Universitá degli Studi di Salerno

Facoltá di Scienze Matematiche Fisiche e Naturali

Corso di Laurea Magistrale in Informatica

Corso di Sicurezza

Agosto 2011

Analisi Forense per Android

Davide Barbuto	0522500028
Francesco Capano	0522500072
Gaetano Contaldi	0522500029
Andrea Vallati	0522500094

Anno Accademico 2010-2011

Indice

Introduzione

Questo studio si focalizza sull'aspetto forense dei dispositivi mobili con sistema operativo Android.

Sono state effettuate tre diverse tipologie di analisi. In primo luogo é stata effettuata l'analisi logica che consiste in una copia bit a bit del dispositivo di memorizzazione a livello logico, conservando le strutture del filesystem. In questo tipo di analisi, poiché si lavora a livello logico, vengono esclusi i file cancellati.

Il secondo tipo di analisi é l'analisi fisica che consiste in una copia bit a bit dell'intero dispositivo di memorizzazione a basso livello. Questo tipo di acquisizione, ha il vantaggio di consentire l'analisi di file cancellati o parzialmente sovrascritti. Il terzo tipo di analisi utilizzato ha un approccio anti-forensic. L'obiettivo dell'anti-forensic consiste nel compromettere la disponibilitá e l'usabilitá di tracce ed evidenze digitali, al fine di renderle inutilizzabili durante il processo di analisi forense.

Di seguito viene descritta la struttura del documento.

Nel capitolo 1 viene illustrata brevemente la storia di Android, e successivamente la sua architettura e i bug piú recenti riguardanti la sicurezza degli utenti.

Nel capitolo 2 viene fatta una panoramica sulle evidenze digitali presenti nei dispositivi Android e sulle possibili tecniche utilizzabili per ricavare tali evidenze.

Nei capitoli 3, 4 e 5 vengono attuate le tecniche di analisi logica, fisica e anti-forensic per analizzare un dispositivo mobile Android.

Capitolo 1

Android

1.1 La storia

Android è un software stack per dispositivi mobili. Un software stack è un insieme di componenti software capaci di offrire servizi e prodotti completi.

Android include un sistema operativo basato su kernel linux e alcune applicazioni chiave. Android Inc. fu fondata nel 2003 da Andy Rubin, Rich Miner, Nick Sears e Chris White. Nel 2005 fu subito acquisita da Google, che comunque lasciò inalterato gran parte dell'organico dell'azienda. Per molti esperti del settore questa mossa segnò l'entrata di Google nel mercato dei dispositivi mobili.

Google insieme ad altre compagnie hanno formato un consorzio chiamato l'Open Handset Alliance che si occupa dello sviluppo di Android. Ne fanno parte diverse compagnie, come: Google, Htc, Intel, Lg, Nvidia, Motorola, Qualcomm, Samsung Electronics, ecc.

Oltre ai servizi già offerti dal sistema operativo, Android utilizza le applicazioni per estendere le sue funzionalità. Esiste una grande comunità di sviluppatori, che mirano alla creazione di applicazioni valide.

Le applicazioni di terze parti sono scaricabili dal market. Il market è un'applicazione preinstallata su tutti i dispositivi Android. Attualmente ci sono più di 150.000 applicazioni disponibili sul market di Android. Gli sviluppatori possono scrivere le applicazioni in linguaggio Java e utilizzare le librerie offerte dal sistema operativo. [12]

Molto del codice sorgente di Android è stato rilasciato con licenze Open Source e Apache. A partire dal 21 ottobre 2008 Google ha pubblicato tutto il codice sorgente, fanno eccezione solo alcune parti che sono pubblicati sotto la licenza Apache.

Anche se il software è open-souce, i dispositivi mobili non possono utilizzare Android senza il consenso di Google. In più senza questa certificazione non possono ricevere le applicazioni che di solito sono installate di default sui dispositivi Android, come il market. È utile sottolineare che queste applicazioni non sono open-source.

Mostriamo ora l'evoluzione di Android attraverso il rilascio delle varie versioni.

- v 1.0, nome in codice $Apple - pie$, rilasciata il 23 settembre 2008;
- v 1.1, nome in codice *Bananabread*, rilasciata il 9 febbraio 2009. Vennero risolti alcuni bug;
- v 1.5, nome in codice *Cupcake* (basata sul kernel linux 2.6.27), rilasciata il 30 aprile 2009. Vennero aggiunte molte funzionalità nuove e l'interfaccia utente venne modificata;
- v 1.6, nome in codice *Donut* (basata sul kernel linux 2.6.29), rilasciata il 15 settembre 2009. Furono inserite nuove funzionalità come il navigatore Turn-by-Turn, la Voice-Search e il Text-to-speech, in più i dispositivi basati su Android 1.6 supportavano la risuoluzione WVGA;

- v 2.0 e 2.1, nome in codice *Eclair* (basata sul kernel linux 2.6.29), furono rilasciate rispettivamente il 26 ottobre 2009 e il 12 gennaio 2010. Fu rinnovata l'interfaccia utente e fu introdotta l'HTML5 e il supporto per Exchange ActiveSync 2.5

- v 2.2, nome in codice *Froyo* (basata sul kernel linux 2.6.32), fu rilasciata il 20 maggio 2010. Furono migliorate le prestazioni di sistema, fu aggiunta la funzionalità di Wi-Fi hotspot tethering e fu introdotto il supporto per Adobe flash;

- v 2.3, nome in codice *Gingerbread* (basata sul kernel linux 2.6.35), fu rilasciata il 6 dicembre 2010. Fu ridefinita l'interfaccia utente, migliorata la tastiera virtuale e le sue features, in più aggiunto il supporto per la NFC (Near Field Communication);

- v 3.0, nome in codice *Honeycomb* (basata sul kernel linux 2.6.36), fu rilasciata il 22 febbraio 2011. Questa versione è stata creata solo per dispositivi Tablet.

- Attualmente si aspetta per la metà del 2011 una nuova versione di Android con nome in codice: *IceCreamSandwich*.

Figura 1.1: I loghi associati a ciascuna versione di Android.
Da sinistra: Apple e Bananabread, Cupcake, Donut, Eclair, Froyo, Gingerbread, Honeycomb.

Le principali caratteristiche di Android sono:

- Layout: supporta diverse risoluzioni, grafica 3D e utilizza librerie OpenGL ES 2.0;
- Memorizzazione: utilizza una versione lite di SQL;
- Connettività: supporta differenti tecnologie come: GSM, EDGE, CDMA, UMTS, bluetooth, Wi-Fi, LTE e WiMAX;
- Media: supporta diversi formati audio-video, tra cui: JPEG, PNG, GIF, MP3, MIDI, MPEG-4, AAC e 3GP;
- Hardware ausiliari: supporta touchscreen, camera, GPS, accelerometro, giroscopio, acceleratore grafico 3D e sensore di pressione, luce e prossimità.

1.2 Architettura

1.2.1 Analisi Strutturale

É possibile suddividere il sistema operativo Android in cinque principali strati.

1. Kernel Linux: alla base del sistema operativo vi è il kernel Linux. Android fa uso di Linux v2.6 per i servizi di sistema più importanti, come la sicurezza, la gestione della memoria e dei processi, la gestione network, ed il driver model. Il kernel inoltre agisce come livello di astrazione tra l'hardware e lo strato software.
2. Runtime Android: Android include un insieme di librerie base che forniscono la maggior parte delle funzionalità disponibili nelle librerie standard del linguaggio di programmazione Java. Ogni applicazione Android viene eseguita come processo indipendente, con la propria istanza della macchina virtuale Dalvik. La macchina virtuale Dalvik è stata scritta in modo

da permetterne esecuzioni concorrenti in modo efficiente. La Dalvik VM esegue file nel formato Dalvik Executable (.dex), che è ottimizzato per minimizzare l'uso di memoria. La Dalvik VM è basata su registri, ed esegue classi compilate da compilatori Java, successivamente trasformate in formato .dex. La Dalvik VM si affida al kernel Linux per funzionalità di basso livello come la gestione della concorrenza e gestione della memoria.

3. Librerie: Android include diverse librerie di C/C++, utilizzate da diverse componenti del sistema operativo. Queste funzionalità sono esposte agli sviluppatori attraverso l'application framework. Alcune di queste librerie sono le seguenti:

    - System C Library: Un'implementazione derivata da BSD della libreria di sistema standard C (libc), ottimizzata per sistemi embedded Linux.
    - Media Libraries: Queste librerie permettono la riproduzione e registrazione di molti formati audio ,video ed immagini statiche. (p.e. MPEG4, H.264. MP3. AAC. AMR. JPG, PNG).
    - Surface Manager: Gestisce l'accesso al sottosistema display, e compone in modo continuo i livelli grafici 2D e 3D da applicazioni multiple.
    - LibWebCore: Un moderno motore web browser, che alimenta il browser Android.
    - SGL: Il motore grafico 2D di base.
    - 3D Libraries: Un'implementazione basata sulle API OpenGL ES 1.0; queste librerie sfruttano, dove possibile, l'accelerazione hardware 3D, altrimenti fanno uso del rasterizzatore software 3D ottimizzato.
    - FreeType: Rendering delle font, sia vettoriale che bitmap.
    - SQLite: Un motore potente e leggero per database relazionali, disponibile a tutte le applicazioni.

4. Application Framework: Fornendo una piattaforma di sviluppo aperta, Android offre agli sviluppatori la possibilità di costruire applicazioni estremamente complesse ed innovative. Gli sviluppatori sono liberi di sfruttare al meglio l'hardware dei dispositivi, accedere ad informazioni riguardanti la posizione, eseguire servizi in background, impostare sveglie, aggiungere notifiche alla barra di stato, ed interagire con altre componenti del sistema operativo. Tutto ciò è possibile in quanto gli sviluppatori hanno pieno accesso alle stesse framework API usate dalle applicazioni di sistema. L'architettura delle applicazioni è stata progettata per semplificare il riuso delle componenti; ogni applicazione può pubblicare le proprie funzionalità, e qualsiasi altra applicazione può farne uso, purchè non vengano violate le regole di sicurezza imposte dal framework. Alla base di ogni applicazione vi è un insieme di servizi e sistemi, tra i quali:

    - Una ricca collezione di Views, cioè viste, che possono essere usate per costruire un'applicazione. Ad esempio liste, griglie, caselle di testo, bottoni.
    - Un fornitore dei servizi, che permette alle applicazioni di accedere a dati di altre applicazioni (ad esempio la rubrica), o di condividere le proprie informazioni con il sistema.
    - Un gestore delle risorse, che fornisce l'accesso per risorse non in codice, ad esempio stringe specifiche al linguaggio in uso, file di grafica e layout.
    - Il gestore delle notifiche permette alle applicazioni di mostrare notifiche personalizzate nella barra di stato.
    - Il gestore delle attività gestisce il ciclo di vita delle applicazioni e fornisce uno stack di navigazione comune.

5. Applications: Le applicazioni sviluppate dai programmatori risiedono in questo strato del sistema operativo. Android di default è fornito di una serie di applicazioni come il client email, il calendario, il browser ed altre. Tutte le applicazioni sono scritte utilizzando il linguaggio di programmazione Java.

