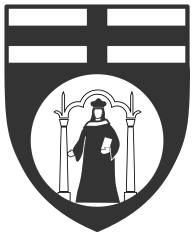
UNIVERSITÀ DEGLI STUDI DI GENOVA
Facoltà di Ingegneria



Corso di Laurea Magistrale in Ingegneria Informatica

Analisi di librerie di user profiling su Android

Relatore:
Prof. Alessio Merlo

Correlatore:
Dott. Davide Caputo

Candidato:
Mattia Parrinello
Matricola n. 3775905

Anno Accademico 2018 – 2019

Indice

Acronimi

API Application Programming Interface

ART Android RunTime

EI Explicit Identifiers

GDPR General Data Protection Regulation

HAL Hardware Abstraction Layer

NDK Android Native Development Kit

QI Quasi-identifiers

SD Sensitive Data

SDK Software Development Kit

UGACLIP Utility-Guided Anonymization of CLInical Profiles

Elenco delle figure

Elenco delle tabelle

1 Introduzione

Nel 2019, report statistici affermano che l'80% della popolazione mondiale possiede uno smartphone[1], che il mercato del mobile banking è in costante crescita [1] e che più di 3,5 miliardi di persone utilizzano i social network (Facebook, Instagram o Twitter) rimanendo connessi a internet mediamente per 6 ore al giorno [2]. Il mercato mobile è in costante crescita e molte aziende hanno quindi deciso di dedicare risorse e investimenti all'espansione delle politiche di advertising su smartphone. Aziende con profitti multimilionari come Amazon e Netflix, hanno introdotto schemi pubblicitari e funzionalità che si adattano alle abitudini del cliente, costruendo il così definito "alias digitale" della propria utenza. Si elaborano algoritmi e si raccolgono dati per poter plasmare le piattaforme, in base ai gusti e agli usi e costumi della persona che ne fruisce, con l'obiettivo di massimizzarne i guadagni.

Tutto questo è possibile grazie a sistemi di profilazione che ogni giorno operano attraverso l'utilizzo di applicazioni nel mondo mobile o navigando su internet per il mondo web. Per quanto concerne il settore degli smartphone, attraverso l'utilizzo di librerie di profilazione (Application Programming Interface (API)) create appositamente per il mercato di riferimento (es., Android), è possibile raccogliere e ottenere un elevato numero di dati su abitudini, preferenze o desideri dei propri clienti. Google, Facebook, Yahoo mettono a disposizione degli sviluppatori di applicazioni mobili, API specifiche per la profilazione degli utenti permettendo la visione, attraverso l'utilizzo di dashboard e schede informative web-based, di informazioni su quanto e come vengano utilizzate le loro applicazioni. Essendo proprietarie di tali librerie, le aziende legittimano la proprietà digitale di tali informazioni arricchendo i loro database con i profili creati dalle applicazioni che ogni giorno vengono sviluppate, distribuite e scaricate dai market di riferimento come il Google Play Store[3]. Si ritiene opportuno capire e contestualizzare quanto tale profilazione possa essere considerata "legittima" o se presenti, nel suo operare, vulnerabilità sotto il profilo della privacy. Perchè se da una parte Android mette a disposizione della propria utenza la possibilità di poter abilitare o meno l'accesso da parte delle applicazioni ad elementi del dispositivo (come GPS, la galleria multimediale oppure la fotocamera), dall'altra non vengono mai indicate le possibilità che la appli-

[1] `https://www.corrierecomunicazioni.it/digital-economy/`
[2] `https://www.studiosamo.it/social-media-marketing/global-digital-2019-statistiche-social/`
[3] `https://play.google.com/store?hl=it`

cazione possiede nel tracciare parametri che ne caratterizzano l'utilizzo. La soluzione che viene proposta e discussa in questo elaborato, da questo punto in avanti definita come *Hidedroid*, opera su due fronti:

- Coinvolgere l'utente nello stabilire un livello di condivisione dei propri dati e delle proprie informazioni, questo perché come unico vero proprietario dei suoi dati personali dovrebbe essere coinvolto attivamente nel prendere decisioni sulla sua privacy.

- Valutare quanto la raccolta di dati e la profilazione degli utenti possa rappresentare un importante contributo per migliorare i prodotti e servizi che l'azienda può offrire. Si può affermare come l'utente stesso possa giovare di queste "indagini di mercato" effettuate, notando cambiamenti del servizio o del prodotto verso quelle funzionalità che più rispecchiano le sue esigenze.

Verranno quindi proposte due possibili progettazioni strutturali di Hidedroid, ne verranno evidenziate le potenzialità e criticità, e successivamente sarà valutato un primo approccio applicativo nell'analisi delle API sopracitate. Studiando circa 50'000 applicazioni è stato possibile elaborare una struttura implementativa che possa, in maniera automatica, individuare le librerie, i metodi e i parametri che all'interno delle applicazioni vengono utilizzati per scopi profilativi e rappresentano un rischio per la privacy dell'utenza. I risultati ottenuti verranno poi utilizzati per stabilire una base di partenza per lo sviluppo di Hidedroid.

1.1 Struttura della Tesi

La tesi viene strutturata nel seguente modo: dopo aver descritto una breve introduzione sul lavoro che si intende svolgere vengono illustrate la metodologia sviluppata e gli strumenti utilizzati per la sua implementazione, infine vengono esposti i risultati e le considerazioni finali ed eventuali sviluppi futuri.

La Sezione 2 si occupa dell'analisi dell'architettura del sistema operativo Android esponendo la struttura dei file che caratterizzano le applicazioni.

La Sezione 3 presenta una panoramica sulle librerie più influenti in ambito di profilazione dell'utenza secondo i risultati estratti durante lo sviluppo del lavoro.

La Sezione 4 si occupa di esporre le tematiche della Data Privacy e della Anonymization andando a studiare in maniera approfondita le tipologie di dato che possono essere interessate da particolari tecniche di anonimizzazione.

La Sezione 5 presenta l'esposizione dell'attuale stato dell'arte.

La Sezione 6 presenta una panoramica sulle possibili implementazioni di Hidedroid delineandone la roadmap e le fasi di sviluppo.

La Sezione 7 si occupa di andare ad argomentare sotto il profilo metodologico e implementativo il lavoro svolto durante il periodo di tesi.

La Sezione 8 espone e giustifica i risultati raccolti durante lo sviluppo della tesi.

2 Android e Smali

2.1 Struttura Android

Android, nel 2017, ha conquistato uno dei titoli più ambiti da chi sviluppa sistemi operativi ovvero il premio come sistema operativo più utilizzato al mondo, sorpassando anche Windows, sistema Desktop che vive nei personal computer della maggior parte delle persone al mondo. Era il 2003 quando Andy Rubin, Rich Miner, Nick Sears e Chris White fondarono Android Inc. il cui scopo era quello di sviluppare "dispositivi cellulari più consapevoli della posizione e delle preferenze del loro proprietario". Ma è nel 2005 che avviene la svolta. Con l'acquisizione da parte di Google per una cifra vicina ai 50 milioni di dollari, Android inizia a diventare un sistema operativo per dispositivi mobili, alla cui base risiede un kernel Linux. Dopo 3 anni di intenso sviluppo, nell'Ottobre del 2008 viene presentato il primo Android su un HTC Dream.

Android è basato su un'architettura a più livelli di astrazione, come visibile in Fig. 1.

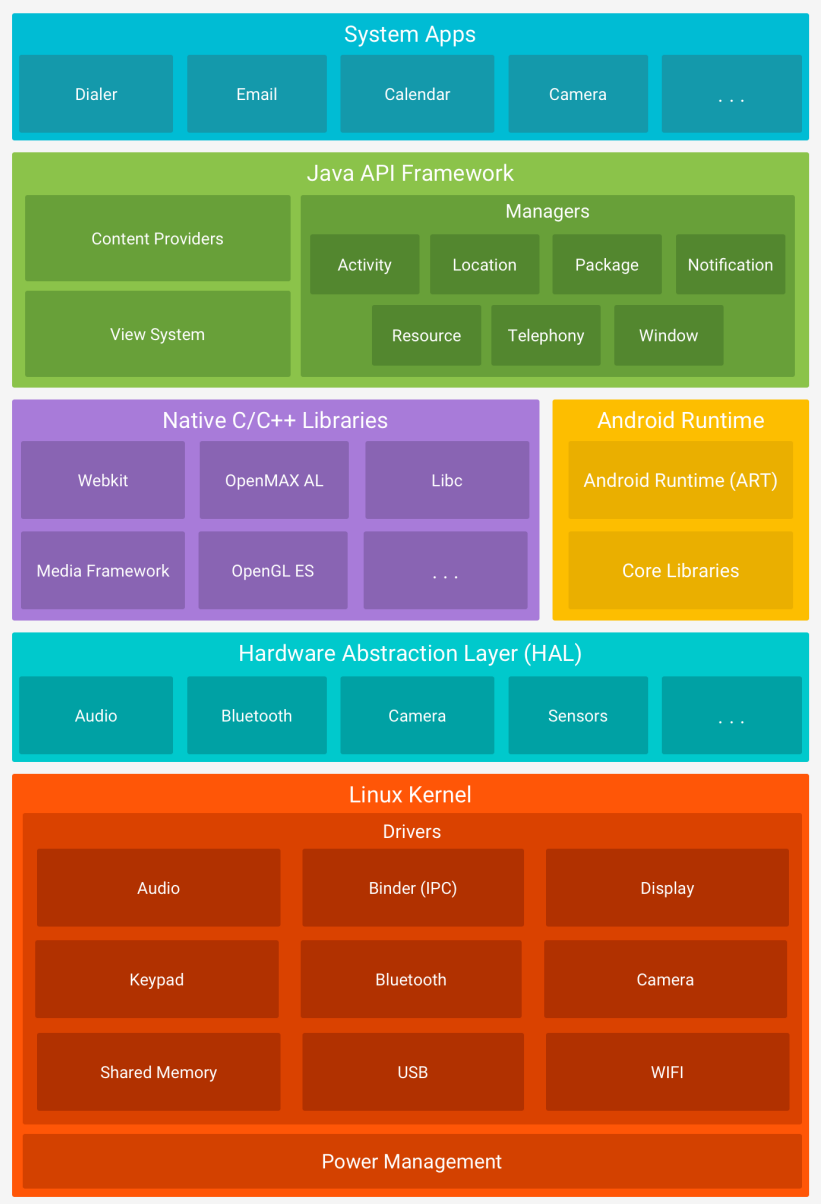
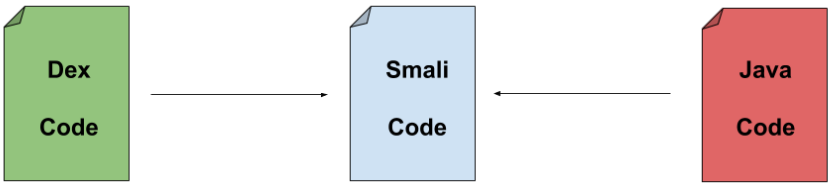


Figura 1: Struttura Android

Il livello più basso è costituito dal **Kernel**. Qui sono presenti tutti i driver che permettono di interfacciarsi con l'hardware fornito dal dispositivo. Inoltre è qui che si trovano tutte le componenti necessarie alla gestione energetica. Sopra di esso è presente l'**Hardware Abstraction Layer (HAL)**. Questo offre un'interfaccia standard che espone le funzionalità hardware del dispositivo ai livelli superiori, in particolare al framework Java. Esso è composto da una serie di moduli, ognuno dei quali implementa un'interfaccia per uno specifico componente hardware, come possono essere i sensori, la fotocamera o il modulo Bluetooth. Quando un'API produce una chiamata per accedere ad un componente hardware, il sistema carica il modulo associato. Il terzo livello è composta dalle **Librerie Native**. Le librerie native consistono in librerie scritte in C/C++, quindi codice nativo, eseguito direttamente, senza dover passare prima dal runtime (ART o Dalvik). Alcune componenti della piattaforma sono scritte in codice nativo, hanno quindi necessità di tali librerie. Queste, inoltre, sono accessibili anche dal programmatore attraverso alcune API del framework. Se invece c'è bisogno di accesso diretto, si possono scrivere porzioni di codice in C/C++ e sfruttare l'Android Native Development Kit (NDK). Il livello più importante per quanto riguarda le argomentazioni esposte in questo elaborato è sicuramente il livello delle **API Java**. Queste sono API scritte in Java che espongono al programmatore tutte le funzioni che il sistema operativo offre. Grazie a loro abbiamo tutto il necessario per sviluppare le nostre app ed accedere a tutto ciò che offrono i livelli inferiori dello stack, senza preoccuparci della loro implementazione. Qui troviamo gli strumenti per: costruire le interfacce delle nostre app; accedere alle risorse, utilizzare le notifiche, gestire il ciclo di vita e la navigazione delle app, accedere ai dati di altre app o per condividere dati con altre app.

