

Hacia la construcción de una herramienta para la creación de bitácoras para Android

Bárbara Cervantes¹, Miguel González-Mendoza¹, Raúl Monroy¹, Christian Eduardo Galdámez Blanco², Eduardo Ismael García Pérez²

¹ Tecnológico de Monterrey, Campus Estado de México, México

² Universidad Politécnica de Chiapas, México

{bcervantesg,mgonza,raulm}@itesm.mx
{christianeduardogb,eduardo78d}@gmail.com
<http://www.itesm.mx>
<http://www.upchiapas.edu.mx>

Resumen. El uso de teléfonos inteligentes se ha vuelto muy popular. La interacción de los usuarios con estos dispositivos genera información que describe su comportamiento en el sistema. Sistemas operativos como Android [17], ofrecen la posibilidad de registrar muchos aspectos de esta interacción, sin embargo, esta información generalmente es utilizada en el momento en el que sucede la interacción sin que se cree un registro persistente de todo lo ocurrido. El acceso a esta información puede resultar muy útil para la caracterización del usuario, un área con múltiples aplicaciones. En este documento se presenta el avance realizado hacia la construcción de una herramienta que permita la colección de información del uso de un dispositivo móvil para su posterior análisis y aprovechamiento.

Palabras clave: interacción humano-computadora, comportamiento del usuario, caracterización del usuario, Android.

1. Introducción

Avances en la tecnología nos han llevado a un mundo en el que la adopción de los dispositivos móviles se ha expandido considerablemente. A diferencia de hace diez años, cuando los teléfonos móviles únicamente contenían el historial de llamadas, algunos contactos y mensajes SMS; ahora, cada dispositivo contiene información acerca de la vida diaria de la persona. El surgimiento de aplicaciones de índoles muy diversos (email, redes sociales, entretenimiento, productividad, salud, etc.) ha potenciado el uso de teléfonos inteligentes y consecuentemente se ha incrementado la cantidad de información generada dentro de los mismos. Cada uno de estos dispositivos está fuertemente ligado a la identidad de la persona y, además, contiene información personal que puede llegar a ser sensible por lo que se debe tener presente la seguridad de esta información.

Cuando un usuario interactúa con un sistema computacional, en este caso móvil, genera información que describe su comportamiento en el sistema. Esta

información puede ser extraída y representada en un modelo a través del cual se pueda reconocer al usuario. Nuestro trabajo se centra en recopilar la información generada por el usuario para su posterior análisis y aprovechamiento. En particular, para este primer avance, se lleva un registro de las tareas que se ejecutan en un dispositivo Android; estas tareas (definidas por el sistema operativo) describen la forma en que se navega a través del dispositivo y las aplicaciones, por lo que las consideramos parte importante de la interacción entre el usuario y el dispositivo móvil.

Existen muchas posibles aplicaciones que se benefician o se basan en conocer el comportamiento de los usuarios. Entre las áreas de aplicación tradicionales de modelado de usuario se encuentran la personalización de servicios [14], los sistemas de recomendación [11,8,14], la mejora de calidad de servicios [12], etc. Áreas de aplicación más recientes y más orientadas a dispositivos móviles incluyen el monitoreo de la salud [13], caracterización de actividades [7,10] y mejoras en seguridad [6,9,16,15], entre otras. Un enfoque es la creación de un modelo de usuario que describa las características relevantes para el problema en cuestión [6,8,9]; otro es la abstracción de un modelo de comportamiento general a partir del cuál se trata de inferir el comportamiento de un usuario en específico [12,10].

En este documento se reporta el avance realizado hacia la construcción de una herramienta que permita la colección de información del uso de un dispositivo móvil para su posterior análisis. Para esto comenzamos por dar una visión general del proceso de creación de bitácoras, seguido del caso específico de sistemas móviles (Sección 2). Posteriormente, en la sección de Arquitectura (Sección 3), describimos los componentes principales de la herramienta y cómo se relacionan, se mencionan las decisiones de diseño tomadas y algunas pruebas realizadas. Lo que nos lleva a presentar el estado en el que se encuentra la herramienta y el trabajo que se realizará en el marco de este proyecto (Sección 4). Finalmente viene la sección de Conclusiones (Sección 5), en la que incluimos nuestros comentarios finales.