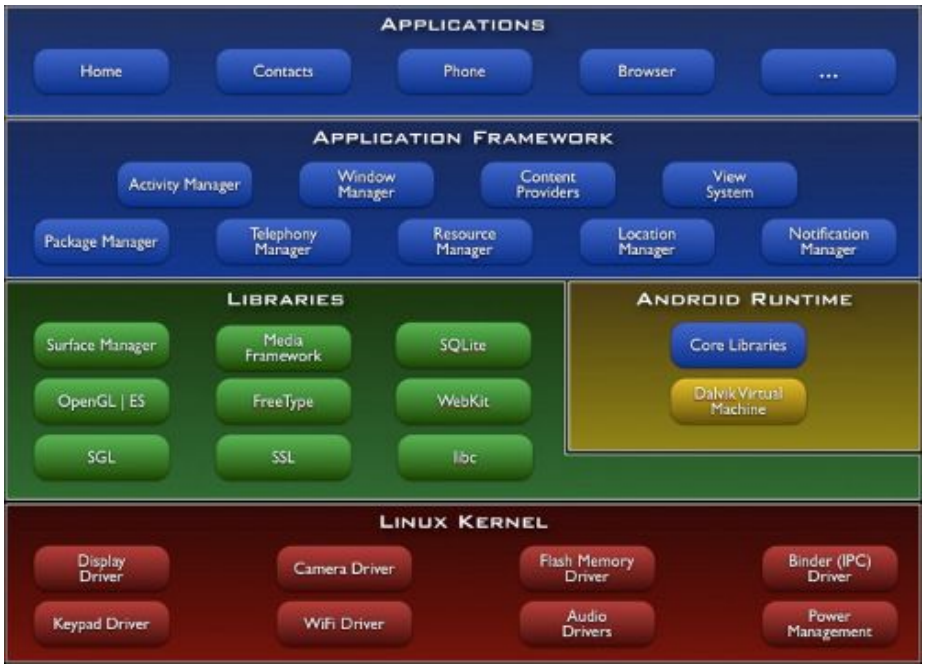

Figura 1.2: Architettura di Android (fonte: android.devapp.it)

Android supporta diversi filesystem, tra i quali yaffs, yaffs2, ext2, ext3, RFS(filesystem proprietario Samsung). Per l'analisi forense, nel caso studio, si è deciso di analizzare versioni di Android basate su yaffs2. È stata presa tale decisione poiché yaffs2 risulta essere il filesystem piú diffuso sui dispositivi android e perchè su ext2 e ext3 esistono numerose documentazioni a riguardo.

1.2.2 Il file system YAFFS2

Yaffs(Yet Another Flash File System) è il file system piú diffuso ed utilizzato all'interno dei sistemi operativi Android. Yaffs è stato sviluppato usando come punto di partenza due dei file system Linux che meglio supportano le memorie flash NOR, JFFS e il suo derivato JFFS2. Yaffs utilizza una memoria flash NAND, per diverse ragioni:

- Utilizzare diverse opzioni di formattazione, come ad esempio i marker per i blocchi di dati corrotti, determinati appositamente dalle infrastrutture NAND, e non modificabili
- Possibilitá di riutilizzare il codice
- Possibilitá di applicare la filosofia "If it ain't broke, don't fix!"

I dati e le informazioni vengono memorizzati all'interno del file system Yaffs in blocchi di taglia fissa, solitamente pari alla dimensione di una pagina (512 bytes). Ogni pagina viene marcata con delle etichette: un file id e il numero del blocco associato. Il numero del blocco inizialmente viene calcolato dividendo la posizione del file per la taglia del blocco. Queste etichette vengono memorizzate nella regione "spare data".
Quando i dati all'interno di un file devono essere sovrascritti, i blocchi interessati vengono sostituiti con un nuovo blocco, che contiene le pagine con i nuovi dati e le etichette delle vecchie pagine: i dati sovrascritti vengono marcati come "discarded"(scartati).
I file "headers" sono memorizzati in una singola pagina, per differenziarli dai file che contengono le effettive informazioni. Solitamente le pagine hanno un marker ulteriore, un numero di 2 bit che viene incrementato ogni volta che la taglia della pagina aumenta. Questa operazione viene effettuata perchè nel caso in cui accada un evento catastrofico prima che la pagina rimpiazzata venga marcata

come “discarded” durante la fase di sovrascrittura, sul filesystem sono presenti due pagine con gli stessi tag: il numero seriale è utilizzato per distinguere quale pagina dovrá essere scartata.
Un blocco del filesystem contenente solo pagine marcate come “discarded” prende il nome di “dirty block” e sará il maggior candidato per la garbage collection. Ogni pagina viene allocata in modo sequenziale a partire dal blocco correntemente selezionato. Quando tutte la pagine all’interno di uno specifico blocco sono state riempite, viene selezionato un nuovo blocco vergine per le successive allocazioni. Se non ci sono blocchi disponibili, allora si provvede a liberare un “dirty block” in modo da renderlo disponibile per nuove allocazioni. Nel caso pessimo in cui non c’é nessun “dirty block” a disposizione, si provvede a rendere disponibile il blocco contenente il maggior numero di pagine discared per la garbage collection.

1.2.3 Suddivisione della memoria

La memoria interna del dispositivo utilizzato è divisa in 6 partizioni, seguendo il partizionamento standard della memoria interna dei sistemi Android.
Ognuna di queste partizioni ha un ruolo distinto per il corretto funzionamento del device.

- /boot
- /system
- /recovery
- /data
- /cache
- /misc

In piú ci sono altre due partizioni per la memoria esterna (SD card):

- /sdcard
- /sd-ext

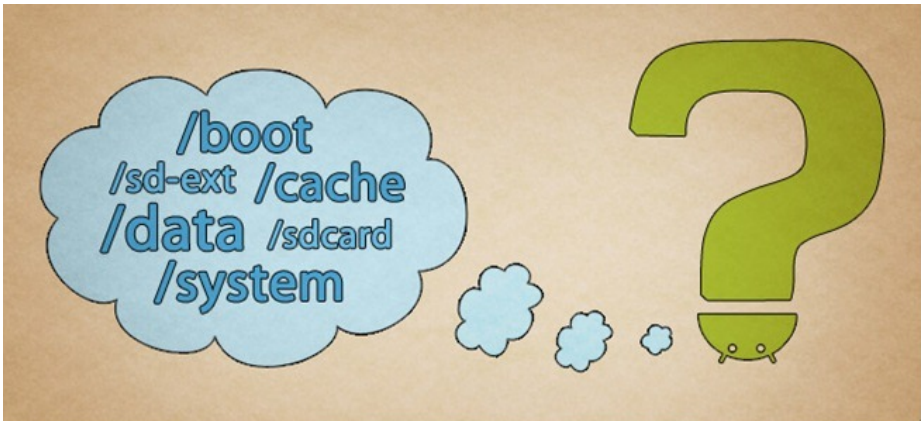

Figura 1.3: Partizioni di Android

Di seguito vengono analizzate tutte le partizioni menzionate precedentemente.

/boot

Questa è la partizione che si occupa del boot del dispositivo, come si può intuire dal nome della stessa. Include il bootloader e il kernel del sistema operativo. Cancellare il contenuto di questa partizione dalla modalitá recovery potrebbe essere fatto, ma il dispositivo non deve essere assolutamente riavviato prima che sia reinstallata una nuova rom, inclusa la boot partition.

/system

Questa partizione contiene l'intero sistema operativo eccetto il kernel. Quindi troveremo anche il modulo che si occupa dell'interfaccia utente, cosí come le applicazioni native di Android. Svuotare questa partizione significa rimuovere Android dal device, ma quest'ultimo è ancora capace di fare il boot.

/recovery

La partizione di recovery può essere considerata come una partizione alternativa per il boot che lancia una console di recovery per operazioni di manutenzione e ripristino avanzato.

/data

La partizione data o userdata contiene, appunto, i dati dell'utente. In particolare contatti della rubrica, messaggi, le impostazioni e le applicazioni installate.

/cache

Questa è la partizione che Android utilizza per memorizzare le informazioni utilizzate con piú frequenza. Svuotare questa partizione non ha effetto sui dati personali dell'utente.

/misc

Contiene le diverse impostazioni di sistema salvate in form booleani, quindi come switch on/off. Questa partizione è fondamentale per il corretto funzionamento del dispositivo, nel caso in cui mancasse o fosse corrotta diversi sevizi potrebbero essere negati.

/sdcard

Non è una partizione della memoria interna del dispositivo. È lo spazio a cui l'utente può accedere liberamente e salvare i propri dati, come documenti, media-file, ecc. Da notare che alcune applicazioni salvano i dati e i settings sulla SD card, di conseguenza anche questa partizione è utile a fini forensi.

/sd-ext

Questa non è una partizione standard, ma è divenuta popolare poiché viene creata da molte ROM non ufficiali. È una partizione aggiunta sulla scheda SD e viene utilizzata come la partizione /data, /cache e /dalvik-cache dalle ROM che hanno i servizi APP2SD+ o data2ext attivi. É utile per i dispositivi che hanno scarsa memoria interna, permettendo all'utente di superare la capacitá fisica di quest'ultima. Quindi l'utente può installare piú applicazioni di quelle che la memoria interna permette di fare. Svuotare questa partizione, essenzialmente, è come svuotare la data partition.

1.2.4 Applicazioni, Processi e Thread

Ogni applicazione è composta da diversi costrutti, ognuno dei quali è un punto di accesso attraverso il quale il sistema può interagire con l'applicazione. Ci sono quattro diversi tipi di componenti:

1. Activities: Una activity rappresenta una singola schermata comprensiva di interfaccia utente.

2. Services: Un servizio è una componente che rimane in esecuzione in background per eseguire operazioni lunghe.

3. Content Providers: Un content provider gestisce un insieme di dati di applicazioni. Attraverso un content provider le applicazioni possono accedere e modificare dati tra di loro.

4. Broadcast Receiver: Una componente che riceve e gestisce messaggi broadcast all interno del sistema. Ad esempio un messaggio broadcast annuncia che lo schermo sta per essere spento, e le applicazioni interessate possono, attraverso un broadcast receiver, ricevere il messaggio e effettuare eventuali operazioni.

Per ogni applicazione l'architettura del sistema operativo Android prevede cinque diversi tipi di processi, che permettono di controllare il comportamento del sistema e dei programmi in esecuzione. Questi hanno diversi ordini di importanza, che sono strettamente subordinati. La gerarchia di importanza dei processi è la seguente:

- Foreground: Un processo che sta eseguendo una Activity, un servizio in fase di avvio o termine, un BroadcastReceiver attivo.
- Visible: Un processo che detiene una Activity visibile ma in pausa, o un servizio legato ad una Activity senza componenti visibili.
- Service: Un processo che esegue un altro processo gia in esecuzione.
- Background: Una Activity non è più visibile, mantenuta in memoria da un processo in background.
- Empty: I processi che non contengono componenti di applicazioni attive, mantenuti in memoria con in solo scopo di ottimizzare la cache.

1.3 Sicurezza su Android

Con gli smartphone che assomigliano sempre piú a dei computer portatili, si delinea allo stesso passo la possibilitá di trovarsi davanti a dei "mobile malware". La sicurezza di tali dispositivi risulta quindi essere un aspetto molto importante. Android, essendo basato su Linux, è un sistema operativo basato sul concetto dei permessi, in cui ogni applicazione installata viene eseguita con un diverso identificatore di sistema (Linux User ID e Group ID). Inoltre, ogni singola parte che compone il sistema operativo viene suddivisa in identitá ben disitinte tra di loro. Ultimo, ma non meno importante, Android isola le applicazioni le une dalle altre e dallo stesso sistema operativo.

Punto centrale dell'architettura di sicurezza di Android è che nessuna applicazione, di default, può ottenere i permessi per portare a termine operazioni che potrebbero avere impatto su altre applicazioni installate o sul sistema operativo stesso. Esempi classici possono essere la lettura/scrittura di dati sensibili o l'accesso alla rete.
L'isolamento delle applicazioni é gestito dal kernel, ed é reso disponibile dall'utilizzo delle sandbox. Ogni app Android viene eseguita all'interno della propria sandbox, e non può in alcun modo interagire con altre applicazioni in esecuzione su altre sandbox.
Nel caso in cui fosse necessario permettere la condivisione di dati tra due applicazioni in esecuzione su due sandbox diverse, si dovranno dichiarare dei permessi specifici per portare a termine operazioni aggiuntive. Ogni app dichiara staticamente i permessi di cui necessita e il sistema operativo chiede all'utente il consenso per l'esecuzione di una operazione specifica.