2.2 Smali

Quando si scrive una applicazione Java per Android il codice sorgente Java viene compilato in un file `.class` che converte il codice Java in Java Byte Code. Attraverso il tool *dx* presente all'interno del Software Development Kit (SDK) di Android, il file `.class` e le sue librerie `.jar` annesse vengono trasformate in un unico file `.dex` scritto in Dalvik Byte Code. [4]. Sarà poi compito dell'Android RunTime (ART) [5], compilare, in fase di installazione, il codice. Lo SMALI non è altro che una rappresentazione intermedia tra codice Dalvik Bytecode e il codice sorgente Java/Kotlin.



Pertanto quando si va ad analizzare un file `.apk`, esplorandone il contenuto si noteranno quelli che sono i suoi files `.dex` contenente bytecode Dalvik. Questi file risultano illegibili e sarà necessario, in qualche modo, "decompilarli" (o tradurli) in codice SMALI.

Supponendo quindi di avere in Java il seguente codice:

int x = 42

E assumendo che questa sia la prima variabile istanziata, la sua rappresentazione in Dalvik Bytecode andrebbe a contenere la seguente riga esadecimale (hex)

13 00 2A 00

Trasformando il Dalvik Bytecode in SMALI si otterrà

const/16 v0, 42

che risulterà ai nostri occhi molto più leggibile rispetto al codice binario mostrato in precedenza.

[4] `https://source.android.com/devices/tech/dalvik/dalvik-bytecode`
[5] `https://source.android.com/devices/tech/dalvik`

3 Librerie Analytics

3.1 Firebase

Firebase è una piattaforma di sviluppo web e mobile nata nel 2011 e acquistata da Google nel 2014. Questa piattaforma offre numerosi servizi e risulta di semplice implementazione. È possibile integrarla in siti web, all'interno di applicazioni Android o iOS e in videogiochi. Offre la possibilità di semplificare numerose impostazioni che agli sviluppatori risultano di difficile implementazione. A questo si aggiunge l'importante opzione di analisi dati per reagire in modo coerente alle impressioni degli utenti aumentandone il flusso o servendosi della pubblicità per trarne profitto[6]. Tra le librerie e servizi offerti troviamo:

- **Authentication** : grazie alla quale è possibile, in pochi minuti creare un portale di login con possibilità di collegarsi con servizi esterni molto famosi tra cui Twitter e Facebook e ovviamente Google.

- **Cloud Messaging**: con il cloud messaging è possibile creare una linea di comunicazione tra chi ha i dispositivi connessi al servizio e chi li fornisce. L'amministratore può infatti decidere di inviare messaggi a tutti i device, a gruppi di device o a uno singolo. In questo modo, usufruendo dell'analisi dati, è possibile pilotare gli utenti verso ciò che più li aggrada.

- **Machine Learning**: il machine learning con le sue potenzialità viene messa al servizio di tutti. Nonostante sia in continua ottimizzazione e stabilizzazione, questo servizio permette di apportare numerose migliorie. Parliamo, nello specifico, di auto-riconoscimento del testo, dei volti, della classificazione delle immagini, della scansione dei codici QR e a barre.

- **Advertising**: Firebase permette di effettuare il collegamento con AdMob in modo estremamente veloce e di verificare gli introiti direttamente accedendo alla console con le proprie credenziali.

- **Analytics**: la libreria più importante, cardine dell'intero lavoro di questa tesi. Questo servizio permette di tenere traccia di eventi e di proprietà appartenenti

[6] `https://www.chimerarevo.com/guide/internet/firebase-305791/`

alla applicazione grazie a una dashboard completa e facilmente accessibile da parte dello sviluppatore. Consente di monitorare le più diverse tipologie di dato come le regioni in cui la app è stata più scaricata, il numero di volte che è stata aperta, quante volte è stato cliccato un determinato bottone o si è aperto un sito web. Si possono creare oltre 500 eventi completamente personalizzati di cui tenere traccia. Possono essere collezionate proprietà dell'utente quali età , paese di provenienza , lingua , interessi come Arte, Cucina, Sport ...

3.1.1 Firebase Analytics

Firebase Analytics mette a disposizione un particolare tipo di API utilizzata per il raccoglimento di eventi che possono essere standard, ovvero definiti dalla libreria, oppure custom cioè liberamente definibili dallo sviluppatore dell'applicazione. Un esempio di come utilizzare questa libreria è mostrato in Codice Sorgente 1.

```
Bundle bundle = new Bundle();
bundle.putString(FirebaseAnalytics.Param.ITEM_ID, id);
bundle.putString(FirebaseAnalytics.Param.ITEM_NAME, name);
bundle.putString(FirebaseAnalytics.Param.CONTENT_TYPE, "image");
mFA.logEvent(FirebaseAnalytics.Event.SELECT_CONTENT, bundle);
```

Listing 1: Codice funzione logEvent in Firebase Analytics

In questo specifico esempio viene esposta la procedura per poter registrare un evento dal nome chiave SELECT_CONTENT (impostato come parametro key) al quale vengono associati l'ITEM_ID, ITEM_NAME e CONTENT_TYPE dell'immagine selezionata (impostati come parametro bundle). Questa coppia di dati rappresenta l'evento che in questo caso viene registrato e inserito all'interno di quella che potrebbe essere una tabella di profilazione degli eventi. In questo caso è stata utilizzata una chiave costante, presente quindi nella libreria di FirebaseAnalytics ma è possibile anche inserire arbitrariamente un numero di parametri key, registrando fino ad un massimo di 500 eventi secondo i limiti della libreria. Si pone in considerazione il fatto che utilizzando in maniera impropria questo tipo di strumento ci possa essere una sorta di profilazione non corretta che possa andare in qualche modo a minacciare la privacy dell'utente che utilizza la applicazione.

3.2 Facebook Analytics

È la nuova piattaforma Analytics di Facebook, annunciata al F8 (aprile 2017, in California). Diretta concorrente di Firebase Analytics ha il compito di tenere traccia di moltissimi dati statistici sugli utenti utilizzatori della piattaforma. La libreria proprietaria offre agli sviluppatori molte categorie per meglio contestualizzare il proprio target commerciale di utenza, grazie anche ai milioni di dati che l'azienda di Zuckenberg raccoglie ogni giorno sui propri "clienti". Facebook Analytics dà la possibilità di analizzare come gli utenti interagiscano con le applicazioni, in modo più dettagliato di quanto avveniva in passato e in ottica multi canale: viene presa in considerazione anche l'interazione con siti web, coi bot di messenger ed eventuali pagine Facebook. Con metodologia personalizzabile è possibile tenere traccia di tutto ciò che riguarda il customer journey di ogni utente: con il termine customer journey si intende l'insieme delle tappe compiute dal consumatore nel suo "viaggio" verso l'acquisto.

Facebook dispone di informazioni molto dettagliate sui suoi utenti, e può quindi dare agli sviluppatori, un aiuto nell'approfondire gli aspetti di profilazione dei visitatori.

Anche Facebook Analytics, così come è stato presentato in Sezione 3.1.1 per Firebase, presenta una API analoga dedicata all'analisi degli eventi che avvengono durante l'utilizzo della applicazione. Così come in Codice Sorgente 1, ci troviamo davanti a una chiamata strutturata in questo modo (vedere Codice Sorgente 2):

```
Bundle params = new Bundle();
params.putString(AppEventsConstants.EVENT_PARAM_CURRENCY, "USD");
params.putString(AppEventsConstants.EVENT_PARAM_CONTENT_TYPE, "product");
params.putString(AppEventsConstants.EVENT_PARAM_CONTENT, "[{\"id\": \"1234\",
\"quantity\": 2}, {\"id\": \"5678\", \"quantity\": 1}]");

logger.logEvent(AppEventsConstants.EVENT_NAME_PURCHASE,
                54.23,
                params);
```

Listing 2: Codice funzione logEvent in Facebook Analytics

Anche qui come si può notare, alla funzione *logEvent*, vengono passati parametri utili alla registrazione di un evento specifico, secondo la forma **logEvent(key, value)**. E così come Firebase Analytics esiste la possibilità di registrare eventi personalizzati.

4 Data Privacy e Anonimizzazione

Per poter comprendere e contestualizzare la privacy dei dati e le tecniche di anonimizzazione che su tali dati vengono eseguite e che verranno esposte nei capitoli successivi, si ritiene opportuno fissare alcuni termini e concetti. Gli attributi che compongono uno specifico dato possono essere catalogati in tre categorie:

- **Explicit Identifiers (EI)** : Attributi che identificano unicamente un individuo, includendo per esempio un codice fiscale oppure un nome.
- **Quasi-identifiers (QI)** : Attributi che includono informazioni geografiche e demografiche, numeri di telefono, email e altro. Vengono anche definiti come quella tipologia di attributo che viene resa pubblica, per esempio, attraverso un database di votanti elettorali.
- **Sensitive Data (SD)** : Attributi che contengono informazioni confidenziali come per esempio condizioni finanziarie o problemi di salute, che non possono in alcun modo essere compromessi.

L'anonimizzazione è un processo logico che separa le informazioni di identificazione (PII) dai dati sensibili. Ad esempio EI e QI vengono logicamente separati dagli SD. Se sotto le condizioni di privacy abbiamo conoscenza dell'identità di un individuo ma non delle caratteristiche a cui esso sono legate, in condizioni di anonimità si ha conoscenza delle caratteristiche ma non dell'individuo a cui sono associate. Si pone l'obiettivo di ottenere una de-identificazione irreversibile. Nel momento in cui i dati personali riferiti ad un individuo sono stati adeguatamente anonimizzati, dovrebbe essere impossibile poter invertire il processo. L'anonimizzazione è uno strumento che consente la condivisione dei dati, garantendo sia la *privacy* che l'*utility* che questi dati forniscono. L'anonimizzazione si può realizzare tramite la rimozione, la sostituzione, la distorsione, la generalizzazione o l'aggregazione degli EI, come il nome completo o altre caratteristiche rilevanti dell'individuo, e indiretti (QI), cioè attributi che combinati con altre informazioni disponibili rendono identificabile un individuo, come per esempio una combinazione di occupazione, stipendio ed età.