2. Trabajo relacionado

Una bitácora (o historial) es un archivo en el que se registran los eventos que ocurren en un sistema operativo, aplicación o algún proceso que se quiera monitorear o respaldar. Este tipo de archivos son generalmente utilizados para entender las actividades del sistema. En el caso de los sistemas operativos para servidores o computadoras personales, existen herramientas que permiten activar el registro de eventos en una bitácora; por ejemplo: *Linux Audit framework* [18] es una herramienta que permite registrar desde la ejecución de archivos hasta las llamadas a sistema. El módulo del kernel *audit* intercepta las llamadas a sistema y guarda los eventos relevantes; el demonio *auditd* escribe estos registros en el disco y varias herramientas de línea de comando (*autrace*, *ausearch* y *aureport*) sirven para visualizar los eventos registrados en la bitácora (ver Figura 1).

| Event Summary Report | |
|----------------------|---------------|
| total | type |
| 2434 | SYSCALL |
| 816 | USER_START |
| 816 | USER_ACCT |
| 814 | CRED_ACQ |
| 810 | LOGIN |
| 806 | CRED_DISP |
| 779 | USER_END |
| 99 | CONFIG_CHANGE |
| 52 | USER_LOGIN |

Fig. 1. Ejemplo de reporte de una bitácora, usando *aureport* (tomado de [18]).

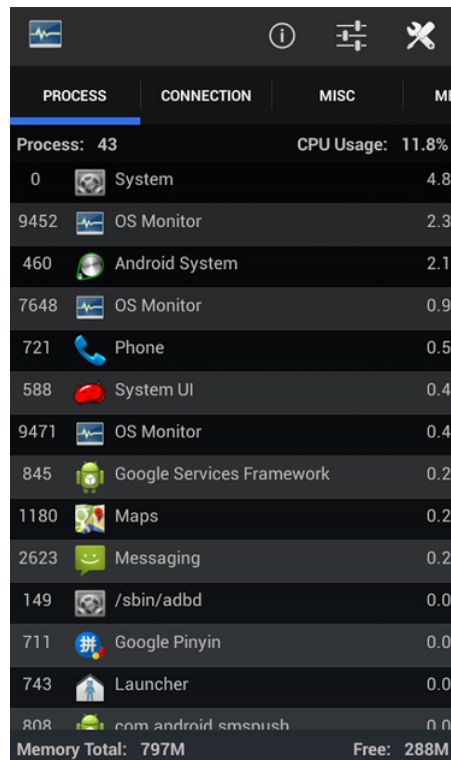
En cambio, cuando hablamos de dispositivos móviles, la mayoría de las herramientas que permiten ver los eventos que ocurren en el sistema están enfocadas a la etapa de desarrollo de una aplicación, cuando se activa el modo de depuración, o a ver el estatus del dispositivo en el tiempo actual. Esto se debe en parte a que el proceso de colección de datos para una bitácora viene a costa del rendimiento. Los dispositivos móviles son equipos más limitados en este sentido (al tener un espacio de almacenamiento mucho menor que el de una PC y mayores restricciones de energía) por lo que es imperativo que las herramientas de colección de datos sean optimizadas para no afectar el funcionamiento y experiencia de uso del móvil.

La mayoría de las aplicaciones disponibles para observar las tareas que se ejecutan en un sistema Android, simplemente consisten en una interfaz gráfica que muestra la información del momento actual al usuario, sin embargo, no existe la función del historial, es decir no hay un registro como tal (ver Figura 2).