Purtroppo, come quasi sempre accade, nonostante si prendano tutte le precauzioni possibili per rendere un sistema totalmente sicuro contro eventuali attacchi, anche nel caso di Android sono state scoperte, man mano, diverse falle nella sicurezza. Due sono state le piú eclatanti:

> il bug affliggerebbe il 99% dei device Android; la falla di sicurezza è causata da una cattiva gestione del protocollo di ClientLogin, falla che è stata sistemata solamente nell'ultima versione del sistema operativo Android Gingerbread 2.3.4; ciò significa che chiunque in questo momento non abbia installato sul proprio device questa versione del sistema Android, è a rischio. Nello specifico, il problema risiede nel fatto che ogni device Android, memorizza un token conosciuto anche come authToken, dopo che si effettua un autenticazione ad un servizio online (ad esempio Facebook o Twitter). Questo token viene tenuto in memoria per 14 giorni, permettendo all'utente di riaccedere ai servizi in maniera piú semplice e

> rapida. Tuttavia questo authToken, può essere facilmente "rubato", utilizzando semplici tecniche di sniffing. Un malintenzionato ad esempio può sfruttare l'accesso ad una rete WiFi pubblica non protetta, per rubare e leggere il vostro authToken, ottenendo dunque l'accesso ai vostri dati di login.[6]

Il problema è che quando si effettua l'autenticazione con user e password, i dati trasmessi arrivano da link che non sono di tipo Https (protocollo standard di protezione), e quindi un pirata potrebbe riuscire ad impossessarsene.

> Insicurity, un team specializzato in computer security, ha sviluppato un rootkit capace di inserirsi all'interno di uno smartphone Android. Il programma, una volta installato sul telefono, si attiva attraverso una chiamata telefonica o un sms. Si aggira quindi in modo furtivo e abile, nascondendo le sue tracce al sistema operativo. In realtá questi tipi di "virus" erano in giro giá su Windows e Unix da anni, ma recentemente questi ricercatori di sicurezza hanno avuto modo di sperimentarli su piattaforme mobili. Dato che il rootkit viene eseguito come modulo nel kernel di Linux, di cui Android si avvale, essa ha il piú alto livello di accesso al telefono stesso e può essere uno strumento molto potente per i malintenzionati. Per esempio, potrebbe essere utilizzato per fare chiamate a raffica a caso, oppure potrebbe monitorare la posizione della vittima o anche reindirizzare il browser ad un sito Web dannoso. C'é da rassicurarsi sul fatto che il malintenzionato avrá vita difficile, nel cercare di inserire un malware sul dispositivo. Infatti dovrebbe prima capire come installare il software sul telefono della vittima. Quindi gli unici modi sono quello di passare dall'Android Market, oppure quello di sfruttare un nuovo bug senza patch nel Kernel Linux di Android. Inoltre Google rende difficile ottenere l'accesso al root del kernel (da remoto), garantendo quindi la sicurezza del kernel stesso.[5]

Capitolo 2

Android Forensics

2.1 Evidenze Digitali

Una evidenza digitale consiste in una qualsiasi informazione, con valore probatorio, che sia memorizzata o trasmessa in formato digitale. [1]

Tali informazioni possono essere utilizzate come prova nell'ambito di un processo. È compito della corte determinare se l'evidenza digitale risulta rilevante ai fini dell'indagine. In generale, può essere utilizzato come evidenza digitale, un qualsiasi file compromettente, la cronologia di un browser web, i file multimediali, messaggi di testo, ecc.

Durante l'utilizzo di un qualsiasi dispositivo elettronico, è possibile che questo memorizzi, oltre alle informazioni permanenti (contatti,messaggi, ecc..), le operazioni effettuate. Tali evidenze permettono agli investigatori di ricostruire le azioni effettuate in modo da individuare elementi di prova che concorrano a dimostrare o confutare dei fatti.

Il sistema che andremo a studiare, per l'analisi delle evidenze digitali, è Android. Le evidenze digitali rilevabili sui sistemi Android riguardano:

- contatti
  - numeri di telefono, indirizzi, immagini relative al contatto;
- history del browser;
- i file multimediali;
- registro chiamate
  - ricevute ed effettuate;
- messaggi di testo;
- messaggi multimediali;
- agenda
  - appuntamenti, date da ricordare.

Tramite delle tecniche anti-forensics sarebbe possibile:

- distruggere le evidenze;
- nascondere le evidenze;
- non permettere la generazione delle evidenze;
- confutare le evidenze.

Ognuna di queste tecniche ha l'obiettivo di raggirare l'analisi forense e quindi di rendere l'operazione di investigazione inutile.

L'investigazione è impossibile se attraverso dei sistemi software si effettua la distruzione delle evidenze. Il software in grado di eliminare le evidenze potrebbe, tuttavia, lasciare delle proprie tracce, e quindi dare la possibilità all'investigatore di sostenere che alcune informazioni, probabilmente compromettenti, sono state eliminate oppure vi è stato un tentativo di nasconderle.

È possibile, inoltre, non permettere la generazione delle evidenze in modo da non lasciare traccia durante determinate operazioni; invece di eliminare le evidenze, con questa tecnica si fa in modo che le evidenze non vengano create.

Il confutamento delle evidenze è una tecnica utilizzata per creare dei "falsi alibi", costruendo delle false evidenze digitali oppure modificando le evidenze preesistenti.

Un processo che permette di ottenere tali risultati consiste nel EEP (Evidence Export Process), il quale fornisce la possibilità di esportare i dati che riguardano l'esecuzione di alcune operazioni.

Figura 2.1: Export Evidence Export

Tali informazioni vengono memorizzate all'interno di una cartella privata, alla quale vi è possibile accedere solo tramite determinati permessi. La cartella contiente informazioni riguardanti operazioni effettuate dal dispositivo ed è formata da diversi file. Uno di questi è un file XML (chiamato export.xml), nel quale vi sono memorizzate tutte le evidenze digitali importate dal database, mentre gli altri file, costituiti da diversi formati, contengono altre informazioni.

```
<database name='MMSSMS'>
<table name='sms'>
<row>
    <col name='_id'>977</col>
    <col name='thread_id'>15</col>
    <col name='address'>YYYYYYYYYYYY</col>
    <col name='person'>1148</col>
    <col name='date'>1265591133661</col>
    <col name='protocol'>0</col>
    <col name='read'>1</col>
    <col name='status'>-1</col>
    <col name='type'>1</col>
    <col name='reply_path_present'>0</col>
    <col name='subject'>null</col>
    <col name='body'>Text of the message</col>
    <col name='service_center'>XXXXXXXXXXXX</col>
</row>
</table>
</database>
```

Figura 2.2: File export.xml

É inoltre possibile, tramite il processo inverso EIP (Evidence Import Process), importare evidenze precedentemente esportate, fornendo la possibilità di alterare le evidenze. L'EIP è quindi capace di recuperare uno stato precedente del dispositivo, in modo da raggirare l'analisi forense.

2.2 Tecniche di Analisi Forense

Con il termine *analisi forense* indichiamo la scienza che studia l'individuazione, la conservazione, la protezione e l'estrazione di informazioni e documenti in formato digitale, reperibili da un qualunque dispositivo elettronico in grado di memorizzare tracce digitali.

I cellulari rappresentano, senza dubbio, l'evoluzione tecnologica piú rilevante degli ultimi anni, come evidenzia la massiccia diffusione di cui tali dispositivi sono stati oggetto ultimamente. Purtroppo, mentre da un lato il numero di cellulari utilizzati in attività criminali è sempre piú elevato, dall'altro la capacità di effettuare operazioni di analisi forense su tali dispositivi è notevolmente limitata da problemi di carattere tecnologico e metodologico.

In particolare, è opportuno sottolineare che le attuali tecniche di analisi forense faticano a tenere il passo della cosiddetta *anti-forensic*, una disciplina in continua fase evolutiva, che si pone come obiettivo principale lo studio di una serie di metodologie atte a focalizzare la propria attenzione su diversi procedimenti che possano compromettere la disponibilità e l'usabilità di tracce ed evidenze digitali utilizzabili durante il processo di analisi forense.

La disponibilità delle evidenze digitali puó essere compromessa nascondendo la loro esistenza o manipolandole a proprio piacimento; l'usabilità, invece, puó essere compromessa cancellando o falsificando le tracce digitali.

Nonostante le innumerevoli difficoltà che si presentano, l'analisi forense cerca comunque di sviluppare tecniche e metodologie efficaci che possano, da un lato, prevenire i problemi relativi all'anti-forensic, dall'altro elaborare strategie e strumenti innovativi che permettano uno studio corretto ed approfondito delle evidenze digitali reperibili dal dispositivo elettronico che si intende analizzare.

Entrando piú nello specifico, l'attenzione verrà focalizzata sulle tecniche e sugli strumenti attualmente utilizzati per eseguire operazioni di analisi forense

su dispositivi elettronici configurati con il sistema operativo Android.

Le metodologie attualmente utilizzate per analizzare le tracce digitali dei dispositivi Android possono essere raggruppate in categorie, specificando i tool disponibili e le tecniche per l'utilizzo piú appropriato di tali tool:

- Android Debug Bridge (ADB): strumento software distribuito con l'SDK Android, che permette l'interazione tra il dispositivo elettronico ed una workstation remota (ad esempio, un laptop). Puó essere definito come un applicativo Client-Server che rende possibile l'esecuzione remota di comandi shell all'interno del dispositivo. Tali comandi sono strettamente limitati se si è sprovvisti dei permessi di root user, che sono comunque facilmente ottenibili tramite un'apposita configurazione del dispositivo. Molti di questi comandi sono fondamentali nell'analisi forense, dal momento che determinate stringhe possono permettere l'estrazione di tutti i dati presenti nella memoria interna.

- Nandroid Backup: un insieme di applicativi software che supportano operazioni di backup e ripristino per dispotivi Android provvisti dei privilegi di root; una caratteristica importante è che Nandroid supporta pienamente l'immagine di memoria flash che incorpora tutti i dati interni del telefono, e che puó essere manipolata tramite una modalità di avvio particolare. Questa tecnica è in grado di aiutare notevolmente le operazioni di analisi forense, dal momento che permette di recuperare tutte le evidenze digitali cancellate dalla memoria interna del dispositivo o dal file system del sistema operativo.

- Comandi seriali tramite USB: in alcuni dispositivi Android è disponibile la connessione seriale tramite USB, che potrebbe essere sfruttata per intercettare dati ed informazioni trasmessi attraverso la rete. Pur essendo molto utile, questa tecnica presenta diversi limiti, dovuti in particolare

all'incompatibilità con alcuni dispositivi, e richiede ancora diversi casi di testing per poterla utilizzare in modo corretto e performante.

- Simulazione scheda SD: l'idea alla base di questo approccio è quella di utilizzare un file di aggiornamento modificato, in modo da prevenire la distruzione dei dati presenti nella memoria interna ed utilizzare strumenti software a livello kernel che supportino l'acquisizione di tali dati. Pur essendo una valida alternativa alle tecniche precedenti, questo tipo di procedimento, peró, non è ancora stato testato, e richiede una quantità di lavoro maggiore per poter essere utilizzato.

- Applicazioni software: ultima, ma non meno importante, è la possibilità, come in un qualunque smartphone attualmente in commercio, di utilizzare il sistema operativo Android per progettare e sviluppare applicazioni personali che abbiano l'obiettivo di recuperare tracce ed evidenze digitali da uno specifico dispositivo, in modo da poterle poi utilizzare in ambito forense.

Capitolo 3

Caso di studio: Analisi Logica

Il dispositivo utilizzato per il caso di studio è un HTC Desire con Android v2.3.3, è comunque utile sapere che la ROM è CyanogenMod v.7(versione modificata a partire da quella ufficiale di Android). In questo capitolo vengono introdotti i software utilizzati per l'analisi logica del dispositivo. Inoltre vengono specificati i risultati ottenuti con questi strumenti e le relative considerazioni.

Gli strumenti software utilizzati per l'analisi logica sono Oxygen e Unyaffs.