Uno dei metodi più comuni per anonimizzare i dati comporta l'eliminazione degli identificatori diretti (EI). Questa singola operazione non è però sufficiente a garantire che l'identificazione della persona interessata non sia più possibile. Per questo si

consiglia di non utilizzare un'unica tecnica di anonimizzazione, ma una combinazione di esse. All'eliminazione degli EI si può aggiungere a titolo esemplificativo la tecnica della generalizzazione, la quale comporta la riduzione del grado di dettaglio di una determinata variabile. Per esempio, le date di nascita di singole persone fisiche possono essere generalizzate per mese o anno, producendo una riduzione del grado di identificabilità. Quindi, eliminare i nomi completi degli individui, mantenendo solo l'anno di nascita degli stessi, permette di de-identificare in modo irreversibile le persone fisiche, potendo comunque effettuare analisi statistiche sul campione di dati. Ne viene esposto un esempio con la Figura 2

Figura 2: Esempio di Anonimizzazione

Esiste poi la pseudonimizzazione definita come "il trattamento dei dati personali in modo tale che i dati personali non possano più essere attribuiti a un interessato specifico senza l'utilizzo di informazioni aggiuntive".

In concreto, questa operazione sostituisce alcuni identificatori con pseudonimi (o token, letteralmente "simbolo" o "simbolico"), cioè dati realistici, ma non veritieri. I dati originali, vengono conservati in un database separato costituito da una tabella delle corrispondenze tra dati originali e gli pseudonimi utilizzati. Tutto ciò permette di re-identificare le persone fisiche, in quanto il titolare, o il responsabile del trattamento, possiede le "informazioni aggiuntive" per poter risalire all'identità degli interessati: si parla perciò di un processo reversibile.

I dati pseudonimizzati non possono essere attribuiti ad un individuo senza utilizzare le predette "informazioni aggiuntive", che in questo caso sono rappresentate dalla tabella delle corrispondenze tra pseudonimi e dati originali. Importante ricordare che le "informazioni aggiuntive" devono essere conservate in un database separato e adeguatamente protetto. Infatti, in questo modo anche se il set di dati pseudonimizzati venisse compromesso, non sarà comunque possibile risalire ai dati originali.

La pseudonimizzazione riduce il rischio di identificazione diretta degli individui, riducendo la correlazione tra un insieme di dati all'identità originaria di una persona interessata, ma non produce dati anonimi. Quindi, i dati pseudonimizzati sono dati

personali e rientrano di conseguenza nella disciplina del General Data Protection Regulation (GDPR), un regolamento dell'Unione europea in materia di trattamento dei dati personali e di privacy [7].

La pseudonimizzazione è uno strumento fondamentale per proteggere i dati personali, ma perché viene scelta questa tecnica rispetto all'anonimizzazione? Di fatto, il principale vantaggio offerto dalla pseudonimizzazione è proprio quello di poter risalire ai dati originali avendo comunque la possibilità di divulgare i dati pseudonimizzati senza rischio di re-identificazione. Questa possibilità ha molteplici risvolti applicativi.

Si pensi, per esempio, ad una società di trasporti che elabora i dati relativi al chilometraggio, ai viaggi e alla frequenza di guida dei propri conducenti. Il trattamento di questi dati è funzionale all'elaborazione delle richieste di rimborso spese per i chilometri percorsi e per addebitare il costo del sevizio ai clienti. Per queste due finalità è fondamentale identificare i singoli conducenti. Tuttavia, un secondo team all'interno dell'azienda di trasporti utilizza gli stessi dati per ottimizzare l'efficienza della flotta di corrieri. Per questa finalità non occorre identificare i singoli conducenti. Pertanto, l'azienda garantisce che il secondo team possa accedere ai dati solo in una forma che non consenta di identificare i singoli corrieri, sostituendo gli identificatori (nomi, etc.) con un equivalente identificativo ma fittizio. Il secondo team può accedere solo al set di dati pseudonimizzato. Nonostante ciò, l'organizzazione stessa può comunque identificare i conducenti, per adempiere alle finalità sopra descritte, grazie ad una tabella delle corrispondenze tra i dati originali e l'identificativo fittizio. Nella Fig. 3 è possibile osservare l'operazione per risalire ai dati originali a partire dai dati pseudonimizzati.

Figura 3: Esempio di Pseudoaninimizzazione

Nell'esempio in Fig. 3 gli pseudonimi sono rappresentati da alcuni numeri (054 e 062) che sono associati agli identificatori originali. Esistono comunque diversi metodi per generare pseudonimi, tra i quali il più comune è l'utilizzo delle funzioni di hash (letteralmente "sminuzzare"), che calcolano, a partire da un insieme di caratteri di lunghezza arbitraria, una stringa alfanumerica di lunghezza determinata. Per quanto

[7] `https://it.wikipedia.org/wiki/Regolamento_generale_sulla_protezione_dei_dati`

l'hash sia una funzione non invertibile (quindi non si può ritornare ai dati originali applicando una funzione inversa di hash), l'utilizzo di una tabella hash (di fatto una tabella delle corrispondenze) rende questa tecnica un utile metodo a servizio della pseudonimizzazione[8]: infatti, nella tabella in Fig. 3 la stringa alfanumerica risultante viene associata ai dati originali rendendo possibile la re-identificazione degli interessati. Nella sessione successiva si andrà a esporre, in maniera più dettagliata, le diverse tipologie di dato esistenti, si analizzeranno le loro criticità strutturali e si esporranno alcune tecniche di anonimizzazione allo stato dell'arte. In particolare si andranno a descrivere i Multidimensional Data, Transactional Data, Graph Data, Time Series Data e Longitudinal Data, e per ogni tipologia di dato quali tecniche si possono applicare per ottenere un valido trade-off tra *privacy* e *utility* dei dati medesimi.

4.1 Multidimensional Data

I dati multidimensionali sono una tipologia di dati che presentano una struttura che può essere affiancata concettualmente a quella di un database relazionale (Tabella 1).

Name	Zip Code	Gender	Income
John Bill	56000	M	10'000
Frank Pots	56010	M	12'000
Julie Ying	56015	F	8'000
Marcus Brown	56127	M	25'000

Tabella 1: Tabella esemplificativa di dati multimensionali

Ogni tupla (o record) è caratterizzata da EI (name), SD (Gender, Income), QI (ZipCode) ed è indipendente dalle altre; non essendoci correlazione logica tra i record, l'applicazione di una tecnica di anonimizzazione su un record non influenza i restanti. Generalmente, le tecniche di anonimizzazione e preservazione della privacy che vengono applicate su questa tipologia di dato vengono raggruppate in due macro categorie: *Random Perturbation Method* e *Group Anonymization Techniques*. Le criticità che si affrontano durante l'anonimizzazione di dati multidimensionali sono :

- Altà dimensionalità delle tuple: non consente una corretta distinzione tra QI e SD.
- Mancata conoscenza del background di un possibile attaccante.

[8] `https://www.iusinitinere.it/`

4.1.1 Tecniche di anonimizzazione

Esistono diverse tecniche per anonimizzare i dati multidimensionali:

- **Aggregation e k-anonymity**: queste tecniche, tramite l'aggregazione con k tuple diverse, tentano di impedire l'individuazione di un record. Attribuendo il medesimo valore ad un attributo per k tuple risulta più difficile l'individuazione di uno specifico record. Si ottiene quindi una generalizzazione, ovvero una sostituzione di più valori specifici con un valore semanticamente simile.

- **L-Diversity**: la L-Diversity estende la tecnica di tipo k-anonymity andando a identificare e classificare in blocchi le tuple che presentano medesimi QI. In ciascun blocco garantisce che tra gli SD ci siano almeno L diversi valori di L attributi.

- **T-Closeness**:questa tecnica rappresenta una evoluzione della L-Diversity,in quanto l'obiettivo è quello di creare classi equivalenti che siano simili agli attributi iniziali. Utile quando si vuole che i valori ottenuti siano quanto più vicini a quelli di partenza. Questa tecnica impone che non solo devono esistere almeno L valori diversi all'interno di ogni classe di equivalenza, così come indicato dalla tecnica di L-diversità, ma anche che ogni valore è rappresentato tante volte quante sono necessarie per rispecchiare la distribuzione iniziale di ciascun attributo.

4.2 Transactional Data

I Transactional Data presentano una struttura che rappresenta le transazioni di individui nei confronti di determinati prodotti. Questi dati hanno la caratteristica di avere una grande dimensionalità ma sono sparsi, ovvero vi è una grande presenza di valori nulli. La sensibilità del dato è legata alla transazione nella sua interezza e non al singolo prodotto. Le tecniche di anonimizzazione che vengono applicate nei database relazionali, per esempio quelle applicate ai dati multidimensionali, non sono applicabili ai dati transazionali perchè non presentano uno schema preciso a differenza dei precedenti e perchè andrebbero ad azzerarne l'utilità visto che si utilizzano principalmente per effettuare studi di Data Mining individuando pattern o correlazione tra prodotti acquistati. (Tabella 2)

Name	T_1	T_2	T_3	T_4	T_5
John	1	-	-	-	1
Joe	-	1	-	1	1
Maria	1	1	1	-	1

Tabella 2: Tabella esemplificativa di dati transazionali

4.2.1 Tecniche di anonimizzazione

Le tecniche di anonimizzazione per i dati transazionali non sono molte e tutte rappresentano unicamente degli approcci sperimentali. Si ritiene importante citare l'elaborato [2] perchè propone un approccio che risulta particolarmente efficiente. Viene eseguito un approccio preliminare alle tecniche conosciute, effettuando un focus su quelle che caratterizzano l-diversity: una *generalization-based* e l'altra *perturbation-based*. Entrambe le tecniche presentano due criticità: assumono uno schema definito e assumono che il dominio dei QI sia relativamente ristretto. La tecnica generalization based partiziona i dati in gruppi disgiunti di transazioni tali che ciascun gruppo contenga record sufficienti per avere l elementi distinti. Quindi tutti i QI verrebbero generalizzati nell'intera categoria e se almeno due transazioni in un gruppo avessero due valori distinti per la stessa colonna, allora tutte le informazioni su quell'elemento andrebbero perse. Pertanto vista l'alta dimensionalità dei QI, utilizzare una tecnica di generalizzazione andrebbe a limitare eccessivamente le informazioni con una conseguente perdita enorme in utilità. Analogo problema presenta una tecnica perturbation-based. Il paper [2] elabora quindi una nuova tecnica la cui euristica si basa su un metodo che consiste in un raggruppamento dei record su correlazione dei propri SD. Questi poi vengono separati dai QI e per ogni gruppo si vanno ad indicare quanti elementi sensibili sono presenti all'interno cercando di mantenere un valore di privacy p fissato.

4.3 Graph Data

Questa tipologia di dato viene rappresentata attraverso una struttura a grafo $G = (V, E)$ dove V rappresenta un insieme di vertici ed E rappresenta l'insieme di archi. Vengono utilizzati principalmente nei social network (ad esempio la rete di utenti e amici che ciascun utente porta con se all'interno della struttura di Facebook), nell'elettronica, nei trasporti, e nelle telecomunicazioni (Fig. 4). Ciascun vertice porta con se molti dati personali e le criticità di una struttura di questo tipo sono legate alle

informazioni di correlazione che ogni vertice ha con gli altri elementi del grafo. EI e QI vengono messi a rischio, nomi, generi, date di nascita, preferenze e relazioni con altri elementi del network.