También existen sistemas más completos que atacan un problema en específico, por ejemplo, *TaintDroid* [4] es una extensión a Android con el objetivo de comprender el flujo de los datos para identificar aplicaciones que traten la información sensible de manera inapropiada. Otro ejemplo, dedicado a la creación de bitácoras se expone en [5], en este caso el problema que se abarca es la detección de fallas en el sistema.

El sistema Android cuenta con una herramienta de registro de mensajes *LogCat* [21], sin embargo, como se mencionó anteriormente, está pensada más bien con fines de depuración durante el desarrollo de la aplicación. La inclusión de un nuevo registro se programa dentro de cada aplicación por lo que cada mensaje tiene la estructura que el programador decida, lo cual dificulta la interpretación de cada uno de estos mensajes. El tipo y la cantidad de mensajes en LogCat son muchos y normalmente se observan en el IDE del desarrollador a través de una conexión con cable USB, en situaciones normales, es decir, cuando el teléfono es utilizado por el usuario final, debido a las limitaciones de espacio y procesamiento del teléfono móvil la bitácora se sobrescribe. Crear una bitácora a partir de lo registrado con LogCat no resulta práctico debido a que es difícil determinar el momento en el que se sobrescribirá el archivo y además habría que

estar filtrando este archivo para obtener únicamente los eventos relevantes, por esta razón vemos la necesidad de crear una herramienta que registre los eventos que nos interesan, como es el caso de [5].



The screenshot shows a mobile application interface with a dark theme. At the top, there are icons for a signal strength indicator, an information icon, a filter icon, and a close icon. Below these is a header with four tabs: 'PROCESS', 'CONNECTION', 'MISC', and 'MEM'. The 'PROCESS' tab is selected. The main content area displays a list of processes with their IDs, names, and CPU usage percentages. At the bottom, there are summary statistics for memory usage.

| Process ID | Process Name | CPU Usage |
|------------|---------------------------|-----------|
| 0 | System | 4.8 |
| 9452 | OS Monitor | 2.3 |
| 460 | Android System | 2.1 |
| 7648 | OS Monitor | 0.9 |
| 721 | Phone | 0.5 |
| 588 | System UI | 0.4 |
| 9471 | OS Monitor | 0.4 |
| 845 | Google Services Framework | 0.2 |
| 1180 | Maps | 0.2 |
| 2623 | Messaging | 0.2 |
| 149 | /sbin/adbd | 0.0 |
| 711 | Google Pinyin | 0.0 |
| 743 | Launcher | 0.0 |
| 808 | com.android.smsush | 0.0 |

Process: 43 CPU Usage: 11.8%

Memory Total: 797M Free: 288M

Fig. 2. Ejemplo de una aplicación que permite ver el estado del dispositivo, OS Monitor [20], se muestran los procesos que corren actualmente y detalles del uso de CPU y la memoria, sin embargo, no hay un registro persistente de estos datos.

3. Arquitectura

La herramienta tiene como objetivo principal registrar los eventos que describen la interacción del usuario con el dispositivo móvil. Entre estos eventos están las interacciones con la pantalla, lecturas de sensores como el acelerómetro o giroscopio y las aplicaciones ejecutadas. Para la primera etapa del proyecto se decidió incluir en la bitácora las tareas y procesos que se ejecutan en un dispositivo Android; éstos describen la forma en que se navega a través del

dispositivo y las aplicaciones, por lo que las consideramos parte importante de la interacción entre el usuario y el dispositivo móvil, y creemos que serán un buen punto de partida para nuestra investigación. Además tienen la ventaja de poder ser obtenidas a bajo costo y estar siempre presentes, es decir, no dependen de la conectividad de la red o de que suceda un evento en específico y no requieren prender ningún sensor.

En esta Sección se describen los componentes principales de la herramienta y cómo se relacionan. Empezamos por mencionar algunas decisiones de diseño y posteriormente describimos cada elemento de la arquitectura.

