PEDECIBA Informática
Instituto de Computación – Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay

Reporte Técnico RT 08-08

Hacia una Especificación Formal del Modelo de Seguridad de MIDP 3.0

Gustavo Mazeikis Gustavo Betarte Carlos Luna

2008

Hacia una Especificación Formal del Modelo de Seguridad de MIDP 3.0
Gustavo Mazeikis; Betarte, Gustavo; Luna, Carlos
ISSN 0797-6410
Reporte Técnico RT 08-08
PEDECIBA
Instituto de Computación – Facultad de Ingeniería
Universidad de la República

Montevideo, Uruguay, 2008

Hacia una Especificación Formal del Modelo de Seguridad de MIDP 3.0

Gustavo Mazeikis, Gustavo Betarte y Carlos Luna

Instituto de Computación, Facultad de Ingeniería, Universidad de la República, Uruguay
mazeikis@adinet.com.uy, {gustun, cluna}@fing.edu.uy

Resumen. En la Plataforma Java Micro Edition, el Perfil para Dispositivos de Información Móviles (MIDP) provee el ambiente de ejecución estándar para teléfonos móviles y asistentes de datos personales. La tercera versión del perfil, de reciente publicación, introduce una nueva dimensión en el modelo de seguridad de MIDP: la seguridad a nivel de aplicación. Para la segunda versión de MIDP, Zanella, Betarte y Luna proponen una especificación formal del modelo de seguridad en el Cálculo de Construcciones Inductivas, usando el asistente de pruebas Coq. Este artículo presenta una extensión de esta especificación, para incorporar los cambios planteados por la tercera versión de MIDP. La extensión planteada conserva las propiedades de seguridad demostradas en el modelo anterior, y permite seguir razonando sobre nuevas propiedades de seguridad.

Palabras Claves: Seguridad, Especificaciones Formales, MIDP, Coq.

1 Introducción

En los últimos años el uso de dispositivos portátiles, tales como teléfonos celulares y asistentes de datos, se ha popularizado a nivel mundial. Este tipo de dispositivos tienen fines y características que son en esencia diferentes de los que tienen, por ejemplo, las computadoras portátiles o de escritorio.

En la actualidad, la gran mayoría de las plataformas tecnológicas para los dispositivos portátiles incorporan la tecnología Java para sistemas embebidos – *embedded systems* – denominada *Java Micro Edition* (JME) [1]. La característica fundamental de esta edición, al igual que toda tecnología Java, es el uso de una máquina virtual, especialmente diseñada y adaptada para aprovechar al máximo los escasos recursos con los que cuentan los dispositivos portátiles. La tecnología JME proporciona mecanismos integrales para garantizar propiedades de seguridad de los dispositivos; en particular, para dispositivos móviles define un modelo de seguridad que restringe el acceso de las aplicaciones a funciones consideradas potencialmente peligrosas.

JME plantea una arquitectura basada en dos capas por encima de la máquina virtual: la capa configuración y la capa perfil. La configuración *Connected Limited Device Configuration* (CLDC) es un conjunto mínimo de bibliotecas de clases para ser utilizadas en dispositivos con procesadores poco potentes, memoria de trabajo limitada y capacidad para establecer comunicaciones de bajo ancho de banda. Esta configuración se complementa con el perfil *Mobile Information Device Profile* (MIDP) para

obtener un entorno de ejecución adaptado a dispositivos portátiles como teléfonos celulares, asistentes de datos personales y localizadores. El perfil define, entre otras cosas, un modelo de seguridad para aplicaciones, el ciclo de vida de las mismas, los mecanismos de comunicación de red, y una biblioteca para el acceso a funciones propias de la clase de dispositivos para la que fue concebido.

En la primera versión de MIDP (versión 1.0) [2] se plantea, y en la segunda versión de MIDP (versión 2.0) [3] se refina, un modelo de seguridad a nivel de plataforma basado en conjuntos de permisos, para acceder a las funciones del dispositivo. En la tercera versión (versión 3.0) [4] se introduce una nueva dimensión al modelo: la seguridad a nivel de aplicación, basada en autorizaciones, para que una aplicación pueda acceder a los recursos que comparte otra aplicación.

Para MIDP 2.0 Zanella, Betarte y Luna proponen en [5, 6] una especificación formal del modelo de seguridad en el Cálculo de Construcciones Inductivas (CIC) [7], usando el asistente de pruebas Coq [8]. Esta especificación es una base formal para la verificación del modelo de seguridad y la comprensión de su complejidad.

En este documento se describe informalmente el modelo de seguridad de MIDP a través de sus distintas versiones. Se propone una extensión conservativa de la formalización del modelo de seguridad de MIDP 2.0, con el objetivo de actualizar dicha herramienta y continuar verificando propiedades relevantes de seguridad del modelo. La extensión de la formalización también está implementada usando el asistente de pruebas Coq.