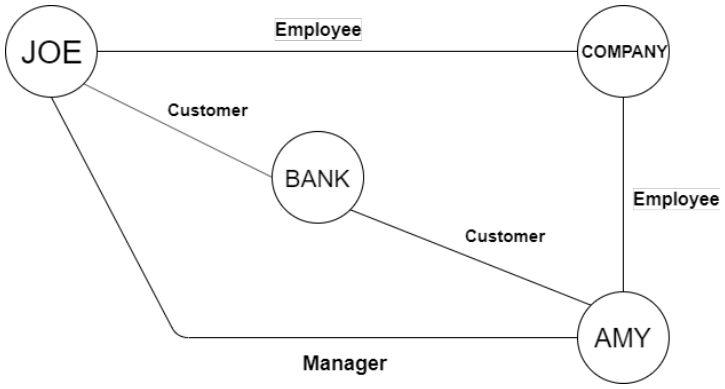


Figura 4: Struttura esemplificativa di Graph Data

4.3.1 Tecniche di anonimizzazione

Per poter effettuare una corretta anonimizzazione di un grafo è necessario prendere in considerazione le proprietà che lo compongono; a differenza dei dati multidimensionali dove ogni tupla può essere anonimizzata indipendentemente dalle altre, nei grafi non possibile avvalersi di tale proprietà. L'anonimizzazione di un singolo nodo compromette i link che esso ha con gli altri nodi e perturba la struttura del grafo. Esistono quindi particolari tecniche di anonimizzazione:

- **Naive Anonymization**: molto efficace quando l'attaccante non presenta conoscenze pregresse. Gli EI vengono sostituiti da valori casuali aumentando l'utilità e diminuendo la privacy.
- **Graph Modification**: si focalizza sul grado di un nodo e di quanto esso sia informativo. Per ogni nodo v ne esistono altri **k-1** nodi con lo stesso grado di v. Questa tecnica di anonimizzazione sacrifica molto l'utilità.
- **Clustering**: tecnica più robusta della naive. Anonimizza il grafo partizionando i nodi e descrive la struttura generale del grafo secondo tali partizioni. Il risultato rappresenta un grafo generalizzato composto da set di super-nodi, uno per partizione, e da super-archi che riportano come informazione la densità degli archi che originariamente connettono le due partizioni.

Spesso a tali tecniche viene affiancata una tecnica di anoninimizzazione su ogni singolo nodo, che spesso è caratterizzato da strutture nella forma di Multidimensional Data (Sezione 4.1)

4.4 Longitudinal Data

I dati longitudinali sono un tipo di dato multidimensionale la cui struttura rappresenta al meglio il raccoglimento di dati ricavati da specifiche misurazioni, effettuate nel corso di un determinato periodo di tempo. I dati longitudinali possiedono le seguenti caratteristiche:

- Rappresentano misurazioni effettuate, spesso, su un unico individuo in un determinato arco di tempo.
- I records sono comunque divisi in EI, SD, QI ma è presente una forte relazione tra l'individuo e gli SD ed esiste una forte correlazione tra i record presenti nel cluster.
- I dati all'interno del cluster presentano un ordine temporale e implicano la presenza di un pattern tra i dati.

ID	Name	DOB	Service Date	Disease	Systolic (mmHg)	Diastolic (mmHg)
1	Bob	1976	12/01/2018	Hypertension	180	95
1	Bob	1976	15/01/2018	Hypertension	160	85
1	Bob	1976	17/01/2018	Hypertension	170	95
1	Bob	1976	21/01/2018	Hypertension	178	98
1	Bob	1976	12/02/2018	Hypertension	160	75

Tabella 3: Tabella esemplificativa di dati logitudinali

E' interessante notare come i dati catturati dalle librerie di user profiling presentino delle analogie con i dati longitudinali, come è possibile vedere in 4

Utente	Timestamp (ms)	Event Key	Value
1	1570205281240	fb_mobile_login_start	1
1	1570205286678	fb_mobile_login_method_not_tried	0
1	1570205293238	fb_mobile_login_method_not_tried	1
1	1570205297207	fb_mobile_login_method_start	1

Tabella 4: Tabella valori catturata in air.com.officemax.magicmirror.ElfYourSelf

I Timestamp di Tabella 4 possono essere riconducibili a le Service Date di Tabella 3 e gli Eventi con i loro valori sono paragonabili ai disturbi e ai valori di pressione del paziente. Nome e DOB del paziente possono essere affiancate al Nome utente e all'Android ID che possono essere recuperate come informazioni generiche del dispositivo in cui sta operando l'applicazione; come spiegato in precedenti capitoli l'Android ID rappresenta un tipo di dato reperibile nell'ambiente mobile ed è un identificativo piuttosto specifico che varia da utente a utente e ne determina la singolarità. La criticità che caratterizza l'anonimizzazione dei dati longitudinali è rappresentata da due principali proprietà :

- Identity Disclosure (Divulgazione di Identità): impedisce un link tra i dati
- Attribute Disclosure (Divulgazione di attributi) : impedisce un collegamento tra dati sensibili (SD)

4.4.1 Tecniche di anonimizzazione

Una delle tecniche che caratterizza l'anonimizzazione di dati longitudinali prende il nome Utility-Guided Anonymization of CLInical Profiles (UGACLIP), un algoritmo che permette la costruzione di un dataset anonimizzato protetto da re-identificazione. L'algoritmo è stato sviluppato in ambito medico e si pone l'obiettivo di soddisfare le policy di privacy andando a soddisfare iterativamente ciascun vincolo che la compone. Inizialmente seleziona il vincolo di privacy associato a più pazienti, dopodichè soddisfa tale vincolo andando a generalizzare iterativamente il QI meno frequente con un altro QI tale da soddisfare il corrispettivo vincolo di utilità, minimizzando la perdita di informazioni. Se il vincolo di privacy non viene soddisfatto il QI viene soppresso. Infine l'algoritmo procede all'iterazione seguente e termina quando la privacy policy viene soddisfatta.

4.5 Time Series Data

I dati time-series rappresentano il risultato di misurazioni effettuate ad intervalli regolari di un qualunque processo/fenomeno. Rispetto ai dati longitudinali (Sezione 4.4) possiedono una dimensionalità più alta ed in costante crescita ma al contrario di essi presentano una minore correlazione tra le misurazioni. Le sfide nell'anonimizzare questa tipologia di dato sono legate alla volontà di mantenere informazioni statistiche

e pattern anche dopo aver effettuato l'anonimizzazione in modo da garantire livelli di utilità adeguati. Tuttavia tali informazioni (media, varianza e covarianza ad esempio) possono rappresentare conoscenze molto importanti per un attaccante. L'elevata dimensionalità delle tabelle porta con se la necessità di molta capacità computazionale nell'esecuzione degli algoritmi di anonimizzazione (Tabella 5).

ID	Company	Address	Week 1	Week 2	Week 3
1	ABC	Palm Springs, 56000	100	200	600
2	CBN	Boulevard Av, 56034	200	400	300
3	CNN	Victoria Plaza, 54372	400	100	700
4	BBC	Queen Street, 57643	200	600	1000

Tabella 5: Tabelle esemplificativa di dati time series

4.5.1 Tecniche di anonimizzazione

Le tecniche di anonimizzazione dei dati time series hanno come punto comune l'obiettivo di mantenere le proprietà caratterizzanti di questa tipologia di dato ovvero:

- Conservazione del pattern.
- Conservazione delle proprietà statistiche come media e varianza.
- Conservazione delle proprietà nel dominio delle frequenze, ovvero la correlazione tra tempo e dominio delle frequenze.

Esistono due tecniche per questa tipologia di dato: la generalizzazione e la modifica del pattern attraverso l'aggiunta di rumore bianco.

La tecnica della **Generalizzazione**, che utilizza una variante della k-anonymity dei dati multidimensionali, viene anche definita *k-p anonymity* [3] e prevede una forte considerazione del pattern dei QI che rappresenta una caratteristica fondamentale dei dati Time-Series. Come descritto in [3] viene preliminarmente effettuata una k-anonymity sui QI, generalizzando un numero definito di *k-group*. Per ogni record in un k-group se esistono almeno *p-1* record con il medesimo pattern, allora si può dire che la *p-anonymity* sia soddisfatta in quel gruppo. Si può partizionare tale gruppo in sottogruppi in cui almeno p record presentino lo stesso pattern. La **White Noise Addiction** invece prevede una perturbazione dei QI attraverso rumore bianco, aggiungendo o rimuovendo valori casuali. Questa tecnica predilige l'utilità, vengono mantenute

le proprietà statistiche dei pattern, ma indebolisce la privacy. E' possibile ricorrere a filtri passa-basso oppure a tecniche di regressione per effettuare una re-identificazione.

5 Stato dell'arte

Analizzando lo stato dell'arte attuale è possibile individuare 3 elaborati che si ritengono importanti e che trattano dell'utilizzo dei dati da parte delle applicazioni e di come queste raccolte dati possano mettere in crisi quella che è la privacy dell'utente finale, non che utilizzatore delle app stesse.

Nel paper proposto da Chen et al., [4] si pone attenzione al rischio di violazione della privacy attraverso i servizi di mobile Analytics e Advertising. Si dimostra nel paper, come un utente malintenzionato sia in grado di estrarre profili e informazioni riguardo l'utilizzo delle applicazioni attraverso le vulnerabilità esposte da librerie di profiling come Google Analytics [5] e Flurry [6]. Tali vulnerabilità, come illustrato nel paper, consentivano l'estrazione di identificatori univoci (device ID) e di ulteriori parametri identificativi (device model, Android ID, ...). Vulnerabilità che sono state poi confermate dalla riuscita ricostruzione di 44 profili che volontariamente si sono prestati all'esperimento. Gli autori hanno dimostrato inoltre come sia possibile effettuare un exploit che possa influenzare le pubblicità proposte dai banner all'interno delle app, manipolando i profili che tali servizi costruiscono collezionando dati. Gli autori fanno notare, in conclusione, come entrambi gli attacchi siano possibili senza che nessuna applicazione maligna sia installata sul dispositivo.

Una differente prospettiva è stata analizzata nel report tecnico proposto dalla University of Washington [7]. All'interno del report viene riportato il risultato di misurazioni effettuate su applicazioni installate all'interno di un campione di device, nel periodo di 3 settimane. Attraverso l'instrumentazione dei dispositivi con tecniche di dynamic taint tracking è stato possibile ascoltare le comunicazioni che tali dispositivi effettuavano verso siti web e app di terze parti con l'obiettivo di registrare eventuali fughe di informazioni profilative. Nel paper viene dimostrato come il 36% dei siti con cui le applicazioni interagiscono raccolgano informazioni profilative e come il 37% di essi tenga traccia di elementi come AndroidID e IMEI, che rappresentano unificatori univoci. Il trasferimento di tali valori avviene senza encryption e spesso ad essi vengono affiancate coordinate di geolocalizzazione. Viene evidenziato nel report come tale raccolta di informazioni rappresenti un rischio per la privacy vista la possibilità di cross-application (ovvero uno scambio di informazioni attraverso le applicazioni presenti all'interno del device e che possano attingere a diverse fonti informative) e cross-site profiling (ovvero la possibilità che diversi siti web possano scambiarsi informazioni e

messaggi e che possano attingere a diverse fonti informative) che un persistent identifier come un ID può dare. Il lavoro si conclude con una euristica soluzione volta a porre maggiore controllo sull'interscambio di informazioni tra applicazioni e siti web per garantire un maggiore controllo sulla privacy dell'utente.
Le problematiche legate alla profilazione e alla divulgazione di dati sensibili non riguarda unicamente i dispositivi mobili ma anche dispositivi IoT e infrastrutture cloud. Il lavoro esposto da Osia [8] affronta tali criticità, esaminando i servizi cloud ai quali vengono inviati moltissimi dati per funzionalità e servizi (ad esempio i programmi di video editing online); dati che potrebbero essere utilizzati per scopi malevoli (ad esempio algoritmi di face recognition per il social advertising). La famiglia di servizi cloud trattati nell'elaborato è rappresentativa di servizi di affitto di software o hardware esterni per la computazione di onerose operazioni nell'ambito del machine learning, che risulterebbero altresì onerose se eseguite unicamente in locale. Gli autori di pongono l'obiettivo di ottenere un trade-off ottimale tra un consumo di risorse di calcolo del device locale e una perdita di privacy per i dati inviati al cloud. Viene strutturata e valutata una architettura ibrida dove il dispositivo locale e il servizio cloud collaborano nell'esecuzione e nel mantenimento di una complessa rete neurale, già inizializzata nel processo di training e testing. Dimostrano come sia possibile inviare unicamente le informazioni rilevanti all'esecuzione del task principale per cui è stato programmato il cloud, preservando al più possibile i vincoli di privacy. L'elaborato termina con la proposta di un framework nel quale l'utente finale non si trovi costretto ne ad effettuare un upload completo di tutti i dati sul cloud (operazione che metterebbe a rischio la sua privacy) ne tentare di nascondere tutte le informazioni attraverso metodi crittografici (computazionalmente onerosi per le risorse locali).

