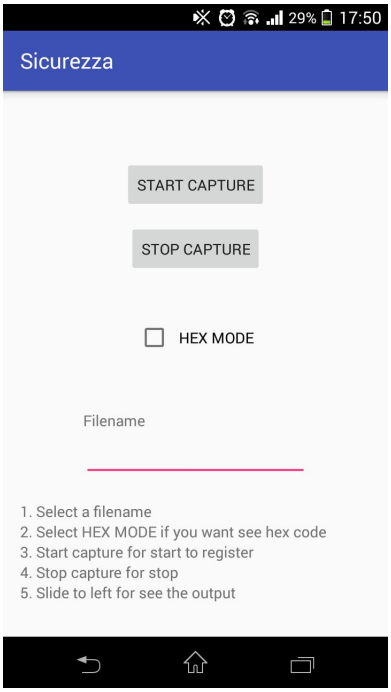
- L'*Activity* di cattura espone un'interfaccia semplice per impostare la modalitá di cattura (specificare il *flag hex mode*). Successivamente l'utente puó specificare il nome del file sul quale vuole che vengano salvati i pacchetti. Infine possibile attivare il servizio di cattura.
- L'*Activity* di filtraggio si occupa di fornire all'utente un elenco dei file contenenti i pacchetti, salvati dall'applicazione stessa nelle precedenti catture. Selezionato il file da elenco possibile inserire la parola chiave di ricerca. Mediante il servizio di filtraggio é vengono scanditi i pacchetti e visualizzati in una lista soltanto quelli contenenti la parola desiderata.
- Il *Service* di cattura si occupa dell'invocazione del comando *tcpdump* con le opzioni e il nome del file passati dall'*Activity* di cattura, redirezionando l'output su quest'ultimo. Quando questo servizio viene interrotto si esegue il comando bash *pkill tcpdump* per terminare il comando dello *sniffer*.
- Il *Service* di filtraggio agisce sul file selezionato esaminando ogni pacchetto e inserendolo in una lista visualizzata a schermo solo se contiene la *keyword* fornita, all'interno dell'*header* o del *body*.

Di seguito inseriamo degli *screen* dell'applicazione appena descritta (figure 3).

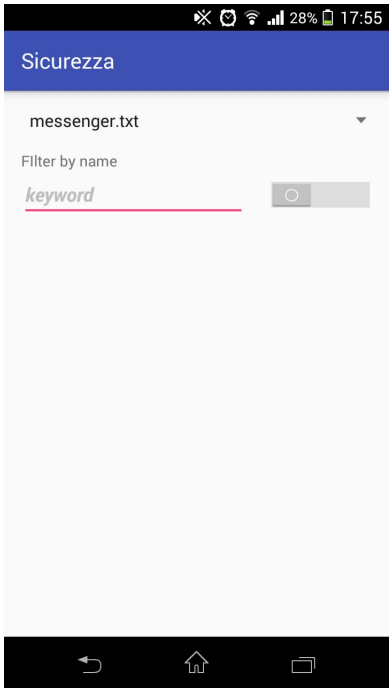
2.2.1 Activity di cattura

L'*Activity* di cattura implementata dalla classe *MainActivity* e realizza la schermata iniziale per l'utente. Essa estende la classe *AppCompatActivity* ed effettua l'*override* dei metodi *onCreate* e *onTouchEvent*. Quest'ultimo implementa un *listener* che, a seguito del primo tocco dell'utente, esegue le seguenti operazioni:

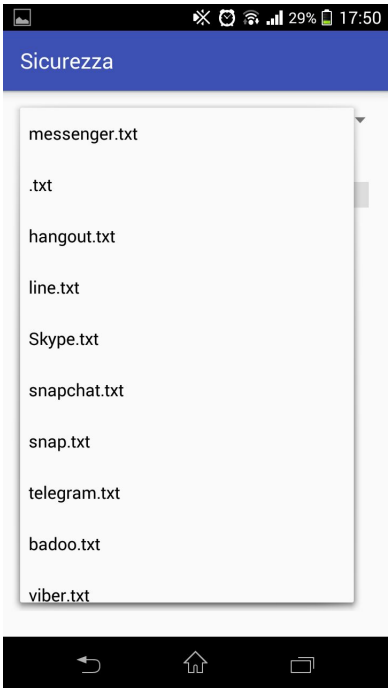
- Salva la posizione iniziale al tocco.
- Al rilascio calcola il *delta* rispetto alla posizione iniziale.
- Se il *delta* é maggiore di 400 *pixel* verso sinistra invoca l'*Activity* di filtraggio.