3.1 Oxygen

Oxygen Forensic Suite 2011 è un software per l'analisi forense su dispositivi mobili che permette di analizzare i file a livello logico. Per poter utilizzare questo software è necessario essere in possesso di un computer con sistema operativo Windows (XP, Vista, 7) e con almeno 100Mb di spazio libero su disco.

Per estrarre le informazioni è stato necessario scaricare i driver necessari per il riconoscimento del telefono HTC Desire. Completata la procedura di installazione del software, si è provveduto a collegare il dispositivo al computer portatile tramite cavo usb.

Tramite una semplice operazione di connessione, con Oxygen è stato possibile recuperare i seguenti dati:

- Informazioni generali del dispositivo;
- Rubrica;
- Messaggi;
- Registro eventi (chiamate perse, ricevute ed effettuate);
- Agenda;
- Cronologia;
- Dati memorizzati nella SD Card
- Dati memorizzati nella partizione Data del sistema.

3.1.1 Informazioni generali del dispositivo

In questa sezione vengono recuperate le informazioni intrinseche del dispositivo ed inoltre una statistica generale di quanto è stato esportato. (Figura 3.1) Nella figura 3.2 è possibile visualizzare le informazioni del dispositivo che Oxygen è riuscito a recuperare. È importante notare che *Nome Venditore* e *Produttore* risultano errati questo, con molta probabilitá, perchè il software ha recuperato quest'informazione da qualche componente interna.

HTC Desire

Nome attributo	Valore attributo
Informazione Generale	
Nome telefono	HTC Desire
Nome Vendite	Sony Ericsson Desire
Produttore	Sony Ericsson
Nome interno	HTC Desire
Hardware:	357841032955177
Versione Hardware	00
Versione Software	2.3.3
strOwnerPhoneNumber	3807515184
Letto nella versione	3.3.0.270
Ora di estrazione	26/05/2011 15:01:22
IMSI	222014801125399
SerialNumber	HT04KPL02498
ICCID	893901000012547092 94
Numero di telefono	3807515184

Figura 3.1: Informazioni dispositivo Oxygen

Sezione	Statistica
Rubrica	140
Contatti	140
Gruppi Chiamanti	6
Messaggi	Sezione non letta
Registro Eventi	500
Chiamate ricevute	68
Chiamate effetuate	265
Chiamate perse	167
Agenda	61
Appuntamenti	30
Tutti gli eventi del giorno	31
Sfoglia Risorse	11867
Imagini	3612
Melodie	62
Video	6
Documenti	226
Applicazioni	1
File database	215
Altri file	7745

Figura 3.2: Statistice esportazione Oxygen

3.1.2 Rubrica

Nella sezione rubrica sono memorizzate tutte le informazioni relative ai contatti. Oxygen, al termine del processo di estrazione dei dati dalla rubrica, permette di visualizzare sia le informazioni base(nome del contatto, numero di telefono, gruppi chiamanti) sia tutte le informazioni memorizzate da applicazioni di terze parti,(Email, informazioni memorizzate nell'account dei social network) sia il valore hash md5 con cui viene codificato il contatto. Nello specifico le informazioni recuperate sono:

- Foto
- Nome
- Lavoro
- Telefono
- Email
- Indirizzo
- Note
- Privato
- Gruppo
- MD5

Il limite riscontrato in questa sezione é che non vengono visualizzate le immagini di tutti i contatti.

3.1.3 Messaggi

Nella sezione messaggi vengono visualizzati tutti i messaggi memorizzati sul dispositivo con tutte le informazioni relative:

- Tipo(sms/mms)
- Cartella
- Contatto
- Numero tel.
- Ora
- Testo
- MD5

3.1.4 Registro eventi

In questa sezione viene mostrato tutto quello che riguarda le chiamate effettuate, ricevute e perse. Si è constatato che l'ora di ogni singola chiamata estratta tramite Oxygen non coincide con l'ora memorizzata sulla corrispondente chiamata presente sul dispositivo. Le chiamate in Oxygen risultano traslate un'ora avanti. Quindi è stata effettuata un'ulteriore estrazione dopo aver annullato l'opzione ora legale dal dispositivo e dopo aver effettuato una chiamata di prova, in modo da verificare se il problema fosse legato ad Oxygen o alle impostazioni del dispositivo stesso. Il risultato della prima estrazione è stata la lista di chiamate effettuate a partire dal 10/02/2011, delle chiamate perse dal 11/02/2011 e delle chiamate ricevute a partire dal 14/02/2011.
La seconda ha riportato le chiamate effettuate, ricevute e perse dal 15/02/2011 e l'ora dell'ultima chiamata è risultata conforme all'ora in cui era stata effettivamente eseguita. Questo perchè, mentre Android setta automaticamente le impostazioni relative all'ora legale, Oxygen calcola l'ora in base al fuso orario corrente, senza tener conto di eventuali pre-impostazioni.

3.1.5 Agenda

In questa sezione vengono visualizzate le informazioni memorizzate nell'agenda del cellulare:

- Tipo
- Inizio
- Fine
- Allarme
- Testo
- Memo
- Posizione
- Ricorrenza
- MD5

Anche per quanto riguarda gli orari degli eventi, nella prima estrazione, l'ora estratta da Oxygen è risultata translata rispetto a quella reale.

3.1.6 Risorse

La suite Oxygen garantisce, inoltre, l'estrazione completa ed efficiente di tutte le informazioni memorizzate nella memoria esterna ed interna del dispositivo. É possibile, grazie ad una classificazione gerarchica di tutti i dati presenti in memoria, sfogliare comodamente le cartelle per analizzarle nei minimi particolari. Le principali cartelle, dalle quali è possibile reperire le informazioni piú rilevanti, sono:

- /sdcard/Android: directory molto importante perchè adibita alla memorizzazione di informazioni utili per il corretto funzionamento delle app

scaricate dal market. Contiene, ad esempio, la cache delle applicazioni di localizzazione e i dati necessari per l'avvio di una specifica app.

- /sdcard/Bluetooth: directory che tiene traccia delle diverse operazioni effettuate tramite connessione bluetooth. Ad esempio, conserva in memoria le informazioni di un contatto trasferite attraverso la rete, o gli specifici file presenti sul telefono al termine di un'operazione di trasferimento.

- /sdcard/DCIM/Camera: è la directory che conserva i dati relativi alla fotocamera, in particolare immagini e video

- /sdcard/External_SD: oltre alla sd_card inserita all'interno del dispositivo, che non puó essere rimossa se il telefono è acceso, è possibile usufruire di un ulteriore scheda di memoria esterna, rimuovibile in qualsiasi momento, le quali informazioni sono memorizzate nella cartella External_SD.

- /sdcard/Download: directory che memorizza tutti i dati scaricati attraverso la connessione internet. Qualunque dato recuperato dalla rete viene memorizzato in questa cartella specifica per poter esser recuperato al momento dell'utilizzo successivo.

Le cartelle sopra descritte sono directory di default all'interno della scheda interna, proprio perchè adibite alla memorizzazione dei dati principali. Oxygen provvede ad estrarre anche moltissime altre cartelle e moltissimi altri file, ognungo, peró, relativo ad un'applicazione specifica. Quando un'app viene avviata per la prima volta, nella maggior parte dei casi il sistema provvede a creare una directory specifica per quell'applicazione, dove verranno salvati tutti i dati ad essa relativi. Solitamente, ad ogni app installata nel dispositivo è quasi sempre associata una cartella all'interno della memoria dello smartphone. Con Oxygen sarebbe possibile estrarre ed analizzare sia le cartelle, sia i dati e le informazioni in esse contenuti.

L'estrazione di tutti i file dalla memoria interna non è stata effettuata per il troppo tempo richiesto dall'operazione. Infatti dopo 4 ore dall'inizio dell'operazione, sono stati prelevati solo 313Mb di dati su 3Gb.

3.1.7 Cronologia

Oxygen provvede anche ad inserire in una sezione apposita, la cronologia, tutti gli eventi e le informazioni presenti nel telefono riguardanti la lista completa delle chiamate e gli appuntamenti in agenda. Pur avendo ciascuna delle due una sezione a parte interamente dedicata, la cronologia permette una sorta di fusione per analizzare in modo incrociato le diverse chiamate con gli eventuali appuntamenti associati. L'operazione ha lo scopo di eseguire una sorta di confronto tra date e chiamate, per verificarne un possibile collegamento.

3.2 Unyaffs

Unyaffs è un software open source molto pratico e funzionale. La caratteristica principale del programma consiste nell'estrarre dati ed informazioni da un'immagine dei file system Yaffs. Dal momento che Android usufruisce proprio di questo tipo di file system, Unyaffs si è rivelata la scelta più efficace per portare a termine la nostra analisi. Tuttavia, per poter ottenere un risultato completo e soddisfacente, si è reso necessario l'utilizzo di **mkyaffs2image**, un'utility, meglio descritta nel paragrafo successivo, che permette, fondamentalmente, di creare un'immagine del file system yaffs. Una volta ottenuta l'immagine del file system, si è potuto utilizzare unyaffs per l'estrazione dei dati. Le cartelle di maggiore interesse per la nostra analisi, descritte dettagliatamente in seguito, sono state:

- **anr**
- **dalvik-cache**

- **data**

Entrando più nel dettaglio dell'analisi logica, come prima cosa sono state effettuate le seguenti operazioni:

- reperito il dispositivo, la batteria è stata rimossa in maniera brutale;
- la SD card del dispositivo è stata sostituita con una SD card vergine;
- è stata inserita nuovamente la batteria;
- il dispositivo è stato avviato in modalitá di recovery, dopodiché è stato collegato al computer tramite cavo usb;
- sul computer è stato avviato il server adb per poter effettuare l'immagine della partizione /data;
- l'immagine è stata fatta utilizzando mkyaffs2image, eseguendo il comando come mostrato nella figura 3.3;
- per navigare l'immagine ottenuta è stato utilizzato unyaffs, che ha prodotto una cartella contente i file del filesystem.

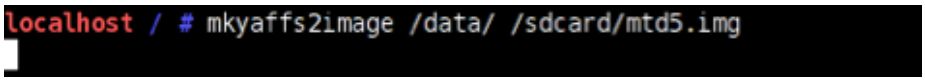

Figura 3.3: Comando mkyaffs2image

Di seguito vengono descritti più dettagliatamente i tool utilizzati e le cartelle, della partizione Data, ove si è riscontrato la presenza di informazioni di interesse.

3.2.1 mkyaffs2image

Mkyaffs2image è un utility che permette di creare un'immagine del filesystem yaffs e yaffs2. È possibile ottenere questa utility installando Nandroid Backup

o BusyBox. Quest'ultimo è un software libero, che permette di estendere le funzionalitá di Android aggiungendo alcuni comandi Linux di base e utility autonome create ad hoc per Android. Originariamente era nato per permettere di memorizzare su un solo floppy un sistema completo. Poi è diventato uno standard per i dispositivi Linux embedded.