6 Hidedroid

Come già delineato nei capitoli e sezioni precedenti, il progetto prevede la creazione e lo sviluppo di una struttura che si possa interfacciare sia con il sistema operativo che con le applicazioni installate all'interno di esso.
Durante la progettazione di HideDroid, sono state individuate due possibili soluzioni per la sua implementazione:

- **Locale**: prevede l'installazione di un applicativo direttamente all'interno dello smartphone per monitorarne i comportamenti.
- **Cloud**: cattura e analizza unicamente il traffico di rete generato da tali API.

6.1 Struttura Locale

Una implementazione di Hidedroid prevede l'installazione di un applicativo all'interno del dispositivo. Come rappresentato in Fig. 5, all'interno dello smartphone è presente un'applicazione che ha il compito di monitorare ed analizzare le applicazioni presenti all'interno dello smartphone e anonimizzare i dati che possano compromettere la privacy dell'utente.

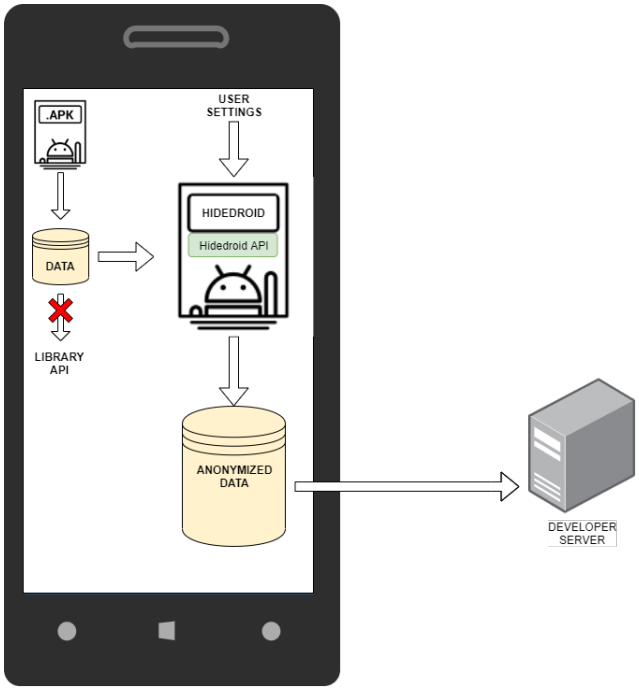


Figura 5: Struttura Locale Hidedroid

Hidedroid si potrebbe concettualmente paragonare ad un proxy che opera tra le applicazioni e i server di raccolta dati. Si pone l'obiettivo di agire come Anonymization Authority provvedendo alle seguenti funzionalità:

- *Data Anonymization Service*: le applicazioni devono inviare i dati raccolti da anonimizzare ad Hidedroid e non utilizzare le proprie API. Hidedroid deve inoltre fornire un resoconto sulle applicazioni analizzate, i criteri di anonimizzazione applicati e se tali criteri possano essere soddisfatti.

- *Detection of Data Exfiltration*: Hidedroid deve essere in grado di identificare quando una applicazione cerca di bypassare il controllo stesso imposto da Hidedroid. L'utente deve essere notificato e devono essere presentate diverse soluzioni.

- *User-Controlled Privacy*: Hidedroid deve permettere all'utente di poter scegliere il livello di anonimizzazione, quanto e cosa dei suoi dati condividere.

- *Retrievable Anonymized Data*: Hidedroid deve rendere possibile l'accesso ai dati anonimizzati.

L'obiettivo è sviluppare un applicativo che possa aggregare e anonimizzare tutte le informazioni, che minacciano la privacy dell'utente, dando all'utente il controllo su questi tipi di dati, indicandone il livello di privacy che intende applicare ad essi prima che possano essere inviati ai server di riferimento (Developer Dashboard). Per poter soddisfare le funzionalità del *Data Anonymization Service* è necessario che il modulo di Hidedroid sappia, in maniera automatica, identificare le librerie profilative più utilizzate ed analizzarne le API. Identificate le API è necessario che Hidedroid intervenga in tale contesto proponendo valide soluzioni di anonimizzazione. Utilizzando algoritmi (discussi in Sezione 4) implementati all'interno del modulo verranno aggregati i dati, considerati potenzialmente pericolosi per la privacy se divulgati non anonimizzati, verranno anonimizzati e inviati al server che risponde alla dashboard dello sviluppatore attraverso un API che presenta una implementazione simile all'API identificata. Per fare un esempio su un API nota e esposta in sezione 3.1.1 e 3.2:

logEvent(key,bundle)

La funzione appartenente alle librerie Firebase (Facebook) verrà elaborata da Hidedroid e sovrascritta come segue:

logEventPrivacy(key,bundle,privacyrules)

Dove il parametro *privacyrules* indicherà il valore di privacy che si intende impostare secondo la scelta dell'utente.

Come descritto precendentemente è stato necessario creare una struttura che potesse, in maniera automatica andare ad analizzare le API presenti all'interno delle applicazioni e di catturarne i parametri. Andando ad identificare le API, i parametri che vengono gestiti, e identificando la tipologia di dato trattato da ciascuna di esse è possibile andare a lavorare su specifici algoritmi di anonimizzazione che siano in grado di operare efficientemente su specifici dati: Multidimensionali, Time Series, Longitudinali etc. (descritti in Sezione 4).

6.2 Struttura Cloud

La seconda possibile soluzione per implementare HideDroid è rappresentata dalla progettazione di un sistema cloud. In questo caso la struttura di Hidedroid viene costruita al di fuori del dispositivo e dell'applicazione. Non si ha più, quindi, la necessità di catturare le API all'interno del dispositivo ma si attende la corretta esecuzione e si opera con un reindirizzamento ai server Hidedroid.

Figura 6: Struttura Cloud Hidedroid

Le criticità che possono presentarsi nella seguente struttura la rendono meno realizzabile quanto ad implementazione. La non operabilità all'interno del dispositivo, rende più complicata la progettazione di questa struttura che presenta maggiori problematiche durante le fasi di raccolta e anonimizzazione dati. Le possibilità di bypassare il controllo di Hidedroid da parte di queste API sono maggiori rispetto alla versione locale e ne diminuiscono le funzionalità. Si è ritenuto necessario focalizzare l'attenzione quindi sull'implementazione di una struttura locale, dimostratasi meno fallace e vulnerabile e che, operando all'interno del dispositivo, offra all'utente una maggiore sensazione di sicurezza e controllo sui propri dati.

6.3 Roadmap e fasi di sviluppo

Al fine di comprendere al meglio le sezioni e capitoli successivi si ritiene opportuno esporre una roadmap di sviluppo di Hidedroid che ne vada a delineare i processi di lavorazione necessari al completamento del progetto:

1. **Raccolta applicazioni**: è necessario scaricare dal Google PlayStore un elevato numero di applicazioni per poter automatizzare e parametrizzare le operazioni che Hidedroid dovrà poi eseguire in maniera autonoma (estrazione delle librerie, analisi API . . .).

2. **Analisi API** : occorre individuare un criterio di analisi delle applicazioni per l'individuazione delle librerie di nostro interesse e delle API che le rappresentano.

3. **Raccolta Dati**: una volta delineate e categorizzate le API di ciascuna libreria, bisogna estrarne i parametri, per scoprire quale tipo di dato trattano e quindi quale algoritmo meglio si adatta all'anonimizzazione.

4. **Sviluppare una GUI**: per poter rendere l'applicativo user-friendly, è necessario progettare una corretta interfaccia utente. Questa deve poter mostrare le funzioni che sono state catturate e i dati che sono stati intercettati e deve poter presentare all'utente diverse possibili soluzioni sulla loro anonimizzazione. Deve avere come scopo principale quello di contestualizzare la gravità delle problematiche affrontate nell'elaborato in maniera semplice, fruibile anche da utenti poco esperti.

5. **Interfaccia Client/Server**: si ritiene utile progettare e sviluppare una interfaccia client/server che permetta un controllo attivo sulla struttura di Hidedroid e che permetta al modulo stesso di poter scambiare informazioni di logging o registrazione errori utile ai gestori della piattaforma per poter mantenere un servizio di alto livello e apportare modifiche utili all'applicativo.

7 Analisi Statica

In questo capitolo si andranno a specificare metodologia e implementazione utilizzate per elaborare e sviluppare i punti 1 , 2 , 3 espressi nella sezione 6.3.

7.1 Metodologia

Come precedentemente introdotto in Sezione 6 e al punto 1 della 6.3 si ritiene necessario elaborare una struttura autonoma in grado di ricevere come unico input un quantitativo generico di applicazioni dalle quali occorre estrarre un elevato numero di informazioni in modo automatico come API, metodi e parametri utilizzati e che rappresenterà un contributo alla futura implementazione e sviluppo di Hidedroid (Fig. 7).

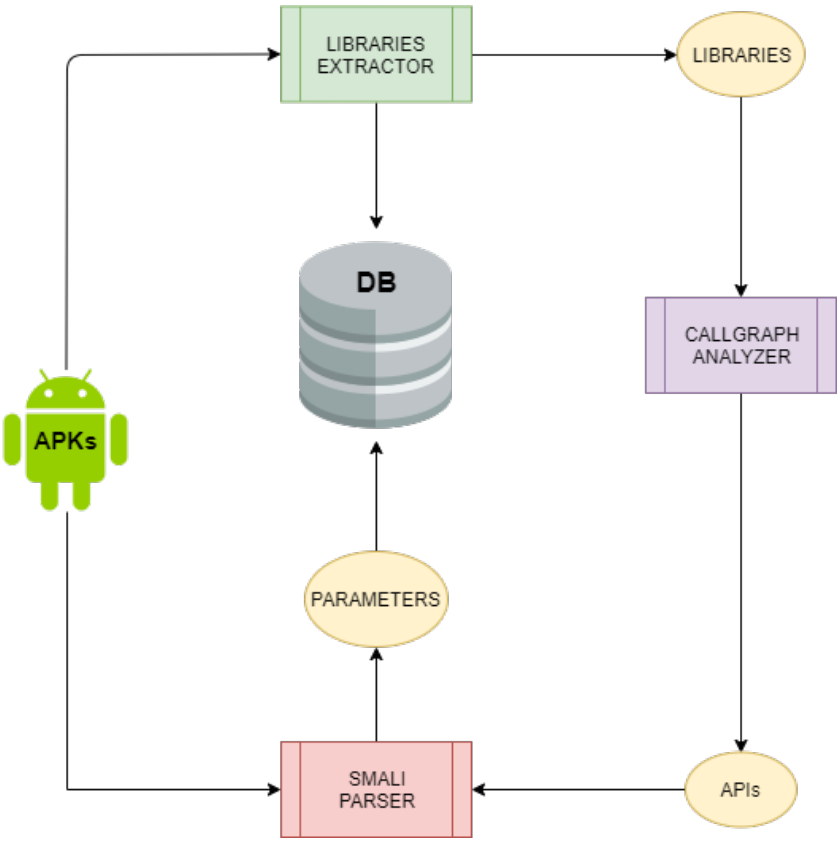


Figura 7: Struttura Generale

Come primo passo è necessario sviluppare un modulo capace di individuare le librerie di analytics presenti all'interno delle applicazioni (**Libraries Extractor**), e catalogarle in base al distributore (es. Google, Facebook, Flurry etc.). Le informazioni ricavate saranno poi inserite in un database.Viene indicato qui l'algoritmo del Libraries Extractor (Algoritmo 1).