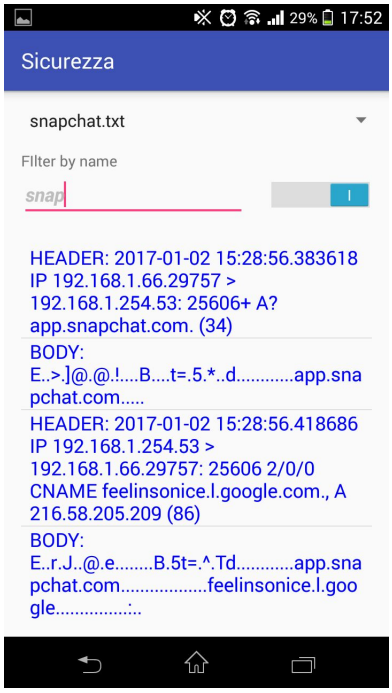
(a) Activity di cattura



(b) Activity di filtraggio



(c) Lista di file



(d) Pacchetti filtrati

Figure 3: Pagine dell'applicazione

Infine ci sono i due metodi legati ai bottoni di *start* e *stop* del servizio di cattura. Il metodo *startclick* esegue i seguenti passi all'invocazione:

- Cerca il riferimento all'*editText* contenente il nome del file in cui salvare i pacchetti catturati.
- Verifica lo stato della *checkbox* relativa alla modalitá *hex mode*.
- Crea l'*Intent* da passare al servizio di cattura.
- Utilizza il metodo *putExtra* dell'*Intent* per passare i parametri di cattura al servizio (*path* del file e eventualmente il *flag hex*).

Il metodo *stopclick* crea un nuovo *Intent* per il servizio di cattura avviato precedentemente e invoca il metodo *stopService* per forzarne l'interruzione.

2.2.2 Activity di filtraggio

Questa *Activity* é implementata dalla classe *SecondActivity* la quale estende la classe *AppCompatActivity*. Il metodo *OnCreate* effettua le seguenti operazioni:

- Recupera lo *switch* relativo all'attivazione del servizio di filtraggio.
- Inizializza lo stato dello *switch* a *false*.
- Fa l'*override* del metodo *OnCheckedChanged*, il quale rappresenta il *listener* dello *switch*. Quest'ultimo:
  - Se lo *switch* é *checked* inizializza la *ListView* che apparirá a schermo e l'*Adapter* ad essa associato. Successivamente invoca il metodo *viewFile* di cui parleremo in seguito.
  - Se lo *switch* é *unchecked* invoca il metodo *stopServ*.
- Preleva tutti i file relativi alle precedenti registrazioni di pacchetti fatti dall'applicazione. I file sono raggruppati in una cartella sul dispositivo.
- Prelevati i file questi vengono inseriti in una lista di stringhe. Quest'ultima verrá associata con un *adapter* ad uno *spinner* per consentire all'utente la selezione di un file.

Il metodo *viewFile* effettua le seguenti operazioni:

- Preleva il nome del file dallo *Spinner* ed il filtro.
- Crea l'*Intent* relativo al servizio di filtraggio ed aggiunge come parametri il *filepath* ed il *filter*.
- Invoca il metodo *startService* con parametro l'*Intent* appena creato.

Il metodo *stopServ* crea un nuovo *Intent* per il servizio di filtraggio avviato precedentemente e invoca il metodo *stopService* per forzarne l'interruzione.

Questa *Activity* utilizza un *BroadcastReceiver* per ricevere i dati elaborati dal servizio invocato. Di questo *Receiver* viene fatto l'*override* del metodo *onReceive* il quale, in ricezione dei dati inviati dal servizio, aggiorna la *ListView* presente nell'*Activiy* stessa. Per utilizzare il *BroadcastReceiver* col metodo *onReceive* esso deve essere registrato mediante il metodo *onResume*, il quale accetta un *flag* per identificare gli *Intent* da gestire (nel nostro sistema *filterMessage*). D'altro canto il metodo *onPause* serve a deregistrare il *BroadcastReceiver* quando l'*Activity* é in pausa.

Infine é implementato il metodo *onTouchEvent* che si comporta esattamente come quello dell'*Activity* precedentemente illustrata.

3 Conclusion

Etiam euismod. Fusce facilisis lacinia dui. Suspendisse potenti. In mi erat, cursus id, nonummy sed, ullamcorper eget, sapien. Praesent pretium, magna in eleifend egestas, pede pede pretium lorem, quis consectetuer tortor sapien facilisis magna. Mauris quis magna varius nulla scelerisque imperdiet. Aliquam non quam. Aliquam porttitor quam a lacus. Praesent vel arcu ut tortor cursus volutpat. In vitae pede quis diam bibendum placerat. Fusce elementum convallis neque. Sed dolor orci, scelerisque ac, dapibus nec, ultricies ut, mi. Duis nec dui quis leo sagittis commodo.

Aliquam lectus. Vivamus leo. Quisque ornare tellus ullamcorper nulla. Mauris porttitor pharetra tortor. Sed fringilla justo sed mauris. Mauris tellus. Sed non leo. Nullam elementum, magna in cursus sodales, augue est scelerisque sapien,

venenatis congue nulla arcu et pede. Ut suscipit enim vel sapien. Donec congue. Maecenas urna mi, suscipit in, placerat ut, vestibulum ut, massa. Fusce ultrices nulla et nisl.

References

[Figueredo and Wolf, 2009] Figueredo, A. J. and Wolf, P. S. A. (2009). Assortative pairing and life history strategy - a cross-cultural study. *Human Nature*, 20:317–330.