3.1. Antecedentes

En esta Sección se describen algunas de las generalidades de Android y se mencionan las decisiones de diseño derivadas de ellas:

Almacenamiento

Existen dos alternativas en cuanto a dónde almacenar la bitácora: usar el almacenamiento externo del dispositivo, comúnmente una tarjeta SD, o usar el almacenamiento interno, menor en tamaño pero siempre disponible. Descartamos la posibilidad de utilizar el almacenamiento externo porque a pesar de que se tendría menor restricción en cuanto a espacio de almacenamiento, esta memoria puede ser leída y modificada por el usuario y por otras aplicaciones. Se eligió guardar la bitácora en el almacenamiento interno del dispositivo ya que un archivo almacenado en la memoria interna del dispositivo solamente puede ser accedido por la aplicación a la que pertenece, para otras aplicaciones e incluso para el dueño del dispositivo está restringido el acceso de estos archivos

En cuanto al formato, se eligió almacenar los registros en una base de datos. La razón principal es nuevamente que una base de datos no será accesible para el usuario u otras aplicaciones. Android provee un soporte de SQLite, [22]. SQLite permite a cualquier aplicación nativa de Android el acceso al almacenamiento en bases de datos, se provee una forma fácil de hacer la conexión y operaciones simples como consultas, inserciones, actualizaciones y borrado de registros; así como operaciones más complejas como uniones, pivotes y otras más. Cada uno de los registros serán almacenados en una base de datos que estará en constante actividad ya que el recabado de registros será constante y por lo consiguiente la inserción también.

Ejecución

Se decidió implementar la herramienta como un servicio del sistema operativo. Un servicio (*android.app.Service* [19]) es un componente de una aplicación que puede realizar operaciones de larga duración en segundo plano y que puede tener, o no, una interfaz de usuario. Un servicio puede continuar ejecutándose en segundo plano incluso si el usuario cambia a otra aplicación.

3.2. Componentes

La herramienta tiene dos componentes principales: un colector y un servidor. El colector reside en el dispositivo móvil y es el encargado de registrar los eventos en la bitácora, la cual se transfiere periódicamente al servidor en donde se almacena permanentemente. A continuación se describen los detalles de implementación de cada uno de los componentes.

Colector

El Colector es quien se encarga de insertar en la base de datos las nuevas tareas (*Android tasks*). Para implementarlo se siguió el paradigma de Modelo-Vista-Controlador (MVC) [2], cada uno de los elementos se encuentra en un paquete diferente. Se siguió este paradigma para tener un diseño modular y que el hecho de registrar un nuevo tipo de evento en la bitácora no involucre modificar toda la arquitectura.

En el primer paquete, llamado «Controllers», se encuentran los controladores; dentro de él tenemos el mayor número de clases las cuales permiten recabar todas las tareas que el dispositivo genere. Es aquí donde se define qué eventos deben ser monitoreados. Como se ha mencionado anteriormente, por el momento obtenemos los procesos y tareas del sistema operativo; esto se logra a través de la clase *ActivityManager* [23]. Este paquete también se encarga de iniciar la transferencia de la bitácora al servidor y de tomar las medidas adecuadas (iniciar o detener el registro de eventos) cuando el dispositivo cambia de estado.

En el paquete «Models» podemos encontrar las clases que crean y manipulan la base de datos. Aquí es donde se definen los modelos de los eventos por obtener: tenemos el caso de *AndroidProcess* y *AndroidTask* para los que se obtienen el nombre de la tarea, identificadores y la fecha (Figura 3). Cada evento es un registro en la base de datos, los cuales se añaden secuencialmente.

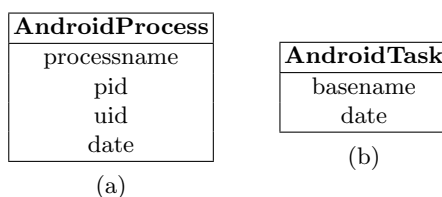


Fig. 3. Cada evento tiene diferentes características a registrar, en el modelo del evento se definen estas características. En la imagen se observa el modelo para los procesos(a) y tareas(b) de Android.

