Android: Google Mobile Platform (parte seconda)

DI ALBERTO NOSEDA, FABIO BERTONE, PAOLO PEREGO, MICHELE MILESI *

Google e OHA hanno posto un forte accento sulla possibilità di sviluppare le proprie applicazioni, tanto da rilasciare lo SDK prima dell'uscita ufficiale e organizzando l'Android Developer Challenge, una gara a livello mondiale per creare la migliore applicazione per Android che sappia sfruttarne a pieno le potenzialità.
Guardando le librerie fornite da Android, si nota immediatamente la differenza con Java Micro Edition. In JME esiste il concetto di profilo, che determina le librerie che ogni costruttore hardware mette a disposizione. Di fatto ogni dispositivo ha un proprio profilo e le librerie disponibili variano da modello a modello, rendendo difficile che un'applicazione complessa possa sicuramente funzionare su tutti i dispositivi.
Android, al contrario, definisce per tutti i dispositivi il medesimo insieme di librerie. Tra queste, alcune sono considerate opzionali, perché presenti ma non attive, se gestiscono componenti hardware che potrebbero non essere disponibili nei dispositivi. L'applicazione non si blocca perché non trova delle librerie, ma, al limite, non può utilizzare determinate funzionalità. Per lo sviluppatore diventa più semplice gestire le differenze di configurazione, dovendo trattare solamente l'evento che si riferisce al dispositivo non attivabile.
Android permette di gestire grafica 2D e 3D, tramite OpenGL; i principali formati audio/video; e la connettività da GSM a HSPA; include il motore SQLite, un relazionale standard SQL; le principali librerie di Java Standard Edition e le librerie HTTP di Apache.

Sviluppare per Android

La versione del linguaggio di riferimento è JSE 5, anche se considerazioni di performance sconsigliano di utilizzarne tutte le funzionalità più avanzate.
Una tipica applicazione è costituita da Activity e View che rappresentano, in un pattern MVC, il controller e la view. La definizione degli oggetti View avviene tramite file xml, evitando la parte tipicamente più noiosa dello sviluppo delle interfacce. Gli oggetti Intent permettono a un'applicazione di


TECNOLOGIE
Dal sogno alla realtà? In dieci passi ci si può provare
UP TO DATE
Android: Google Mobile Platform (parte seconda)


TECNOLOGIE

navigare tra le varie Activity; gli Intent sono assimilabili, tramite l'utilizzo dei Content Provider, ai Model in MVC. I Content Provider sono il metodo preferibile per accedere, in modo strutturato, ai dati di un'applicazione; e per condividere le informazioni tra le applicazioni. Esistono vari tipi di Content Provider standard per gestire i file multimediali, i contatti della rubrica, ecc...
Gli Intent permettono anche di attivare un'altra applicazione, a patto di averne i diritti. In questo caso l'applicazione di destinazione deve aver definito cosa pubblicare tramite degli Intent Receiver. In ultimo è possibile definire dei servizi, Service, che possono girare in background senza un'interfaccia grafica per, ad esempio, suonare un file MP3
Android SDK. Lo SDK di Android fornisce, oltre alle librerie, alcuni tool di supporto, la documentazione e vari esempi. La documentazione e gli esempi sono ottime basi per iniziare a sviluppare: esiste persino un tutorial che guida passo dopo passo nello sviluppo di un'applicazione. Il materiale diventa scarso, a volte fuorviante, quando si vuole iniziare a creare applicazioni di una certa complessità. In questo caso sia i blog che i gruppi di discussione forniti da Google, sono un'ottima fonte d'informazione. Il vero punto di forza dell'SDK sono i tool forniti. Per citarne qualcuno esiste un plugin di Eclipse che gestisce il manifest dell'applicazione, mappa tramite classi le risorse - immagini, testi, view - lancia il simulatore e ne permette il debug.

Il modello di sicurezza di Android

Nell'architettura di Android il kernel Linux riveste un ruolo importante anche per la sicurezza. Il modello di sicurezza alla base del file system è quello tipico di Unix. A ogni utente, e a ogni gruppo, è associato un codice univoco che ne identifica il ruolo all'interno del sistema e quindi i permessi associati. In Android l'utente è un'applicazione e non la classica connessione remota a una console.
All'installazione di un'applicazione, Android vi associa il codice identificativo con il quale applicare i meccanismi di protezione. Inoltre ogni applicazione è eseguita all'interno di una sandbox che, a differenza degli applet, è dedicata all'applicazione stessa.
Quindi due applicazioni differenti, chiamate anche package, non condividono lo stesso UID e sono eseguite in sandbox

differenti. Di fatto non possono condividere alcun tipo di risorsa. È comunque possibile nel *Manifest* file delle due applicazioni specificare, attraverso l'attributo shareUserId, che si vuole che queste condividano lo UID, in questo caso si deve accettare che le due applicazioni siano trattate come una sola entità.

In fase d'installazione, il package propone all'utente le autorizzazioni sul sistema di cui ha bisogno (accesso alla rubrica, al microfono, al GPS, ...). L'utente può decidere quindi con quali risorse il package deve interagire; nessuna autorizzazione è ereditata dal sistema e solo l'utente decide quali concedere. Se l'utente nega un'autorizzazione, il package è installato ugualmente, ma non potrà accedere alla risorsa.

In termini di sicurezza, la principale differenza con altre piattaforme mobili risiede nella sandbox e nella protezione che essa garantisce. Al contrario dell'iPhone, dove le applicazioni sono eseguite con i privilegi di root, all'interno della versione ridotta del sistema operativo Mac OS X in esecuzione sul terminale.

Android Code Review. Non essendo ancora disponibile Il codice sorgente dell'SDK una valutazione sull'effettiva sicurezza del codice può essere fatta unicamente seguendo un approccio blackbox. Compiendo una code review con Findbug si ottengono comunque dei risultati interessanti. In primo luogo sembrano esserci parti di codice vulnerabili a una SQL injection perché sono costruite delle query concatenando stringhe non costanti. L'impatto sul sistema è mitigato dal meccanismo di sicurezza già descritto in precedenza e dall'utilizzo di SQLite. Esiste comunque la possibilità da parte di un utente di corrompere i dati di una singola applicazione.

All'interno del codice ci sono inoltre numerose violazioni delle best practices di safe coding, in particolare quelle riguardanti l'utilizzo corretto degli operatori di confronto tra Stringhe, la dereferenziazione di puntatori e molte altre anomalie che possono portare al blocco dell'applicazione con conseguente DoS. Si consideri che quest'analisi può soffrire di falsi positivi dovuti alla mancanza del codice sorgente da utilizzare come controprova per le vulnerabilità rilevate.

In ogni caso, dal punto di vista della sicurezza, Android sembra essere robusto. Si pensi solo al fatto che non è possibile per un'applicazione accedere direttamente al kernel linux sottostante e che le librerie fornite - SQLite, Apache HTTP, JDBC, javax.crypto... - permettono di scrivere codice che rispetti le best practices di safe coding

** Gli autori sono consulenti, rispettivamente, in Technology Reply, Atlas Reply, Spike Reply e @logistics Reply*