3.2.2 anr

ANR è l'acronimo di Application Not Responding. File riscontrati:

traces.txt.bugreport **trace.txt**	nel caso in cui un'applicazione non risponda per un determinato periodo ad un evento, questi file tengono traccia degli eventi. All'interno viene scritto il pid del processo, la data e l'orario in cui è avvenuto il crash e i thread del processo. Quindi è possibile elaborare queste informazioni per comprendere se il dispositivo era acceso o era utilizzato nel momento del crash in base all'applicazione ritrovata all'interno del file

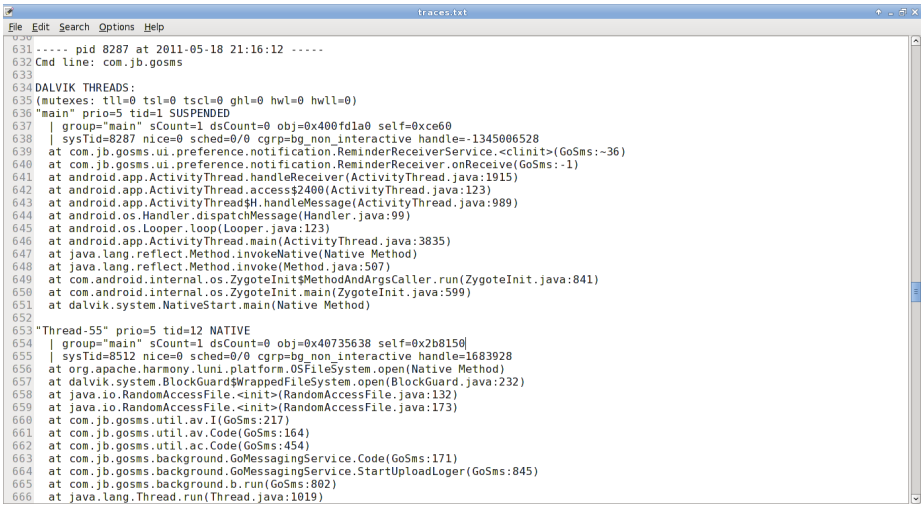

Figura 3.4: File traces.txt

3.2.3 dalvik-cache

Sul dispositivo analizzato è vuota perchè questi dati vengono salvati sulla memory card esterna. All'interno ci sono i file creati da ogni istanza della Dalvik virtual machine in modo da ottimizzarne l'esecuzione.

3.2.4 data

Ogni applicazione ha un proprio spazio privato (cartella) dove memorizzare le informazioni che ritiene necessarie. Ovviamente essendo privato un'applicazione non puó accedere alla cartella di un'altra applicazione. Verranno analizzate solo le cartelle delle applicazioni che possono memorizzare dati di interesse, resta comunque possibile visualizzare qualsiasi dato salvato dalle applicazioni. Di seguito la descrizione dettagliata dei file di interesse.

android.tether

Applicazione di terza parte che permette di utilizzare il tethering.

Directory	Informazioni di interesse
./var	All'interno del file **tether.log** vengono memorizzate tutte le informazioni relative all'utilizzo della di questa feature.

cn.goapk.market

Applicazione di un market non ufficiale.

Directory	Informazioni di interesse
./databases	Nella tabella **soft_managen** di **goapk_asset.db** è possibile recuperare per ogni applicazione installata sul sistema, il nome, il package name, la versione, eventuali commenti dell'utente.
	In **goapk_download.db** è possibile recuperare delle informazioni relative alle applicazioni scaricate attraverso questo market. Le informazioni ottenute sono reperibili dalla tabella **downloads**. I campi rinvenuti sono: l'url attraverso il quale sono state scaricate, il percorso dove sono state salvate, la versione e il nome del package..

com.adobe.reader

Applicazione che permette la lettura di file pdf.

Directory	Informazioni di interesse
./shared_prefs	All'interno del file **com.adobe.reader.preferences.xml** è possibile trovare una lista degli ultimi file aperti comprensivi di path.

com.android.browser

Applicazione browser di default.

Directory	Informazioni di interesse
./app_databases	Ci sono i dati memorizzati nel local storage di ogni pagina visitata.
./app_geolocation	La tabella **CachedPosition** di **CachedGeoposition.db** contiene altitudine, longitudine, altezza, direzione e accuratezza, in piú ha anche il timestamp.
./app_plugins	è stata rilevata la cartella del plugin per il flash player contenente alcuni cookie delle pagine visitate..
./app_icons	Vengono ritrovati diversi file, che opportunamente combinati possono aiutare a ricreare la cronologia parziale o totale del browser.
./databases	In **browser.db** è possibile trovare la lista dei preferiti, con data e ora di creazione, data di ultima visita. In piú è possibile trovare le ricerche effettuate con data e ora.
	webview.db contiene: • le password memorizzate dall'utente nel formato host, username e password. • le password httpAuth memorizzate con lo stesso formato della tabella precedente. • tutte le informazioni introdotte in form di compilazione, utile per la feature di auto-completamento. • i cookies.
	webviewCache.db organizza i file e i dati presenti nella cartella cache.

com.android.providers.calendar

Applicazione agenda di default sui dispositivi Android.

Directory	Informazioni di interesse
./databases	All'interno di **calendar.db** vengono rinvenute informazioni circa tutti gli eventi salvati dall'utente.

com.android.providers.contact

Cartella privata dell'applicazione dei contatti.

Directory	Informazioni di interesse
./databases	All'interno di **contact2.db** è possibile trovare informazioni circa: • gli account che sono utilizzati per sincronizzare la rubrica (per esempio mail e account facebook). • cronologia delle chiamate con relativa data, ora, durata, tipo chiamata, numero chiamata e nome del contatto nella rubrica • contatti, comprensivi di mail. Su alcuni contatti è possibile trovare le foto, da notare che sul dispositivo ogni contatto ha una foto, poiché viene sincronizzata da facebook.le password memorizzate dall'utente nel formato host, username e password.

com.android.providers.downloads

Cartella del download manager di default

Directory	Informazioni di interesse
./databases	All'interno di **downloads.db** è possibile trovare informazioni circa tutti i file scaricati attraverso il download manager di Android. I campi della tabella in questione sono: url, nome file, file path, nome del file, l'applicazione che ha avviato il download.

com.android.providers.media

Applicazione di default per i file multimediali

Directory	Informazioni di interesse
./databases	All'interno di **external-*NUMERO*.db**, dove il numero è in funzione della memoria esterna, è possibile ricavare: • lista degli album. • copertine degli album ascoltati. • playlists. • immagini e video. Per questo tipo di file è possibile conoscere la provenienza del file (per esempio se viene da un programma, se è stata scaricata o se un'immagine scattata dalla camera). Chiaramente è possibile leggere tutti i metadata di questi file.
	All'interno di **internal.db** è possibile trovare le stesse informazioni del database precedente, ma riguardano i file salvati nella memoria interna.

com.android.providers.settings

Applicazione che permette all'utente di settare le impostazioni del sistema

Directory	Informazioni di interesse
./databases	**settings.db** mostra quale applicazione di default viene lanciata per i servizi base come: musica, messaggi, calendario, email, contatti, browser. In piú specifica anche le impostazioni del sistema.

com.android.providers.telephony

Applicazione che si occupa di gestire la parte telefonica del dispositivo

Directory	Informazioni di interesse
./databases	**mmssms.db** mostra: • il contenuto degli sms e mms. • se il messaggio è stato letto. • la data di invio o ricezione.
./app_parts	è possibile trovare tutti i file multimediali inviati come allegato in un mms.

com.android.providers.vending

Market ufficiale di Android

Directory	Informazioni di interesse
./databases	In **assets14.db** c'é la lista delle applicazioni installate dal market, le relative informazioni come auto-update e package.
	suggestions.db contiene le ricerche effettuate attraverso il market con la relativa data.
	webview.db contiene i cookies attraverso il quale si accede al market android.

com.dropbox.android

Applicazione dropbox per Android

Directory	Informazioni di interesse
./databases	**db.db** mostra la lista dei file contenuti nella dropbox, con data di modifica e path.
./files	**log.txt** viene salvato il log ed è possibile visualizzare gli accessi, le modifiche effettuate alla cartella dropbox.

com.facebook.katana

Applicazione default per Facebook

Directory	Informazioni di interesse
./databases	**fb.db** mostra: • le info dell'account che si collega a facebook e la sua password cifrata. • le sessioni salvate con data e ora di accesso o uscita dal servizio. • le notifiche di facebook che ancora devono essere visualizzate. • i messaggi della mail di facebook: contenuto, tempo, mittente e se è stato letto. • la lista degli amici: nome, cognome, immagine del profilo, data di nascita, email, cellulare. • non è stato possibile recuperare le conversazioni tra due contatti con le relative informazioni di contorno.

com.fring

Applicazione di chat e videochat

Directory	Informazioni di interesse
./databases	**fring.db** contiene la lista dei contatti e delle conversazione priva di contenuto.
	google_analytics.db contiene la lista delle operazioni effettuate, quasi come un file di log. I campi ritrovati sono id utente, categoria dell'operazione, timestamp per la prima volta che viene effettuata una certa operazione. In piú la tabella tiene traccia delle ultime due volte che viene effettuata l'operazione in questione. In un'altra tabella dello stesso database è possibile trovare la lista delle sessioni con il tempo di accesso al servizio.

com.google.android.apps.chrometophone

Applicazione ufficiale di Google che permette di interfacciare il dispositivo con il borwser Chrome installato su un computer

Directory	Informazioni di interesse
./databases	**history.db** nel caso in cui il dispositivo sia sincronizzato con Google, è possibile trovare la lista di link inviati dal browser chrome di un computer al dispositivo. I campi rinvenuti sono: url e titolo.

com.google.android.apps.maps

Applicazione ufficiale di Google Maps

Directory	Informazioni di interesse
./databases	**da_destination_history** contiene lo storico dei percorsi ricercati in google maps, con indirizzo di partenza, indirizzo di destinazione, coordinate, data e ora.
./files	**DATA_preferences** contiene le ultime ricerche effettuate da google maps.

com.google.android.apps.translate

Applicazione ufficiale di Google Translate

Directory	Informazioni di interesse
./files	**historydb** contiene le ultime ricerche effettuate, con le relative traduzioni e descrizioni aggiuntive

com.google.android.apps.reader

Applicazione ufficiale per leggere feed rss

Directory	Informazioni di interesse
./cache	nella cartella **webviewCache** è possibile trovare le immagini contenute nei feed letti.
./databases	**reader.db** contiene: • username dall'account con cui è collegato. • lista dei feed a cui è iscritto, comprensive di url. • lista dei singoli articoli, comprensive di titolo e url. **webviewCache.db** permette di collegare le immagini contenute nella carella webviewCache ai feed che le contenevano.
./files	è possibile trovare i file html relativi a gran parte degli ultimi feed letti. I nomi dei file sono contenuti nel campo _id della tabella item del database **reader.db**.

com.google.android.gm

Applicazione ufficiale di Google Mail

Directory	Informazioni di interesse
./databases	**downloads.db** contiene la lista degli allegati scaricati dalle mail. I file originali risiedono nella cartella **cache/download**.
	mailstore.*NOMEACCOUNT*@gmail.com.db contiene: • la lista dei filename degli allegati e i relativi id che permettono di fare join con le altre tabelle ed avere, cosí, informazioni piú complete. • la lista delle conversazioni, i campi sono: oggetto, snippet e destinatario. • la lista delle mail collegate all'account. • la lista delle mail inviate e ricevute. I campi sono: from, to, cc, bcc, data invio, data recezione, oggetto e snippet.
	suggestions.db contiene le ultime ricerche effettuate nell'applicazione (Es. ricerca del testo di una mail).

com.google.android.googlequicksearchbox

Applicazione di ricerca sul dispositivo

Directory	Informazioni di interesse
./databases	**qsb-log.db** contenente le ultime ricerche effettuate. I campi sono: parole ricercate, url cliccato, data e ora.

com.google.android.gsf

Applicazione ufficiale per Google Talk

Directory	Informazioni di interesse
./databases	**talk.db** contiene i contatti di google Talk e gli ultimi messaggi.

com.google.android.location

Servizio di localizzazione del dispositivo

Directory	Informazioni di interesse
./files	**cache.cell** contiene informazioni riguardanti le ultime celle alle quali il cellulare si è collegato. I campi sono: key cell(mcc), accuratezza, confidence (ovvero sicurezza della rilevazione), latitudine, longitudine e timestamp.
	wifi.cell contiene informazioni riguardanti gli access point (wifi) ai quali il cellulare si è collegato. I campi della tabella risultano gli stessi del file precedente tranne per il KeyCell, poiché in questo file viene salvato il mac address dell'access point.

com.google.android.street

Applicazione ufficiale per Google Street View

Directory	Informazioni di interesse
./cache	è possibile trovare una serie di immagini riguardanti le ultime visualizzazioni in streetView.

com.google.android.youtube

Applicazione ufficiale per la visualizzazione dei servizi di YouTube

Directory	Informazioni di interesse
./databases	**history.db** contiene le ultime ricerche effettuate su youtube. I campi rinvenuti sono query, e timestamp in cui è stato effettuata la ricerca.

com.google.zxing.client.android

Applicazione di barcode scanner

Directory	Informazioni di interesse
./data	**barcode_scanner_history.db** contiene la cronologia degli ultimi codici letti. I campi ritrovati sono testo letto, tipo di codice letto e timestamp.

com.skype.raider

Applicazione Skype

Directory	Informazioni di interesse
./data	***NOMEACCOUNT*** contiene i partecipanti alle ultime chat e la lista degli eventi notificati da skype.

mobi.mgeek.TunnyBrowser

Applicazione browser Dolfin

Directory	Informazioni di Interesse
./app_*	vengono memorizzati dati su richiesta delle pagine web visitate.
./databases	**browser.db** contiene: • tutti i segnalibri, comprensivi di titolo, url e data di creazione. • la cronologia con titolo, url, data di creazione, ultima data di visualizzazione e numero totale di visualizzazioni. • le ricerche effettuate con la relativa data. **downloads.db** contiene gli ultimi download effettuati. I campi rinvenuti sono: url, nome file, path in cui è stato salvato e data completamento download. **google_analytics.db** contiene la lista delle operazioni effettuate. I campi ritrovati sono id utente, categoria dell'operazione, timestamp per la prima volta che viene effettuata una certa operazione. In piú la tabella tiene traccia delle ultime due volte che viene effettuata l'operazione in questione. In un'altra tabella dello stesso database è possibile trovare la lista delle sessioni con il tempo di accesso al servizio.