Por último el paquete «Views» contiene las clases necesarias para el funcionamiento de la interfaz gráfica. Aunque la interacción con el usuario es mínima,

existe una interfaz gráfica que permite al usuario confirmar si se llevará a cabo la transmisión de la bitácora al servidor (para evitar realizarla en un momento no deseado) y elegir el tipo de eventos a registrar.

A grandes rasgos el funcionamiento del Colector es el siguiente: al iniciar el dispositivo se crea un thread de servicio que estará monitoreando las tareas del sistema operativo; en el momento en que se detecta una tarea nueva se crea el modelo que la representa y se añade a la base de datos. El colector también está pendiente de los cambios de estado del dispositivo, es decir, cuando pasa a un estado suspendido, cuando sale de este mismo estado, cuando se reinicia, etc., y realiza las acciones pertinentes para que la colección siga activa sin necesidad de interacción por parte del usuario del dispositivo. Existe la posibilidad de detener el proceso de registro si el usuario así lo desea y de elegir el tipo de información que se registra. Por otro lado tiene una alarma programada para que cada día se intente hacer la transmisión al servidor, esta alarma se activa a la hora especificada y pregunta al usuario si desea hacer la transmisión en ese momento, en caso de decidir no hacerlo se enviará un recordatorio a los 30 minutos.

Servidor

El servidor recibe los archivos del historial para realizar un respaldo, el objetivo es que se pueda liberar ese espacio del dispositivo móvil y que la bitácora quede almacenada en un servidor seguro para su posterior análisis.

Para crear el servidor se utilizó el lenguaje de programación Java debido a que se desean heredar las características de seguridad que ofrece. El servidor es de tipo *socket* (*ServerSocket*), el cual está diseñado para estar siempre a la escucha de peticiones de los clientes para realizar el respaldo. Se define un puerto que es el que escuchará las peticiones. El servidor tiene la capacidad de crear subprocesos concurrentes para poder realizar el respaldo de más de una bitácora al mismo tiempo; con ayuda de la clase *Thread* es posible crear n número de subprocesos dentro del servidor los cuales cobran vida cada vez que una petición es aceptada por el servidor, de esta forma se evita la sobrecarga del servidor.

3.3. Seguridad

Debido a que la información que se maneja en las bitácoras es información sensible, buscamos que al hacer la transmisión la información no se vea comprometida. Para esto se establece una conexión entre el dispositivo y el servidor siguiendo el protocolo de Diffie-Hellman[1].

Otra de las medidas tomadas para asegurar la transmisión de una bitácora del dispositivo al servidor es que se realice en forma cifrada. Java en conjunto con el paquete de seguridad «*security*» hace uso del algoritmo AES[3] para el cifrado de los datos que el cliente, en este caso el dispositivo móvil, esté enviando por medio de la red. Uno de los puntos a tomar en cuenta dentro de este algoritmo

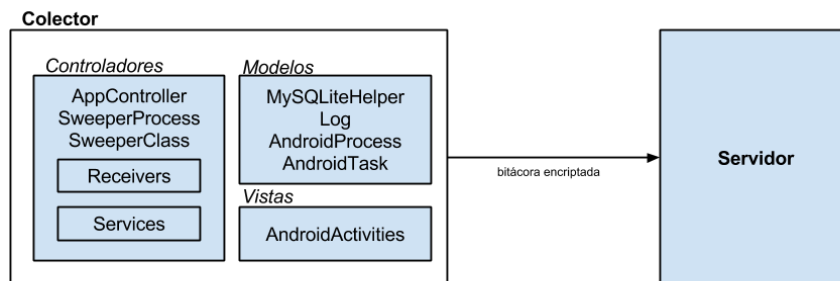


Fig. 4. Arquitectura del sistema

es que ya que es un algoritmo que se centra en el cifrado por bloques, contiene un límite de bits, en este caso se trabaja con 16 bits. Los datos se envían por partes y cifrados hacia el servidor el cual una vez aceptando la petición del cliente comienza a recibir los bloques de 16 bits desde el cliente y descifra estos datos. Una vez descifrados los datos pasan a ser concatenados en el servidor para que cuando la transmisión de datos se haya completado el archivo quede almacenado sin ningún problema y sin ningún byte perdido en la transferencia de los datos por la red.