El artículo está organizado como sigue. En la sección 2 se describe el modelo de seguridad de MIDP. En la sección 3 se resume la especificación formal del modelo de seguridad de MIDP 2.0. La sección 4 plantea extensiones a la especificación formal que reflejan los cambios introducidos por MIDP 3.0 y, finalmente, la sección 5 presenta las conclusiones y los trabajos futuros.

2 El Modelo de Seguridad de MIDP

El modelo de seguridad de MIDP ha evolucionado a través de las distintas versiones del perfil. En las primeras dos versiones el principal objetivo fue la protección de funciones sensibles del dispositivo, mientras que en la tercera versión se busca proteger los recursos de una aplicación que pueden compartirse con otras aplicaciones. Estos dos niveles de seguridad se describen a continuación.

2.1 Seguridad a Nivel de Plataforma

En MIDP 1.0 toda aplicación que no haya sido preinstalada en el dispositivo (por ejemplo, por el fabricante), se ejecuta en un ambiente controlado denominado *sandbox* que prohíbe el acceso a aquellas funciones del dispositivo que se consideran potencialmente peligrosas.

En MIDP 2.0 se abandona el modelo sandbox en favor de un modelo menos restrictivo, basado en el concepto de *dominio de protección*. En los dispositivos compatibles con MIDP 2.0 cada aplicación instalada está asociada con un único dominio de protección. La política de seguridad es una matriz de control de acceso, donde los su-

jetos son las aplicaciones asociadas a cada dominio de protección, y los objetos son las funciones que deben protegerse.

Cada dominio de protección especifica un conjunto de permisos junto con la forma en la que los mismos pueden ser otorgados a las aplicaciones asociadas. Un dominio de protección consiste de: un conjunto de permisos que son otorgados incondicionalmente, sin intervención alguna del usuario; un conjunto de permisos que requieren autorización previa del usuario para ser otorgados, cada uno con un modo máximo que indica la forma y frecuencia en la que se requerirá la autorización. Los modos son: *blanket*: el permiso otorgado vale hasta que la aplicación sea desinstalada; *session*: el permiso otorgado vale hasta que finalice la ejecución de la aplicación; y *oneshot*: el permiso otorgado vale por única vez.

El conjunto de permisos efectivamente otorgados a una aplicación se determina en base a su dominio de protección, los permisos solicitados por la aplicación y los permisos otorgados por el usuario. Una aplicación posee un determinado permiso cuando lo ha solicitado en su descriptor y/o manifiesto, y además su dominio de protección otorga el permiso incondicionalmente, o su dominio especifica un modo de interacción con el usuario para el permiso, y el usuario ha dado una autorización que continúa siendo válida.

El modelo establece dos clases de aplicaciones. Una aplicación no confiable es aquella para la que no se ha podido determinar el origen y la integridad. Toda aplicación no confiable, así como una desarrollada sobre MIDP 1.0, se asocia a un dominio de protección que provee un entorno de ejecución restringido. Una aplicación confiable es aquella para la que se ha podido determinar su origen e integridad. Cuando se verifica el origen de la aplicación, se determina también el dominio de protección al que se asocia.

2.2 Seguridad a Nivel de Aplicación

En MIDP 3.0 se permite que aplicaciones de diferentes dominios de seguridad compartan datos en formato *Record Management System* (RMS) y se comuniquen en tiempo de ejecución mediante eventos, o mediante el protocolo *Inter-Midlet Communication* (IMC). La necesidad de autorizar el acceso a los recursos compartidos por una aplicación define una nueva dimensión en el modelo de seguridad de MIDP. La seguridad a nivel de aplicación, basada en el concepto de *autorización*, es complementaria de la seguridad a nivel de plataforma, basada en el concepto de permiso.

Los mecanismos para validar la identidad de las aplicaciones involucradas se basan en la Infraestructura de Clave Pública X.509 [3,9]. Una aplicación se considera firmada si en su descriptor declara la firma del archivo de aplicación (realizada con la clave privada del vendedor) y el certificado de clave pública del vendedor. Los certificados raíz instalados en el dispositivo permiten verificar los certificados y las firmas declarados.

Para que una aplicación comparta recursos debe declarar autorizaciones de acceso, que se diferencian por las credenciales exigidas a las aplicaciones solicitantes. La declaración de autorización se realiza en el descriptor y puede tomar algunas de las cuatro formas posibles: autorización para las aplicaciones del mismo dominio de seguridad; autorización para las aplicaciones firmadas con un certificado específico;

autorización para las aplicaciones de un vendedor determinado, firmadas con un certificado específico; y autorización para las aplicaciones sin firmar, de un vendedor determinado.