Directory	Informazioni di Interesse
	webview.db contiene: • le password memorizzate dall'utente, nel formato host, username e password. • le password httpAuth memorizzate nello stesso formato della tabella precedente. • tutte le informazioni introdotte in form di compilazione, utile per la feature di autocompletamento. • i cookies. **webviewCache.db** organizza i file e i dati presenti nella cartella cache.

3.2.5 local

Local contiene una cartella "download" dove è possibile ritrovare degli apk temporanei.

3.2.6 misc

Misc contiene le diverse impostazioni di sistema:

Directory	Informazioni di interesse
./bluetoothd	**names** contiene la lista di tutti i dispositivi bluetooth rilevati dal device. I campi del file sono: mac address e nome del dispositivo.
	lastsee per ogni dispositivo bluetooth rilevato memorizza la data dell'ultima volta che è stato rilevato. I campi del file sono: mac address e data.
	profiles contiene la lista dei dispositivi bluetooth accoppiati con il device.
	lastused contiene informazioni circa l'ultima volta che il bluetooth del device ha comunicato con altri bluetooths. I campi del file sono: mac addred e data
./wifi	**wpa_supplicant.conf** contiene informazioni su tutte le reti wireless alle quali il dispositivo si è associato. I campi del file sono: ssid, tipo cifratura ed eventuale password in chiaro.

3.2.7 system

La cartella system contiene informazioni circa il sistema e la sua struttura.

Directory	Informazioni di interesse
./registered_services	**android.accounts.AccountAuthentication.xml** e **android.content.SyncAdapter.xml** contengono la lista delle applicazioni che utilizzano la sincronizzazione dei dati. Questa feature puó essere attivata o disattivata dall'utente.
	packages.list contiene la lista dei pacchetti installati sul dispositivo.

3.3 Oxygen vs Unyaffs

Oxygen e Unyaffs sono due software molto diversi; infatti Oxygen è un complesso tool forense commerciale, molto conosciuto e diffuso, mentre Unyaffs è un piccolo software open source, la cui unica funzionalità è quella di analizzare l'immagine di un disco avente file system YAFFS e creata con il tool mkyaffs2image, per estrarne la struttura gerarchica delle cartelle e file.

Nonostante Oxygen sia conosciuto come uno dei migliori software forensi per dispositivi mobili, lo abbiamo trovato piuttosto carente per quanto riguarda la quantità di informazioni che è stato in grado di esportare dal cellulare.

Oxygen, al momento dell'estrazione delle informazioni, installa sul dispositivo un'applicazione alla quale vengono assegnati tutti i permessi previsti dal sistema operativo Android, e tramite questa applicazione vengono esportate tutte le informazioni possibili. In altre parole, viene effettuato l'evidence export process di cui si è parlato precedentemente.

Gli svantaggi di un tale approccio sono diversi; in primo luogo l'istallazione dell'applicazione sul dispositivo implica la modifica della partizione /data/ del dispositivo, che potrebbe compromettere parte delle informazioni eliminate precedentemente. Il secondo svantaggio di un tale approccio è che l'applicazione installata è limitata dal sistema di sicurezza di Android e non può, quindi, accedere ai dati di altre applicazioni o del sistema operativo.

Se Oxygen prevedesse la possibilità di essere eseguito su un dispositivo con la root, cioè sul quale è possibile ottenere i privilegi di super user, allora potrebbe ottenere ulteriori informazioni che lo renderebbero più interessante rispetto al tool Unyaffs.

Unyaffs, invece, permette di accedere all'intero file system della partizione /data/, rendendo possibile estrarre una quantità molto superiore di informazioni, senza comprometterne la validità, dato che per ottenere il dump la partizione è stata montata in sola lettura.

In conclusione, Unyaffs si è rivelato molto più utile per i nostri scopi; tuttavia se venisse migliorata la compatibilità del software Oxygen con Android, sicuramente sarebbe un'alternativa altrettano valida.

Capitolo 4

Caso di studio: Analisi Fisica

Il dispositivo utilizzato per il caso di studio è un HTC Desire con Android v2.3.3, è comunque utile sapere che la ROM è CyanogenMod v.7(versione modificata a partire da quella ufficiale di Android). In questo capitolo vengono introdotti i software utilizzati per l'analisi fisica del dispositivo. Inoltre vengono specificati i risultati ottenuti con questi strumenti e le relative considerazioni.

Gli strumenti software utilizzati per l'analisi fisica sono Scalpel e Photorec.

4.0.1 Scalpel

Scalpel è un software forense open source e gratuito, sviluppato a partire da Foremost. Entrambi questi software sono in grado di effettuare "data carving", cioè di estrarre file a partire da immagini di dischi, senza conoscere il file system utilizzato. Questo è possibile in quanto operano a livello di bit, ed effettuano ricerche di precise sequenze che indichino la presenza di un file.
Scalpel esamina i dati che gli vengono forniti alla ricerca di header predefiniti, ciascuno dei quali indica l'inizio di un file in una determinata posizione del disco. Grazie a questo strumento è stato possibile esaminare l'immagine della memoria interna del dispositivo android che, utilizza come file system YAFFS2, ed

essendo quest'ultimo generalmente poco diffuso, sarebbe stato particolarmente complesso da montare in linux.

Scalpel è particolarmente efficace nel recupero di file cancellati, anche se il suo utilizzo ci ha causato non pochi problemi. Analizzando la partizione /data, di taglia pari a 300Mb, Scalpel ha estratto piú di 100GB di dati, rendendo di fatto impossibile, per questioni di tempo, un'analisi accurata di tutte le informazioni ottenute.

Il problema della disparitá di dimensioni è dovuto al fatto che, per ogni header rilevato, Scalpel considera il file terminato solo quando trova un footer dello stesso tipo, o quando arriva alla fine del file immagine.

Questa caratteristica del software comporta un enorme ridondanza di informazioni nei file recuperati. L'enorme mole di dati recuperati é dovuta anche alla presenza di numerose copie di uno stesso file. Inizialmente questa conseguenza era stata attribuita al funzionamento del file system YAFFS che, per come progettato, farebbe pensare alla possibilitá di avere in memoria piú copie dello stesso file. Successivamente, per confutare questa ipotesi, é stato effettuato un test su un altro dispositivo, un Samsung Galaxy S, provvisto del file system RFS, diverso da YAFFS. Il risultato é stato identico: anche analizzando il secondo dispositivo, sono state riscontrate piú copie degli stessi file. L'ovvia conclusione é stata quindi che la presenza di diversi duplicati, e quindi l'enorme mole di informazioni estratte tramite Scalpel, é dovuta alle memorie flash NAND installate in questi particolari dispositivi.

Per questo motivo abbiamo optato per l'utilizzo del software Photorec in concomitanza con Scalpel.

4.0.2 PhotoRec

Photorec è un software forense open source, che allo stesso modo di Scalpel consente di effettuare estrazioni di informazioni su raw data, senza quindi interessarsi al file system utilizzato per memorizzare le informazioni, la tabella delle partizioni ed eventuali corruzioni del file system del disco.
I concetti e le idee alla base del funzionamento di questo software sono uguali a quelle descritte prima ed utilizzati da Scalpel; tuttavia Photorec utilizza algoritmi di ricerca diversi e, pertanto, il set dei file ripristinati è notevolmente ridotto rispetto a Scalpel.
Per questo motivo, i file che Photorec è stato in grado di ripristinare sono solo una parte di quelli recuperati dall'altro tool forense. Ad esempio, molte immagini recuperate da Scalpel non sono state identificate da Photorec. Per ovviare a questa carenza del tool forense abbiamo sfruttato un'opzione fornita da Photorec, che consente di creare un dump a partire dall'originale, contenente solo le informazioni non ripristinate, nel nostro caso avente grandezza di circa un terzo, cioè 1GB.
Successivamente ci è stato possibile analizzare quest'immagine con Scalpel ed abbiamo, quindi, recuperato anche le ultime immagini.

4.0.3 Scalpel vs Photorec

Scalpel e Photorec sono entrambi concettualmente pressocchè identici; entrambi consentono l'estrazione su raw data, entrambi analizzano dump di dischi alla ricerca di un set predefinito di header, ed entrambi consentono di memorizzare o scartare i file corrotti.
Tuttavia, in quanto alla capacitá di analizzare in modo esaustivo il dump e recuperare il maggior numero di informazioni, Scalpel si è dimostrato nettamente superiore, superando spesso le nostre aspettative. Entrambi hanno estratto una

notevole quantitá di informazioni in modo ridondante e Scalpel in particolar modo, rendendo impossibile un'analisi accurata delle informazioni dato il tempo limitato a disposizione.

Non per questo motivo, peró, Photorec è da considerarsi migliore visto che non è stato in grado di estrarre, ad esempio, alcune immagini che Scalpel ha estratto e che sono risultate non essere neanche parzialmente corrotte.

In conclusione, a nostro giudizio, non vi è una superiorit á netta di uno di questi tool forensi rispetto all'altro. Analizzando i risultati ottenuti, la strategia piú efficace per recuperare la maggior quantitá possibile di informazioni è quella di analizzare inizialmente il dump con il software Photorec, impostato in modo da salvare in un apposito file le informazioni che non è stato in grado di interpretare, ed esaminare in seguito questo file con Scalpel.

4.0.4 Cancellazione e recupero di informazioni su filesystem YAFFS2

Per verificare il funzionamento del filesystem YAFFS2 e la gestione della memoria interna del sistema operativo Android abbiamo creato e successivamente eliminato diversi file. In seguito abbiamo analizzato la memoria nel tentativo di recuperare le informazioni cancellate. In questo modo ci è stato possibile capire appieno le possibilitá di ottenere informazioni cancellate, nonchè la gestione della memoria effettuata dal fs YAFFS. YAFFS infatti è progettato appositamente per le memorie flash, molto diffuse nei dispositivi mobili. Queste memorie hanno un numero limitato di letture e scritture per ogni blocco, oltre il quale le prestazioni possono deteriorarsi fino ad un eventuale fallimento del blocco. Ad ogni scrittura inoltre deve essere riscritto un intero blocco della memoria, il che rende cruciale sia dal punto di vista delle prestazioni, che per quanto riguarda la vita della memoria, che il file system riesca ad eseguire le operazioni in modo

ottimale. Date le caratteristiche appena descritte delle memorie flash nand, il file system in questione cerca di minimizzare e distribuire le scritture sui blocchi; quando un file deve essere modificato, invece di effettuare la modifica on place, che comporterebbe la riscrittura dell'intero blocco, la pagina del blocco viene modificata e scritta in un altro blocco inutilizzato, e la vecchia pagina viene marcata come inutilizzata (discarded).