Algorithm 1: Extraction Algorithm

Input: APKs Folder

Output: Statistical API collection

Result: Statistical Report

```
1 setup_DB_connection() ;
2 foreach APK in APKFolder do
3     analysisfile ← analyze_APK(apk);
4     analysisfile.find_libraries_informations() ;
5     collection ← analysis.create_report() ;
6     write_report_on_file();
7     collection.saveon_db() ;
8 aggregation_query()
```

Viene inizialmente stabilita una connessione con il database di riferimento (riga 1). Ogni applicazione presente all'interno della cartella viene analizzata e ne viene generato un report (righe 2-5). Per questioni di sicurezza è stato deciso che il report venga salvato anche in locale (riga 6). Si inserisce poi il risultato ottenuto nel database(riga 7). Infine tutti i dati vengono raccolti, ordinati e scritti su un file per i primi risultati (riga 8). L'output di questo modulo è una lista completa delle librerie più utilizzate e che necessitano di maggiore attenzione.
Una libreria che presenta un alto numero di occorrenze rappresenta un dato importante: perchè identifica anche il numero di applicazioni che utilizzano tale libreria e la possibile raggiungibilità di utenza e di raccolta dati che si possono effettuare.

Pertanto di queste librerie, se ne studiano e analizzano, a livello di documentazione, quelle API che si presentano, per struttura e scopo, le più adatte ad effettuare operazioni di raccolta dati e profilazione (Vedi 3.1.1 e 3.2). Identificate tali API si analizza quante e quali chiamate vengano effettuate dalle funzioni esistenti all'interno delle applicazioni e se quest'ultime possano fare riferimento a funzioni di libreria oppure a funzioni personalizzate, create dallo sviluppatore. È compito del modulo **CallGra-**

phAnalyzer individuare il call graph delle API sopracitate.

Algorithm 2: CallGraph Algorithm

Input: APKs Folder

Output: CallGraph

```
1  foreach APK in APKFolder do
2      analysisfile ← Analyze(apk);
3      methodsTofind ← arrayofmethodswewanttosearch();
4      foreach item in methodsTofind do
5          initialize_a_empty_graph();
6          foreach m in analysisfile.searchmethod(item,methodclass) do
7              ancestors ← gettheancestors(graph, method) ;
8              arcestors.add(method) ;
9              foreach node in ancestors do
10                 c ← node.get_class_name();
11                 if c is equal to methodclass then
12                     is_a_library_recursive_call();
13                     ancestors.remove(node)
14         graph ← callgraph.subgraph(ancestors);
```

Nell'algoritmo 2 vengono analizzate le applicazioni all'interno di una cartella (`apk`). Inizializzando i nomi delle API (estratti attraverso uno studio della documentazione) in un array, vengono ricercate all'interno di tutti i metodi estratti.(righe 1-4). Per ciascun metodo trovato viene inizializzato un grafo inizialmente vuoto (riga 5-6). Dopodiché (righe 7-8) vengono esaminate tutte le chiamate effettuate da altre funzioni a tale metodo e ogni chiamata viene aggiunta, come nodo, ad una lista. Se la radice di un nodo rappresenta una funzione di libreria, essa viene eliminata perché non contribuisce a determinare l'utilizzo effettivo del metodo in tale applicazione. Al contrario se la classe del metodo radice appartiene al package name dell'applicazione analizzata esse vengono mantenute come nodi perché implementate dallo sviluppatore (righe 9-13). Con i restanti nodi presenti nella lista viene creato un grafo (riga 14).

Utilizzando il call graph è possibile individuare quelle API che oltre ad eseguire operazioni di raccolta dati presentano numerose occorrenze all'interno degli `apk` di riferimento. Un API chiamata un elevato numero di volte presenta un grafo ampio e il nodo rappresentante l'API ricercata ha un grado elevato. Di queste occorre capire come cat-

turarne i parametri, perchè delineano le tipologie di dato che possono essere raccolte da tali API. Per effettuare tale procedura in maniera statica si ritiene opportuno ricorrere ad una analisi statica del codice SMALI.

Sia per la funzione logEvent di FirebaseAnalytics 3.1.1 sia per la funzione logEvent di FacebookAnalytics 3.2, il pattern di implementazione in codice SMALI è il seguente (Codice Sorgente 3)

```
const-string v2, "Evento"
...
invoke-virtual {v0,v2,v1},Class->Method(ParametersType)returnValue}
```

Listing 3: Codice SMALI chiamata metodo

Si è sviluppato un algoritmo per poter raccogliere non solo ogni singolo parametro passato all'API , ma anche le occorrenze totali di tali parametri in tutte le applicazioni poste sotto analisi.
In questo modulo si ritiene importante la fase di estrazione del parametro *key* all'interno della funzione *logEvent(key,bundle)*(effettuata nelle righe 11-22 di Algoritmo 3).

Algorithm 3: Smali Algorithm

Input: APKs Folder,smalireportFolder,librariesFile

Output: DB Instance

```
1  foreach apk in apkfolder do
2      filename ← apk.getfilename();
3      md5 ← encode.md5(filename) ;
4      colll ← db[md5];
5      occdict ← dictionary(md5);
6      if apk is not analized then
7          smalifiles ← disassembler(apk);
8          dd[md5] ← new;
9          scan(librariesfile);
10         lib ← libraryname;
11         foreach smali in smalifiles do
12             lines ← smali.read();
13             if line.search(lib) then
14                 if line.search(...) then
15                     index ← findmethod();
16                     foreach line in backlines do
17                         if search.("const-string"+param, line) then
18                             vParam ← extractParam();
19                             dd[md5] ← insert(vParam);
20                             addOccToDict(dictionary, vParam.encode());
21                             addOccToDict(ddd, vParam.encode());
22                             index ← i;
23         insert_into_db(dictionary);
24     remove(smalireport);
25     insert_into_db(ddd);
```

L'algoritmo 3, implementato all'interno dello **SMALI Parser**, analizza le medesime applicazioni analizzate nell'algoritmo 1, che vengono catalogate e a cui viene associato uno specifico spazio all'interno di un database(righe 1-4). Viene poi creato un dizionario, quello delle occorrenze, atto a contenere il numero di parametri, appartenenti alle API, collezionati e ordinati per nome (riga 5). L'applicazione viene poi disassemblata (processo che permette la traduzione dei file `.dex` in codice SMALI (Sezione 2.2)) e ne viene attribuito un dizionario atto a contenere le informazioni dei parametri estratti

dall'API (riga 7-8).

Un file contenente le informazioni delle API viene poi scansionato (riga 9-10) alla ricerca, per ogni libreria, della classe e del metodo di profilazione di cui si intendono estrarre i parametri. Viene effettuata una scansione di ogni file SMALI estratto da quello specifico `apk` (riga 11), analizzandolo riga per riga, per trovarne le API estratte dai moduli precedenti (riga 12), fino a che non si incontra una chiamata all'API (riga 13-15). Il metodo *findmethods()* restituisce l'indice della riga in cui è stata trovata la chiamata all'API. Dopo aver individuato l'API all'interno del codice smali, si ha la necessita di individuare il valore del parametro passato come argomento all'API, per fare ciò si vanno ad analizzare le righe precedenti all'invocazione dell'API (righe 16-17). Individuato , tale parametro viene salvato in due posizioni, nel dizionario specifico della applicazione (riga 19-20) e nel dizionario delle occorrenze (riga 21). Infine entrambi i dizionari generati, terminato il programma, vengono inseriti all'interno del database (righe 22-25).
Al termine dell'algoritmo si ottiene un database contenente le informazioni sui parametri, le occorrenze totali e specifiche di ogni applicazione.

Identificare nello specifico i valori di ciascun parametro *key* (appartenti a logEvent(key,bundle), di ciascuna chiamata alla API, ha permesso una possibile categorizzazione della tipologia di dato che poteva trasportare, sia essa stata una stringa , un intero, boolean ...
Il valore del parametro *bundle* associato a tali chiavi non è stato possibile individuarlo staticamente, tuttavia sono state prese in considerazione, unicamente dal lato progettuale, altre strategie per l'individuazione utilizzando tecniche di analisi dinamica. Nella sezione successiva si discuterà delle scelte prese a livello implementativo per la creazione della struttura precedentemente esposta.

7.2 Implementazione

Ciò che verrà esposto in questo capitolo, riguarda l'implementazione dei moduli espressi nella sezione 7.1. A livello implementativo si è deciso di utilizzare il linguaggio Python e l'automazione di comandi bash parametrizzati.
Per salvare e catalogare tutte le informazioni utili alla identificazione di librerie e API dannose per la privacy è stato scelto un database NoSQL, in particolare **MongoDB**. Si è scelto un database Mongo non solo perché vengono fornite moltissime API nel

linguaggio di programmazione scelto ma anche perché un database orientato al documento, NoSQL, che risulta efficiente in situazioni di scalabilità orizzontale. Il concetto di riga viene rimpiazzato dal documento e il concetto di tabella da collezione creando così un modello più flessibile. Non ci sono schemi predefiniti: chiavi e valori di un documento non sono di tipo e dimensione fissa quindi aggiungere o rimuovere informazioni è molto più semplice. Inoltre, le collezioni possono ospitare documenti di diverse dimensioni. Consegue come la possibilità di avere documenti con struttura diversa e dimensioni diverse si adatti molto bene a rappresentare diverse `apk` con diversi parametri e metodi al loro interno.
Il modulo di **Libraries Extractor** è stato realizzato con la creazione di un *Exodus Fork* in grado di estrarre le librerie che sono presenti all'interno delle applicazioni sotto analisi.