3.4. Pruebas

Hasta el momento se han realizado pruebas en dos teléfonos móviles. El objetivo de estas primeras pruebas era comprobar que el hecho de tener la herramienta corriendo no afecta la experiencia de usuario (desempeño) del dispositivo y medir el espacio que las bitácoras ocupan en el dispositivo, ya que al estar almacenadas en la memoria interna si fueran de gran tamaño causarían inconveniente al usuario. Después de instalar la herramienta y utilizar el celular de manera cotidiana los dos usuarios reportaron no haber percibido diferencia en el desempeño del dispositivo, el sistema corría a la misma velocidad y el consumo de batería no fue notorio, y después de un día de interacción el tamaño de la bitácora no superó los 64 kB. En pruebas posteriores es nuestra intención formalizar estas percepciones, es decir, determinar exactamente cuál es el porcentaje de energía que se dedica a la herramienta y cómo varía el tamaño de la bitácora dependiendo del usuario y los eventos registrados.

4. Estado actual y trabajo futuro

Hasta el momento la herramienta tiene la opción de registrar dos tipos de eventos: procesos y tareas del sistema operativo. Se busca extenderla para registrar otros tipos de eventos, como ejemplo tenemos el uso de sensores del

propio dispositivo (*i.e.* giroscopio, GPS, etc.) así como de interfaz con el usuario (*i.e.* touch screen).

Se busca desplegar el servidor y reclutar a un grupo piloto de usuarios para la creación de una base de datos de interacción con teléfonos móviles. Una vez conformada, se pretende dar un enfoque hacia la identificación de los niveles de interacción entre el dispositivo y el usuario, enfocado hacia la detección de anomalías. Se evaluará el uso de estos dos tipos de eventos en la autenticación implícita[9], se espera que los resultados sean positivos bajo la hipótesis de que éstos son capaces de capturar la interacción que distingue a un usuario de otro. Además, buscamos considerar que la manera en que interactúa el usuario depende del nivel de carga mental que tiene en ese momento, es decir, la interacción no será la misma cuando el usuario presta 100 % de su atención al dispositivo que cuando el usuario está caminando o realizando alguna otra actividad en paralelo al utilizar el dispositivo móvil, probablemente en este último escenario el tiempo entre las tareas que ejecuta el usuario sea mayor. Esto se deberá de ver reflejado en el modelo del comportamiento de cada usuario.

Posteriormente, se extendería la herramienta para que esta colección de bitácoras no solamente sirva para crear una base de datos sino que además tenga como objetivo monitorear en tiempo real el uso del dispositivo para identificar cuando el usuario salga de su modelo de comportamiento.

5. Conclusiones

La versión aquí presentada de la herramienta para el registro de bitácoras en un sistema operativo para dispositivos móviles (Android), permite el registro de las tareas y los procesos del sistema operativo que el usuario ejecuta deliberadamente. Posteriormente se registrarán también eventos resultantes del uso de sensores del dispositivo.

Esta herramienta nos ayuda a sobrepasar el obstáculo tecnológico que implica la colección de las bitácoras para poder enfocarse en las aplicaciones que nacen del análisis del comportamiento de los usuarios.

De igual forma, amplía el horizonte de estudio para adaptar a este paradigma en móviles (dadas las restricciones de capacidad de cómputo y energéticas), estudios de análisis forense y seguridad de entornos de cómputo tradicionales (con mayor capacidad de procesamiento, almacenaje y sin restricciones energéticas importantes).