Per creare, eliminare, ed analizzare le informazioni di interesse abbiamo effettuato le seguenti operazioni:

Procedura

1. Creazione ed invio di un nuovo SMS
2. Creazione ed invio di un nuovo MMS, contenente un'immagine scattata dalla fotocamera dall'applicazione di messaggistica di android
3. Creazione di un file di testo all'interno della cartella di sistema /data/misc/
4. Modifica del file di testo appena creato
5. Cancellazione del file di testo
6. Cancellazione dei messaggi precedentemente inviati
7. Collegamento del dispositivo al computer tramite cavo usb
8. Avvio del server adb
   adb shell
9. Esecuzione della copia bit a bit della partizione /data ed analisi con "photorec"
   /system/bin/dd if=/dev/mtd/mtd5 of=/sdcard/dump/mtd5.img bs=4096
   photorec /log /d photorec/ mtd5.dd

10. Esecuzione della copia logica della partizione /data ed analisi con il software "unyaffs"
    mkyaffs2image /data/ /sdcard/mtd5.img
    unyaffs mtd5.img

11. Esecuzione dell analisi del dump di dd tramite editor esadecimale
    bless mtd5.img

Tutte le operazioni hanno richiesto un tempo di esecuzione pressocchè immediato, ad eccezione dei punti 9 - 10. Sia la copia bit a bit, sia la copia logica della partizione /data hanno impiegato un discreto lasso temporale per essere portate a termine. Il motivo riguarda, ovviamente, la grandezza della partizione che è stata copiata, ma i tempi di esecuzione sono comunque risultati essere inferiori ai 30 minuti.

Risultati

Tramite l'analisi logica del file system siamo riusciti a risalire al testo dei messaggi inviati e cancellati, dato che erano ancora presenti nel database dell'applicazione di messagistica di android. Questo risultato è molto importante in quanto ci dimostra che è possibile rintracciare alcune informazioni, credute eliminate dall'utente, senza dover ricorrere ad analizzare a livello fisico la partizione.
L'immagine del messaggio multimediale inviato e cancellato è stata recuperata tramite il software photorec; l'immagine recuperata era perfettamente integra.
Per quanto riguarda, invece, il file di testo che abbiamo creato, modificato ed in seguito cancellato, non è stato possibile come previsto recuperarlo con software di analisi logica. Neanche photorec è stato in grado di trovare questo file, né nella prima versione, né nella seconda.
L'unica alternativa è stata, quindi, ricorrere all'utilizzo di un editor esadecima-

le, attraverso il quale abbiamo trovato il contenuto del file sia prima che dopo essere stato modificato.

Capitolo 5

Anti-Forense

Continuando ad esaminare le tecniche di analisi forense applicabili sugli smartphone android, il passo successivo è stato quello di procedere modificando delle evidenze digitali e quindi di applicare delle tecniche anti-forensics.
Per anti-forensic si intende quell'insieme di tecniche necessarie per manomettere o cancellare tracce di operazioni illecite al fine di evitare che un'eventuale analisi possa portare alla luce le informazioni relative alle azioni effettuate. Così come un normale malintenzionato cerca di nascondere le tracce delle proprie malefatte, anche con i moderni dispositivi elettronici è possibile utilizzare tecniche apposite per la creazione di un falso alibi.
Sfruttando lo smartphone HTC Desire, il lavoro si è concentrato sulla modifica e sulla cancellazione di evidenze digitali relative ad applicazioni di geolocalizzazione. Lo scopo è stato quello di modificare determinati dati per testare la possibilità di camuffare la reale presenza del dispositivo in uno specifico luogo.

5.1 Un primo tentativo: Google Maps

Il primo tentativo è stato quello di sfruttare le molteplici opzioni messe a disposizione dall'applicazione Google Maps, provvista anche di navigatore. Dopo aver costruito e seguito un semplice itinerario, il passo successivo è stato recuperare le informazioni memorizzate dal navigatore Google Maps (mappe,suoni,percorsi) per cercare di modificare determinati dati e impostare date e luoghi completamente diversi da quelli reali: lo scopo era quello di camuffare la reale posizione dello smartphone per creare un alibi digitale attendibile.

Con Google Maps, sfortunatamente, l'esperimento ha dato esito negativo. Dopo alcune ricerche, si è scoperto che l'applicazione Google Maps salva tutte le informazioni relative al navigatore nella cartella **/sdcard/Android/Data/-com.google.androids.apps.maps**: all'interno della cartella sono stati trovati i file vocali e le informazioni relative ai percorsi effettuati. L'idea era quella di modificare sia le date di accesso e modifica a questi file, sia i file stessi, inserendo orari, luoghi e coordinate fittizi. Esclusi i file vocali, tutte le altre informazioni, all'interno di qualsiasi altro file, erano memorizzate in un formato sconosciuto, e non è stato possibile in alcun modo risalire al testo in chiaro per effettuare le modifiche necessarie.

Dal momento che non è stato possibile trovare una soluzione efficace che risolvesse il problema, e di conseguenza non si è riusciti in alcun modo ad accedere ai file necessari per la manomissione delle evidenze, si è deciso di intraprendere una strada diversa.

5.2 Il secondo tentativo: CoPilot

CoPilot è un'applicazione installabile per Android, iPhone, iPad e Windows Mobile, che consiste in un navigatore satellitare, molto efficiente e funzionale.

L'applicazione, come un qualsiasi navigatore satellitare, permette la pianifica-

zione di itinerari sfruttando la tecnologia GPS all'interno degli smartphone. A differenza di Google Maps, tutte le informazioni relative a orari, coordinate e percorsi effettuati vengono memorizzate in plain text all'interno di una cartella creata nella sd card al momento del primo avvio dell'applicazione.
La cartella **/sdcard/copilot**, dedicata all'applicazione CoPilot, è strutturata come segue:

- **/copilot/audio**: contiene i file audio di avviso, come ad esempio allarmi per gli autovelox o per il superamento dei limiti di velocità.
- **/copilot/EU**: contiene tutte le informazioni relative alle mappe scaricate; nel caso specifico, sono presenti solo le mappe dell'Italia
- **/copilot/[MESE][GIORNO][ANNO].gps**: la cartella fondamentale, che contiene tutti i dati relativi ai percorsi effettuati.
- **/copilot/info**: contiene informazioni generali sull'applicazione, come ad esempio la versione, e il numero di licenza delle mappe scaricate.
- **/copilot/log**: contiene due specifici file di log: uno relativo agli eventuali errori che si verificano durante l'esecuzione, ed un altro, più generico, che contiene tutte le altre informazioni utili.
- **/copilot/maps**: contiene una serie di mappe predefinite, adibite per lo più al cambio di stile di visualizzazione della mappa corrente. Ad esempio, è presente la mappa *autumn*, che viene caricata di default quando avviene il passaggio dal giorno alla notte, o la mappa *2D*, che puó essere caricata a discrezione dell'utente per ottenere uno stile di visualizzazione diverso.
- **/copilot/save**: contiene informazioni di configurazione del software Copilot.
- **/copilot/skin**: contiene una serie di temi predefiniti, comodamente modificabili per cambiare il tema principale dell'applicazione.

- **/copilot/speech**: contiene tutti i file audio relativi alle indicazioni date durante il tragitto dalla voce guida. Ogni pacchetto di file è associato ad una voce specifica, scaricabile tramite l'applicazione.
- **/copilot/uiconfig**: contiene i dati relativi al settaggio e al boot dell'interfaccia utente al momento dell'avvio dell'applicazione.

Tutti i file che si trovano all'interno delle sottocartelle sopra descritti contengono informazioni in chiaro, ed è quindi semplice accedere e modificare determinati dati. Nel prossimo paragrafo, verranno descritte in modo dettagliato tutte le operazioni effettuate per creare, tramite CoPilot, un alibi convincente per camuffare informazioni relative alla geolocalizzazione del dispositivo in uno specifico luogo, ed in uno specifico orario.

5.3 L'esperimento con Copilot

In questo paragrafo descriveremo la metodologia utilizzata per manomettere le tracce generate da Copilot.
Per limitare le incongruenze che possono venir fuori nella creazione dell'alibi, tutte le operazioni effettuate in seguito sono state eseguite dopo aver terminato l'applicazione telefono.

5.3.1 Analisi delle tracce

Per modificare le suddette evidenze non é stato sufficiente studiare il metodo di memorizzazione delle informazioni da parte del programma, in quanto questo ci ha dato solo una visione parziale delle tracce lasciate da una sua esecuzione. Informazioni quali file aperti, modificati, e creati durante la simulazione di un percorso sfuggono ad una analisi "statica"della memory card.

L'approccio utilizzato per venire a conoscenza di tutte le tracce da modificare é stato il seguente:

1. É stato istallato il navigatore sul dispositivo.
2. Copilot è stato avviato più volte ed è stato utilizzato su un breve percorso, in modo da assicurarci che tutte le informazioni come le voci, le mappe, p.o.i. (punti di interesse) e autovelox fossero state scaricate da internet.
3. É stato simulato un viaggio.
4. É stata fatta una copia bit per bit della memory card.
5. L'immagine è stata analizzata alla ricerca di informazioni che risalissero all'intervallo di tempo relativo alla simulazione del viaggio.

Sono stati annotati gli orari in cui le operazioni su descritte sono state effettuate. Poiché per modificare in modo coerente le informazioni é essenziale sapere con certezza durante quale fase i timestamps di ogni file vengono modificati. Ad esempio, sapere se la modifica di un file é avvenuta durante la simulazione di un viaggio o la prima esecuzione dell'applicazione. Di seguito viene riportato il log degli orari.

23-06-2011	
12:13	Rimosso la cartella "/sdcard/Copilot"
12:14	Avvio il navigatore
12:15/12:26	Scarico le voci e le mappe
12:26/12:28	Estrazione dati scaricati
12:29/12:50	Utilizzo e modifica impostazioni
12:50/13:05	Simulazione Percorso 1
15:43/16:00	Simulazione Percorso 2

La descrizione dettagliata della fase di analisi viene riportata in seguito. Per capire quali fossero le tracce lasciate da Copilot durante il suo utilizzo

abbiamo creato uno script bash che, se fornito di un percorso, una data, l'ora iniziale e l'ora finale, analizza la directory specificata e mostra un elenco dei file sui quali è stata compiuta una qualche operazione nell'intervallo di tempo specificato. Il suo funzionamento si basa sui timestamp associati ad i file, che vengono aggiornati al momento di creazione, modifica ed accesso degli stessi. Questo programma ci ha permesso di risalire ad alcuni file che, ad una prima analisi manuale, ci erano sfuggiti, e che vengono modificati ad ogni esecuzione di Copilot. Partendo da questa lista di file, abbiamo potuto notare che è possibile suddividerli in tre categorie, in base alle modifiche effettuate da Copilot durante la simulazione:

1. I file per cui viene aggiornata solo la data dell'ultimo accesso ad ogni esecuzione, e quindi il contenuto rimane inalterato.

2. I file per cui viene modificato il timestamp di "change".

3. I file il cui contenuto viene modificato.

In quest'ultimo caso, soltanto nei file trip.log e [MESE][GIORNO][ANNO].gps le informazioni che vengono scritte possono considerarsi indizi per risalire al momento in cui il navigatore è stato utilizzato, e dunque andranno modificate.

5.3.2 Il formato NMEA

Analizzando le diverse cartelle dell'applicazione CoPilot, la cartella contenente le informazioni più rilevanti è la cartella **/sdcard/copilot/[MESE][GIORNO][ANNO].gps**. Questa directory contiene una serie di file con estensione *.gps*, ognuno dei quali avente come nome la data in cui è stato effettuato uno specifico tragitto. Ogni file contiene informazioni dettagliate relative ad uno specifico percorso effettuato.
Entrando più nel dettaglio, è opportuno analizzare in maniera più specifica un

file Data.gps, al fine di comprenderne la struttura e il meccanismo di memorizzazione dei dati relativi ad un tragitto.