Exodus è un tool complesso, realizzato in linguaggio Python, il cui codice risulta opensource perchè appartenente ad una organizzazione francese no profit [9]. E' importante far notare come nell'implementazione dell'algoritmo 1 venga sfruttata la classe messa a disposizione da Exodus:

```
class AnalysisHelper(StaticAnalysis):
    def create_json_report(self):
        return {
            'application': {
                'handle': self.get_package(),
                'version_name': self.get_version(),
                'version_code': self.get_version_code(),
                'uaid': self.get_application_universal_id(),
                'name': self.get_app_name(),
                'permissions': self.get_permissions(),
                'libraries': [l.decode('utf-8')
                for l in self.get_libraries()],
            },
            'apk': {
                'path': self.apk_path,
                'checksum': self.get_sha256(),
            },
            'trackers': [{'name': t.name, 'id': t.id}
```

```
            for t in self.detect_trackers()],
        }
```

Le librerie estratte dall'Exodus Fork rappresentano l'input per il **CallGraph Analyzer** che si occupa dell'individuazione delle API. Il modulo implementa l'algoritmo 2, utilizza Androguard [10] una libreria utilizzata per l'analisi delle applicazioni Android e la libreria Networkx [11] per la gestione dei nodi appartenenti al call graph.
Le API individuate vengono inserite insieme alle APKs all'interno dello **SMALI Parser**. Questo modulo implementa l'algoritmo descritto in 3, disassemblando le applicazioni attraverso apktool [12], estraendo i parametri passati alle API di riferimento e salvandoli all'interno del database Mongo secondo specifici criteri:

- dedicando una collezione a ciascuna applicazione, in cui sono inseriti i parametri specifici di ciascuna API di ogni specifico `apk`
- raccogliendo in un altra collezione le occorrenze di tutti i parametri estratti catalogandoli per libreria di appartenenza

Per motivi prestazionali e di tempi di esecuzione durante la scrittura dell'algoritmo 3 sono state fatte inoltre le seguenti considerazioni:

- In MongoDB e in particolare in un documento BSON non si può superare la dimensione di 16MB, pertanto occorre un controllo che tale soglia non venga superata da nessuna collezione presente al momento dell'inserimento.
- Le collezioni che superano la soglia dimensionale imposta, nel caso delle collezioni contenenti le occorrenze, vengono elaborate e divise in più collezioni con un numero crescente alla destra del medesimo nome di libreria (ex. Facebook_1, Google_9).
- L'md5 delle app analizzate viene inserito all'interno di un file di testo che ne evita la rianalisi in presenza di doppie copie oppure di shutdown improvvisi del sistema con conseguente riavvio.

I risultati di quest'ultimo script saranno poi accuratamente mostrati e discussi nella sezione 8.

8 Risultati

In questa sezione verranno discussi i risultati ottenuti e verranno esposte le considerazioni finali che tali risultati giustificano.
Tutto lo sviluppo dell'elaborato è stato svolto principalmente su un unica macchina messa a disposizione dall'Università degli studi di Genova che presenta le seguenti caratteristiche:

CPU	i7-3770
RAM	16 GB
Sistema Operativo	Ubuntu 18.04.2 LTS
HDD	488,2 GB

Tabella 6: Caratteristiche Macchina

Il linguaggio utilizzato per lo sviluppo e il calcolo dei dati statistici è stato Python 3.5 [13] affiancato ad Android Studio [14]per la creazione di applicativi per il test dei moduli CLI (ovvero provvisti di una Command Line Interface in grado di eseguire comandi specifici da terminale) appositamente creati.

Si vanno ora ad mostrare e discutere i risultati ottenuti durante lo sviluppo del percorso di tesi. Come primo risultato si discute l'output del modulo Exodus Fork, che rappresenta l'individuazione delle maggiori librerie presenti all'interno del campione di riferimento, che si ricorda essere $50,000$ applicazioni provenienti dal Google Play Store (Fig. 8).

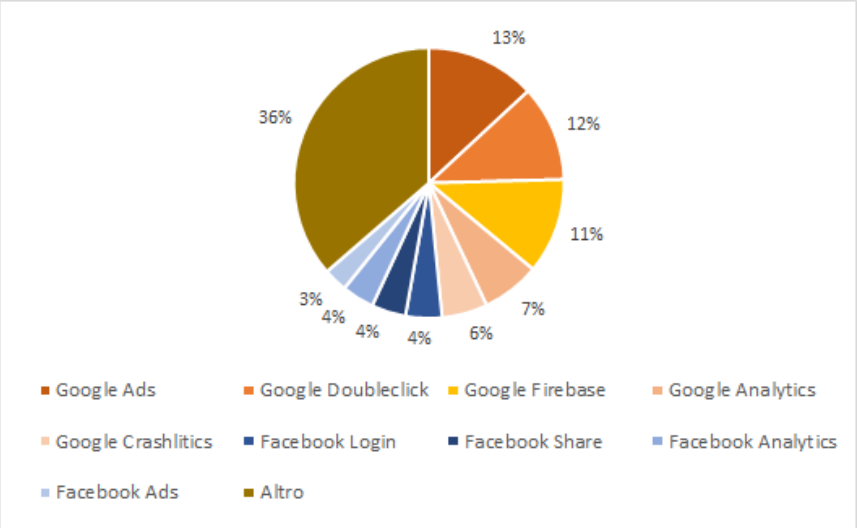


Figura 8: Distribuzione Librerie

Il grafico a torta esposto in Figura 8, fa notare come Google, Firebase e Facebook siano aziende proprietarie e distributrici della maggior parte delle librerie presenti all'interno delle applicazioni (sono presenti nel campione 211143 librerie). La stretta collaborazione tra il team Firebase e il team di Google [9] evidenzia come il 43% (circa 90'000) delle librerie presenti nel campione siano unicamente di proprietà Google. Di seguito compare Facebook che tra librerie di advertising, sharing e di analytics rappresenta il 21% (circa 45'000). Meno rilevanti le librerie presenti nel restante 36% (circa 76'000) che risultano troppo poco utilizzate per poterne esaminare l'impatto che possano avere in ottica di profilazione dell'utenza.
La distribuzione di tali librerie risulta interessante se si decide di andare ad analizzare la loro suddivisione all'interno delle categorie più rappresentative tra le 50'000 applicazioni che nel nostro campionario sono risultate: Education, Games, Books, Tools e Dating. In particolare, per queste categorie, si è analizzato l'impatto delle librerie unicamente rivolte all'analytics e alla profilazione dell'utenza.

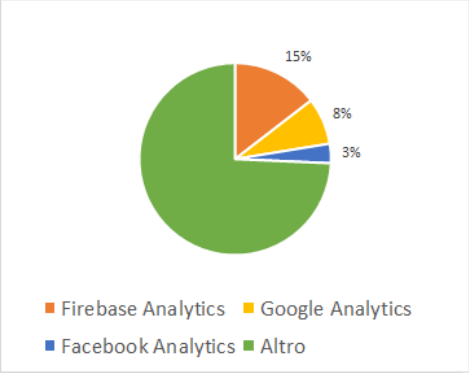


Figura 9: Education Report

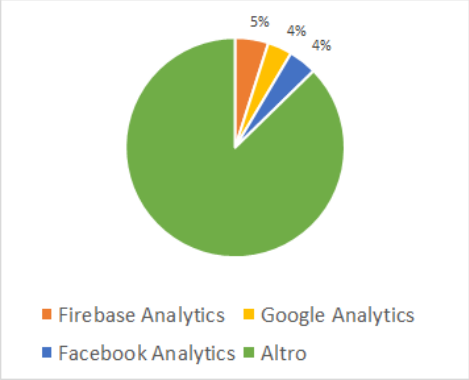


Figura 10: Games Report

Nella Figura 9 si può notare come le librerie Google e Firebase Analytics gestiscano una percentuale maggiore rispetto alla controparte Facebook. Questo indica che nello sviluppo di applicazioni che si dedichino all'intrattenimento, ci sia più interesse nello sfruttare le potenzialità di Firebase e i suoi metodi di profilazione. Tra le applicazioni analizzate ne troviamo alcune dedicate al mondo scolastico e accademico [10], oppure tutorial[11]

In maniera del tutto differente è la rappresentazione per la categoria Games (Figura 10). Qui le librerie di analytics risultano equamente distribuite e sono presenti con meno

[9] https://support.google.com/analytics/answer/7388022?hl=it
[10] https://play.google.com/store/apps/details?id=edu.osu.osumobile&hl=en
[11] https://play.google.com/store/apps/details?id=com.InstrumentationTutorials&hl=it

frequenza. Il risultato è rappresentativo di una categoria di applicazioni che utilizza principalmente librerie legate alla grafica e alla creazione di motori di gioco. L'utilizzo di una libreria di analytics può essere ad esempio giustificato dalla presenza di una funzionalità social (che il gioco stesso potrebbe integrare) o a un sistema di cloud-save per la gestione dei progressi di gioco.

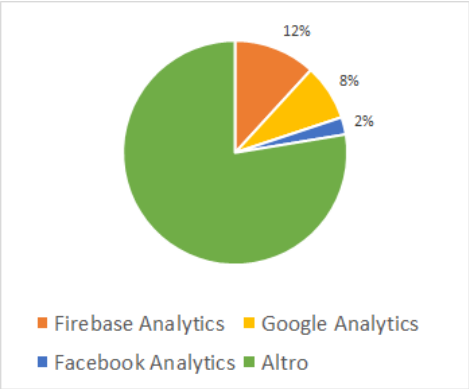


Figura 11: Books Report

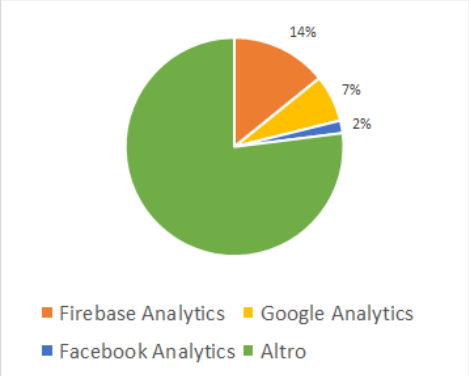


Figura 12: Tools Report

Analizzando la categoria Books e la categoria Tools possiamo notare come per entrambe (Figura 11 e 12) si prediliga l'utilizzo di librerie Google (o Firebase) mentre con meno frequenza si utilizzino librerie di Analytics provenienti da Facebook.Appartengono a queste categorie rispettivamente applicazioni per la lettura di paper o documentazione [12] e applicazioni che coinvolgono l'utilizzo di Bluetooth, GPS, ... per aggiungere funzionalità sul proprio device o interfacciarlo con differenti apparecchiature. [13] Unicamente la categoria Dating, vede un utilizzo delle librerie equamente distribuito tra Firebase Analytics e Facebook Analytics, presentando una percentuale inferiore per la libreria Google. (Fig. 13)

[12] https://play.google.com/store/apps/details?id=christian.dream.interpretation&hl=en_US

[13] https://play.google.com/store/apps/details?id=com.pelix.loopvideorecorder&hl=it

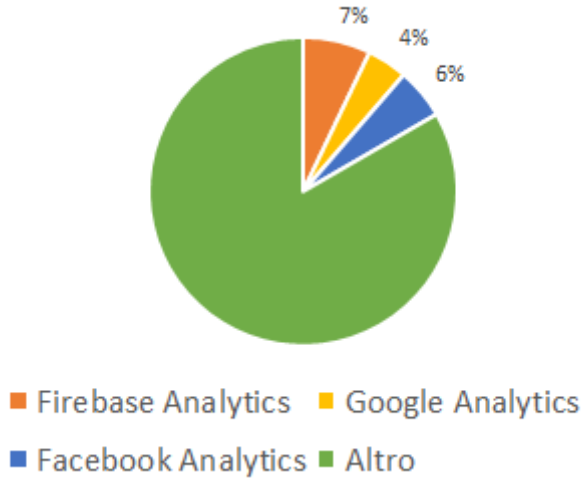


Figura 13: Dating Report

Come esposto nelle precedenti sezioni 3.1.1 e 3.2 occorre andare anche ad analizzare i risultati che evidenziano i parametri *key* estratti dalla funzione *logEvent(key,bundle)*: utilizzata per il tracciamento e la profilazione degli eventi che si decide di registrare all'interno delle applicazioni. Analizzando i file SMALI di oltre $50,000$ applicazioni sono emersi i seguenti risultati (Figura 14 e 15).