Referencias

1. Diffie, W., Hellman, M. E.: New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6), 644–654 (1976)
2. Krasner, G. E., Pope, S. T.: A Description of the Model-View-Controller User Interface Paradigm in the Smalltalk-80 System. (1988)
3. Daemen, J., Rijmen, V.: The design of Rijndael: AES-the advanced encryption standard. Springer Science & Business Media (2002)

4. Enck, W., Gilbert, P., Chun, B.-G., Cox, L. P., Jung, J., McDaniel, P., Sheth, A. N.: TaintDroid: An Information Flow Tracking System for Real-time Privacy Monitoring on Smartphones. *Commun. ACM*, 57(3), 99106 (2014)
5. Cinque, M., Cotroneo, D., Testa, A.: A Logging Framework for the On-line Failure Analysis of Android Smart Phones. In: *Proceedings of the 1st European Workshop on AppRoaches to MObiquiTous Resilience*, Vol. 1, pp. 2:12:6, ACM (2012)
6. Schmidt, A. D., Peters, F., Lamour, F., Scheel, C., Çamtepe, S. A., Albayrak, Ş: Monitoring Smartphones for Anomaly Detection. *Mobile Networks and Applications*, 14(1), 92106 (2009)
7. Cao, H., Bao, T., Yang, Q., Chen, E., Tian, J.: An Effective Approach for Mining Mobile User Habits. In: *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pp. 16771680, New York, NY, USA: ACM (2010)
8. Woerndl, W., Schueller, C., Wojtech, R.: A Hybrid Recommender System for Context-aware Recommendations of Mobile Applications. In: *Data Engineering Workshop, 2007 IEEE 23rd International Conference on*, pp. 871878 (2007)
9. Shi, E., Niu, Y., Jakobsson, M., Chow, R.: Implicit authentication through learning user behavior. *Lecture Notes in Computer Science*, vol. 6531, pp. 99113, Springer-Verlag Berlin Heidelberg (2011)
10. Furletti, B., Gabrielli, L., Renso, C., Rinzivillo, S.: Identifying Users Profiles from Mobile Calls Habits. In: *Proceedings of the ACM SIGKDD International Workshop on Urban Computing*, pp. 1724, New York, NY, USA: ACM (2012)
11. Park, S., Kang, S., Kim, Y.-K.: A channel recommendation system in mobile environment. *IEEE Transactions on Consumer Electronics*, 52(1), 3339 (2006)
12. Wu, S., Fan, H.-H.: Activity-Based Proactive Data Management in Mobile Environments. *IEEE Transactions on Mobile Computing*, 9(3), 390404 (2010)
13. Chan, V., Ray, P., Parameswaran, N.: Mobile e-Health monitoring: an agent-based approach. *IET Communications*, 2(2), 223230 (2008)
14. Gavalas, D., Kenteris, M.: A web-based pervasive recommendation system for mobile tourist guides. *Personal and Ubiquitous Computing*, 15(7), 759770 (2011)
15. Frank, M., Biedert, R., Ma, E., Martinovic, I., Song, D.: Touchalytics: On the Applicability of Touchscreen Input as a Behavioral Biometric for Continuous Authentication. *IEEE Transactions on Information Forensics and Security*, 8(1), 136148 (2013)
16. Feng, T., Yang, J., Yan, Z., Munguia Tapia, E., Shi, W.: TIPS: Context-aware Implicit User Identification Using Touch Screen in Uncontrolled Environments. In: *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications (HotMobile14)*, pp. 9:19:6, ACM (2014)
17. Google, Android, <http://www.android.com/>
18. The Linux Audit Framework, https://www.suse.com/documentation/sled10/pdf/doc/audit_sp1/audit_sp1.pdf
19. Google, Android Service, <http://developer.android.com/reference/android/app/Service.html>
20. OS Monitor - Android Apps on Google Play, <https://play.google.com/store/apps/details?id=com.eolwral.osmonitor>
21. Google, LogCat, <http://developer.android.com/tools/help/logcat.html>
22. SQLite, <https://www.sqlite.org/>
23. Google, Activity Manager, <http://developer.android.com/reference/android/app/ActivityManager.html>