I file *.gps* sono scritti in formato **NMEA**. NMEA 0183, o più comunemente NMEA, è uno standard di comunicazione di dati utilizzato soprattutto in nautica e nella comunicazione di dati satellitari GPS. L'ente che gestisce e sviluppa il protocollo è la **National Marine Electronics Association**.

Il protocollo si basa sul principio che la fonte, il **talker**, può soltanto inviare i dati (*sentences*) e la ricevente, detta *listener*, può soltanto riceverli.

I record dei file scritti in formato NMEA sono generalmente strutturati nel modo seguente:

$PREFISSO,dato1,dato2,....,datoN-1,datoN*CHECKSUM

- **Prefisso**: la prima parte della stringa, che serve a specificare il tipo del *talker*, come, ad esempio, autopilota, dispositivo GPS, controllo velocità, controllo direzione. Nel caso dell'utilizzo di un dispositivo GPS, la prima parte del prefisso è **GP**, seguito dal tipo della frase. Tutte le frasi vengono identificate con tre lettere (*RMC,RMB,ecc*).
- **Checksum**: viene calcolato escludendo il carattere di inizio stringa e il carattere *****. L'algoritmo utilizzato è l'**exclusive OR 8 bit**, componendo il risultato in due lettere o numeri. La cifra più significativa tra le due sarà quella che verrà inviata per prima.

É interessante anche analizzare le sentences più interessanti:

- **$GPAAM - Waypoint Arrival Alarm**
- **$GPBOD - Bearing, Origin to Destination**
- **$GPBWW - Bearing, Waypoint to Waypoint**

- **$GPGGA - Global Positioning System Fix Data**
- **$GPGLL - Geographic Position, Latitude/Longitude**
- **$GPGSA - GPS DOP and Active Satellites**
- **$GPGST - GPS Pseudorange Noise Statistics**
- **$GPGSV - GPS Satellites in View**
- **$GPHDG - Heading, Deviation and Variation**
- **$GPHDT - Heading, True**
- **$GPRMB - Reccomended Minimum Navigation Information**
- **$GPRMC - Reccomended Minimum Specific GPS/TRANSIT Data**
- **$GPRTE - Routes**
- **$GPVTG - Track Made Good and Ground Speed**
- **$GPWCV - Waypoint Closure Velocity**
- **$GPWNC - Distance, Waypoint to Waypoint**
- **$GPWPL - Waypoint Location**
- **$GPXTE - Cross-Track Error, Measured**
- **$GPXTR - Cross-Track Error, Dead Reckoning**
- **$GPZDA - UTC Date/Time and Local Time Zone Offset**
- **$GPZFO - UTC and Time from Origin Waypoint**
- **$GPZTG - UTC and Time to Destination Waypoint**

Nel file creato dall'applicazione CoPilot sono state riscontrare 3 delle 22 sentences sopra descritte:

- **$GPGGA**

Name	**Example Data**	**Description**
Sentence Identifier	$GPGGA	Global Positioning System Fix Data
Time	170834	17:08:34 UTC
Latitude	4124.8963,N	41d 24.8963' N or 41d 24'54" N
Longitude	08151.6838,W	81d 51.6838' W or 81d 51'41" W
Fix quality: – 0 = Invalid – 1 = GPS fix – 2 = DGPS fix	1	Data is from GPS fix
Number of Satellities	05	5 Satellities are in view
Horizontal Dilution of Precision(HDOP)	1.5	Relative accuracy of horizontal position
Altitude	280.2,M	280.2 meters above mean sea level
Height of geoid above WGS84 ellipsoid	-34.0,M	-34.0 meters
Time since last DGPS update	blank	no last update
DGPS reference station id	blank	no station id
Checksum	*75	used by program to check for trasmission errors

- **$GPGSA**

$GPGSA,A,3,19,28,14,18,27,22,31,39,,,,,1.7,1.0,1.3*34

Field	**Description**
1	Mode: – M = Manual, forced to operate in 2D or 3D – A = Automatic, 3D / 2D
2	Mode: – 1 = Fix not available – 2 = 2D – 3 = 3D
3-14	PRN's of Satellite Vehicles (SV's) used in position fix (null for unused fields)
15	Position Dilution of Precision (PDOP)
16	Horizontal Dilution of Precision (HDOP)
17	Vertical Dilution of Precision (VDOP)

- **$GPRMC**

$GPRMC,hhmmss.ss,A,llll.ll,a,yyyyy.yy,a,x.x,x.x,ddmmyy,x.x,a,m*hh

Field	Description
1	UTC Time of fix
2	Data status (A = Valid Position, V = Navigation Reciver Warning)
3	Latitude of fix
4	N or S of longitude
5	Longitude of fix
6	E or W of longitude
7	Speed over ground in knots
8	Track made good in degrees True
9	UTC Date of fix
10	Magnetic Variation Degrees (Easterly var.subtracts from true course)
11	E or W of magnetic variation
12	Mode indicator, (A = Autonomous, D = Differential, E = Estimated, N = Data not Valid)
13	Checksum

Il primo passo è stato quello di modificare il file *.gps* relativo ad uno specifico percorso, inserendo date ed orari fittizi rispetto a quelli reali, in modo da alterare l'esatto momento in cui il percorso è stato effettuato.
Data la mole di dati presenti all'interno del file, e constatata l'impossibilità di apportare delle modifiche manuali, la soluzione più veloce ed efficiente per ap-

portare dei cambiamenti è stata quella di sviluppare un programma in linguaggio C che, presi in input 3 parametri:

- **path/nome_file**: il nome del file *.gps* da modificare
- **offset (in minuti)**: i minuti da aggiungere all'orario reale presente nel file.
- **data**: la nuova data di ogni record.

modifica automaticamente tutti i record del file in esame, cambiando, nello specifico, l'ora e la data di uno specifico percorso.
Terminata l'esecuzione del programma, il file conterrà il percorso reale, ma le date e gli orari associati saranno del tutto differenti da quelli veri: è stato quindi raggiunto un punto fondamentale per la creazione di un alibi convincente.
Constatato l'effettivo funzionamento di questa procedura, il passo successivo è stato quello più complicato: accedere e modificare uno specifico file comporta l'aggiornamento delle date di *accesso,modifica,cambiamento,creazione* di quel specifico file. Per rendere davvero credibile l'alibi, si è dovuto studiare un meccanismo tramite il quale fosse possibile modificare le date suddette, in modo da evitare discrepanze evidenti tra le informazioni.

5.3.3 Modifica delle tracce

Una volta ottenuta la lista di tutte le tracce lasciate dal navigatore durante un suo utilizzo, è stato realizzato il software che ne consente la modifica. In primo luogo abbiamo studiato come modificare il contenuto dei due file sopra citati. Il primo file analizzato è "/sdcard/Copilot/[MESE][GIORNO][ANNO].gps", il cui formato è descritto nel paragrafo 5.3.2, ed il tool sviluppato per modificarlo è "parser"; il secondo file è invece "/sdcard/Copilot/EU/Italy/save/[DATA]Trip.log"

e contiene un record per ogni tragitto percorso con il navigatore, ed il tool sviluppato per modificarlo è "trip".

Una volta modificate le informazioni sensibili all'interno dei file della cartella "/sdcard/Copilot/", la fase successiva di modifica delle tracce è stata la modifica dei timestamp dei file, per la quale abbiamo creato uno script bash, in grado di aggiornare i timestamp di tutti i file della cartella Copilot. Gli argomenti per eseguire lo script sono i seguenti:

1. Il percorso alla cartella Copilot
2. Il nuovo giorno del viaggio
3. Il nuovo mese del viaggio
4. Il nuovo anno del viaggio
5. Di quanti minuti si vuole traslare l'utilizzo del navigatore
6. Il nome del file "[MESE][GIORNO][ANNO].gps"
7. Il giorno in cui è stato effettuato il viaggio che si deve modificare

Dal momento che questo script, oltre a modificare i timestamp, avvia a sua volta anche i programmi C prima accennati con i parametri corretti, al termine della sua esecuzione avrà modificato tutte le tracce lasciate dal navigatore, rendendo dunque la creazione di un alibi estremamente semplice.
Per verificare l'efficienza delle modifiche effettuate é stato realizzato un test per controllare se, dopo aver modificato un file, vengano lasciate tracce riscontrabili dall'analisi fisica della sdcard. Il test ha dimostrato che la modifica di un file ne comporta l'effettiva sovrascrittura, quindi non é possibile trovare tracce della versione precedente del file.

Capitolo 6

Sviluppi Futuri

Il lavoro svolto ha portato alla luce diversi risultati; tuttavia ci sono ancora delle caratteristiche del sistema operativo che hanno bisogno di una migliore analisi. Per far luce sulle problematiche riscontrate, sono stati formulati una serie di casi di test.

Test Case 1

Il primo test case é atto ad evidenziare il comportamento del sistema operativo Android rispetto alla cancellazione logica dei file. A seguito degli esperimenti effettuati é stato evidenziato che dopo la cancellazione logica di un file, le celle non vengono immediatamente sovrascritte.
Quindi la domanda aperta é: dopo quanto tempo il SO sovrascrive le celle che ospitavano il file cancellato?
Passi da seguire:

1. Creazione di diversi tipi di file (testo, multimediali, etc.) di diversa taglia
   (a) n file di 1KB
   (b) n file di 10KB

(c) n file di 100KB

(d) n file di 1MB

Ovviamente tali file dovranno essere creati sulla memoria interna del terminale.

2. Effettuare un dump della memoria NAND del terminale.

3. Cancellare a livello logico i file creati al punto 1.

4. Effettuare dump ciclici, preferibilmente ogni ora.

5. Annotare dopo quanto tempo le pagine che prima contenevano i file al punto 1, vengono riallocate e/o sovrascritte.

Test Case 2

Cercare di decodificare i file contenenti le rotte del navigatore di Google Maps. Dagli studi effettuati risulta che i file contenenti le rotte siano serializzati, quindi sarebbe necessario conoscere la struttura dell'oggetto presente nel file.

Test Case 3

Replicare il primo test case utilizzando il software SHREDroid[1] per la cancellazione dei file.

[1]Il sito web di SHREDroid é http://www.securedeletion.com/

Bibliografia

[1] Scientific Working Group on Digital Evidence.

[2] National Marine Eletronics Association (NMEA) - http://www.nmea.org/.

[3] Joe Mehaffey and Jack Yeazel - GPS Informarion resource - http://www.gpsinformation.org/dale/nmea.htm.

[4] CoPilot Mobile Navigation - http://www.copilotlive.com/uk/.

[5] Sicurezza su Android, presentato il primo rootkit - http://www.opengeek.it/android/sicurezza-android-rootkit/.

[6] Dan Goodin - 99http://www.theregister.co.uk/2011/05/16/android_impersonation_attacks/, 2011.

[7] BusyBox - http://www.busybox.net/.

[8] Yaffs - http://www.yaffs.net/.

[9] Google Projects for Android - http://code.google.com/android/.

[10] Ben Elgin - Google buys Android for its mobile arsenal - http://www.businessweek.com/technology/content/aug2005/tc20050817_0949_tc024.htm, 2005.

[11] Android SDK - http://developer.android.com/sdk/index.html.

[12] Statistics AndroLib - http://www.androlib.com/appstats.aspx.

[13] Android website - http://www.android.com/about/.

[14] Android Developers - http://developer.android.com/guide/basics/what-is-android.html.

[15] G. ME, F. Pace, and A. D. Stefano, *Digital Investigation* **7** (2010).

[16] J. Lessard and G. C. Kessler, *SMALL SCALE DIGITAL DEVICE FORENSICS JOURNAL* **4** (2010).