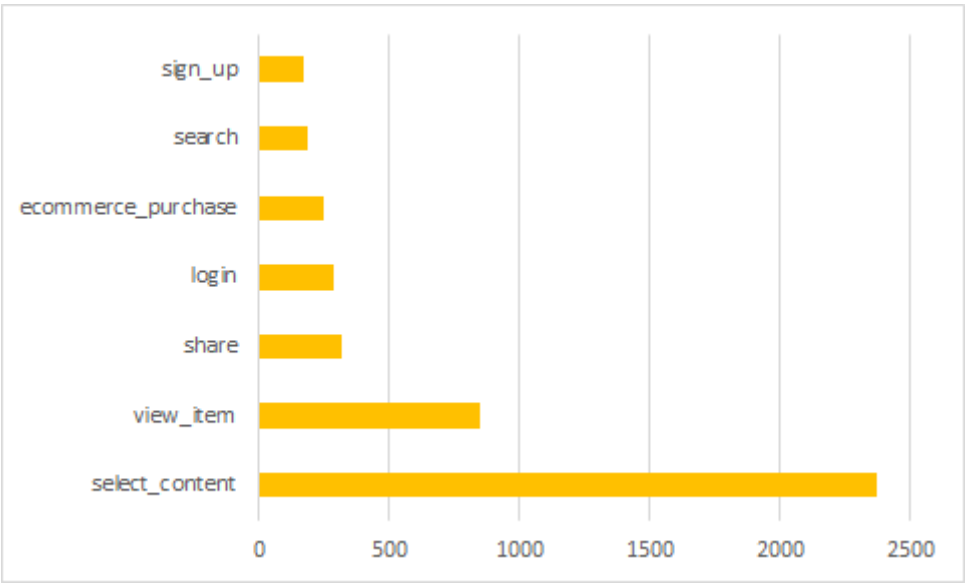


Figura 14: Parametri Firebase Analytics

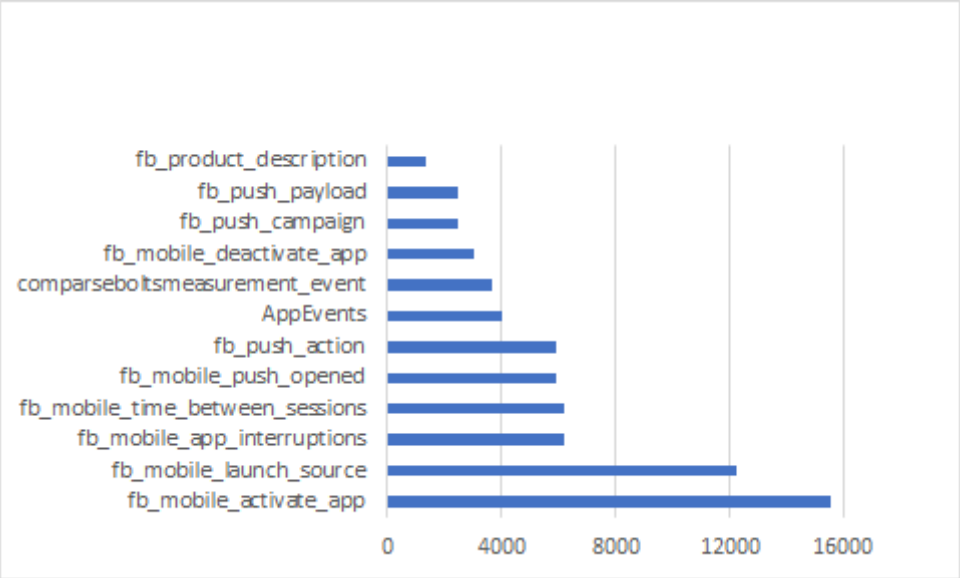


Figura 15: Parametri Facebook Analytics

Il numero di parametri, associati ad eventi (catturati attraverso il modulo di SMALI Parser presentato in sezione 7.1) appartenenti alle librerie di Google risulta inferiore ai parametri catturati per Facebook, nonostante la presenza di librerie Google all'interno delle applicazioni sia maggiore di quella Facebook. Si nota come in questa analisi non siano stati esaminati parametri associati ad eventi personalizzati, creati dallo sviluppatore, non rilevanti in termini di frequenza e poco identificabili e classificabili semanticamente, ma unicamente eventi che la stessa libreria di riferimento mette a disposizione dello sviluppatore. Questo delinea una implementazione meno frequente ma più consistente delle librerie Facebook che quando vengono implementate nelle applicazioni vengono utilizzate per il tracciamento di un elevato numero di eventi e quindi, nella nostra contestualizzazione dell'elaborato rappresentano una maggiore raccolta di dati profilativi che possono mettere in pericolo la privacy dell'utente. Parametri come fb_mobile_launch_source oppure fb_mobile_push_campaign possono rappresentare dati tipologicamente assimilabili a quelli presentati in Tabella 3 se raccolti secondo una regolare frequenza e provenienti dal medesimo utente; ne possono determinare un pattern ed è possibile che la privacy possa essere messa a rischio.

9 Conclusioni e sviluppi futuri

In questo lavoro di tesi è stata proposta una struttura automatica per l'analisi e lo studio di librerie e relative API a scopo profilativo per un numero definito di applicazioni. Analizzando circa $50,000$ applicazioni esistenti sul Google Play Store è stato possibile quantificare a livello statistico quanto la profilazione possa coinvolgere i dati generati dall'utenza inconsapevole. Esaminando in particolare le librerie di analytics appartenenti a aziende come Facebook e Google si è potuto quantificare e identificare il numero di eventi che possono essere collezionati durante l'esecuzione della maggior parte delle applicazioni e di come questi dati possano rappresentare un pericolo per la re-identificazione degli stessi individui che scaricano e utilizzano le applicazioni nei loro smartphone. Sono state inoltre poste le basi per la progettazione di una struttura che possa salvaguardare la privacy dell'utente e che possa garantire pieno controllo sulla gestione dei propri dati. Hidedroid rappresenta una soluzione che non è stata mai presa in considerazione dai documenti che attualmente risiedono allo stato dell'arte e si ritiene che il suo sviluppo rappresenti un importante passo avanti all'interno della data protection and privacy nel mondo mobile.
Gli sviluppi futuri che possono derivare da questo elaborato sono molteplici:

- Sviluppare una metodologia e una struttura implementativa (attraverso ad esempio moduli personalizzati di Xposed [15] che permettono l'instrumentazione del dispositivo per il monitoraggio dell'applicazione) per una analisi dinamica efficiente che possa consentire il raccoglimento di maggiori informazioni derivanti da questi eventi come ad esempio la frequenza di accadimento.

- Focalizzare l'analisi dinamica sulle librerie Analytics di Facebook perchè giudicate influenti come sistemi di profilazione (Sezione 8)

- Programmare e sviluppare una GUI per Hidedroid, che possa essere comprensibile e fruibile anche dall'utenza meno esperta e che garantisca un pieno controllo sui parametri catturati

- Sviluppare e implementare algoritmi di anonimizzazione, basandosi sulle attuali tecniche qui presentate (Sezione 4) e quelle che esistono attualmente allo stato dell'arte, per poter anonimizzare i dati catturati dai moduli Xposed.

Riferimenti bibliografici

[1] Mark Page ; Maria Molina, "The_Mobile_Economy_2013.pdf.crdownload," p. 100, 2013. [Online]. Available: http://www.gsmamobileeconomy.com/GSMAMobileEconomy2013.pdf

[2] G. Ghinita, Y. Tao, and P. Kalnis, "On the Anonymization of Sparse High-Dimensional Data," Tech. Rep.

[3] L. Shou, X. Shang, K. Chen, G. Chen, and C. Zhang, "Supporting Pattern-Preserving Anonymization For Time-Series Data," 2011.

[4] T. Chen, I. Ullah, M. A. Kaafar, and R. Boreli, "Information leakage through mobile analytics services," *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications, HotMobile 2014*, 2014.

[5] GoogleAnalytics, "https://analytics.google.com/." [Online]. Available: https://analytics.google.com/

[6] Flurry, "https://www.flurry.com/." [Online]. Available: https://www.flurry.com/

[7] S. Han, J. Jung, and D. Wetherall, "A Study of Third-Party Tracking by Mobile Apps in the Wild," *Clinical Microbiology and Infection*, vol. 17, pp. S669–S834, 2011.

[8] S. A. Osia, A. S. Shamsabadi, S. Sajadmanesh, A. Taheri, K. Katevas, H. R. Rabiee, N. D. Lane, and H. Haddadi, "A Hybrid Deep Learning Architecture for Privacy-Preserving Mobile Analytics," pp. 1–12, 2017. [Online]. Available: http://arxiv.org/abs/1703.02952

[9] Exodus, "https://exodus-privacy.eu.org/en/." [Online]. Available: https://exodus-privacy.eu.org/en/

[10] Androguard, "https://github.com/androguard/androguard." [Online]. Available: https://github.com/androguard/androguard

[11] Networkx, "https://networkx.github.io/." [Online]. Available: https://networkx.github.io/

[12] ApkTool, "https://ibotpeaches.github.io/apktool/." [Online]. Available: https://ibotpeaches.github.io/Apktool/

[13] Python, "https://www.python.org/." [Online]. Available: https://www.python.org/

[14] A. Studio, "https://developer.android.com/studio." [Online]. Available: https://developer.android.com/studio

[15] Xposed, "https://repo.xposed.info/module/de.robv.android.xposed.installer." [Online]. Available: https://repo.xposed.info/module/de.robv.android.xposed.installer

Ringraziamenti

Dedico quest'ultima pagina ai ringraziamenti alle persone che mi hanno seguito e accompagnato durante tutto il mio periodo universitario e di tesi.

In primis ci tengo a ringraziare il mio relatore, Prof. Alessio Merlo e il correlatore Davide Caputo per l'aiuto e il supporto che hanno mostrato in questi lunghi e tortuosi mesi di tesi (sopratutto gli ultimi).

Ringrazio mia madre, mio padre e mio fratello per essere stati sempre dalla mia parte, per aver dimostrato, compiendo sacrifici, un attaccamento particolare alla costruzione del mio futuro e per aver creduto in me fino a questo punto. E ringrazio anche Mia che nonostante i suoi abbai indesiderati e ciabatte strappate, ha saputo, nell'ultimo anno e mezzo, allentare la tensione universitaria con affetto incondizionato.

Ringrazio Nonno, Zii e Cugini per aver sempre dimostrato profondo rispetto e fiducia nel mio percorso di tesi e nella mia persona e per avermi sempre etichettato come "il più intelligente della famiglia" spronandomi ad andare avanti a testa alta.

Ringrazio i miei più cari e vicini amici, Andrea, Simone, Sara, Cristian e Alessio per essere sempre stati un punto di riferimento o un obiettivo da raggiungere.

Ringrazio i miei amici e colleghi dell'OpenLab, Filippo, Mirko, Filippo, Vittorio, Andrea e Carola perchè in quel laboratorio ho avuto la possibilità di imparare un sacco di cose nuove e interessanti e ho avuto la fortuna di passare ore di risate e divertimento.

Tutto questo lo dedico a voi, famiglia, parenti, amici e ai nonni che purtroppo non ci sono più e che quindi non hanno potuto assistere alla fine di un percorso importante per me ma che avrebbero con tutto il cuore avuto il piacere di vedere...

Grazie ... di cuore ... a tutti