Cuando otra aplicación pretende acceder a recursos compartidos, el dispositivo aplica un conjunto de reglas para determinar si está autorizada. Las reglas evalúan la declaración de la aplicación que comparte y las credenciales de la aplicación que solicita, como ser: el nombre de su dominio de protección y, el nombre, la firma y los certificados de clave pública de su vendedor. Si las credenciales de la aplicación solicitante se ajustan a las autorizaciones declaradas entonces se concede autorización de acceso a los recursos compartidos.

Existen algunas diferencias, además del nivel, entre la autorización de acceso y los permisos. En primer lugar, la autorización de acceso es para todos los recursos compartidos por una aplicación, mientras que el permiso es específico para una función del dispositivo. En segundo lugar, la autorización de acceso dura mientras la aplicación permanece instalada en el dispositivo, mientras que algunos permisos pueden otorgarse por un determinado lapso de tiempo. Sin embargo, ambos niveles de seguridad son complementarios y coexisten en el dispositivo. Por ejemplo, una aplicación restringida a un dominio de seguridad no confiable puede acceder a los recursos compartidos por otra aplicación asociada a un dominio de seguridad confiable.

3 Especificación Formal del Modelo de Seguridad de MIDP 2.0

Esta sección presenta la notación utilizada en la especificación formal del modelo de seguridad de MIDP 2.0, y los principales elementos del modelo de seguridad, que serán necesarios para formalizar la extensión a MIDP 3.0: tipos de datos, estado del dispositivo y eventos que lo afectan. Finalmente, se mencionan algunas propiedades de seguridad verificadas en el modelo.

3.1 Notación

En la formalización se usa la notación estándar para la igualdad y los operadores lógicos $(\wedge, \vee, \neg, \rightarrow, \forall, \exists)$. La implicación y la cuantificación universal pueden codificarse en Coq usando productos dependientes, mientras que la igualdad y los otros operadores pueden definirse inductivamente.

El uso de tipos registros será muy frecuente en la formalización; la definición de éstos genera un tipo inductivo no recursivo R, con un único constructor llamado mkR, y con funciones de proyección $campo_i: R \rightarrow A_i$, donde

$$R =_{def} \{ campo_1: A_1, \ldots, campo_n: A_n \} \quad (1)$$

La aplicación de las proyecciones sobre un registro r se abrevia con la notación punto ($campo_i\ r = r.campo_i$). Por cada $campo_i$ de un tipo de registro r se define una relación binaria $\equiv_{campo_i}$ entre objetos del tipo como sigue,

$$r_1 \equiv_{campoi} r_2 =_{def} \forall j, j \neq i \rightarrow r_1.campo_j = r_2.campo_j \quad (2)$$

Se asume como tipo inductivo predefinido al tipo paramétrico *option T* con los constructores *None: option T* y *Some: T→ option T*.

3.2 Aplicaciones y Política de Seguridad

Las aplicaciones MIDP, comúnmente llamadas MIDlets, suelen agruparse para su distribución en paquetes denominados suites. Se denota al conjunto de los dominios de protección del dispositivo con *Domain* y al conjunto de los identificadores válidos para MIDlet suites con *SuiteID*. Un tipo registro *Suite* representa una suite instalada, con campos para su identificador, su dominio de protección asociado y su descriptor.

$$Suite =_{def} \{ id : SuiteID, domain : Domain, descriptor: Descriptor \} \quad (3)$$

El conjunto de los permisos definidos por cada dominio de protección para cada API o función protegida del dispositivo se denota *Permission*. Una suite además de contener las clases Java y sus recursos, dispone de un archivo especial llamado descriptor. Los permisos requeridos por una suite para su normal funcionamiento se declaran en el descriptor con el predicado *required*, mientras que aquellos considerados opcionales se representan con el predicado *optional*. El descriptor de la suite es un registro que contiene ambos predicados.

$$Descriptor =_{def} \{ required, optional : Permission \rightarrow Prop \} \quad (4)$$

La política de seguridad define para cada dominio de protección los permisos que se otorgan a las suites ligadas al dominio, así como los permisos que pueden ser otorgados por el usuario en una de tres modalidades. Se denota con *Mode* el conjunto enumerado de estas modalidades {*oneshot, session, blanket*}. La política de seguridad es una constante de tipo *Policy*.

$$\begin{aligned} Policy =_{def} \{ \quad & allow: Domain \rightarrow Permission \rightarrow Prop, \\ & user \quad : Domain \rightarrow Permission \rightarrow Mode \rightarrow Prop \} \end{aligned} \quad (5)$$

Los permisos efectivamente otorgados a una suite resultan de la intersección de los permisos requeridos en su descriptor con la unión de los permisos otorgados incondicionalmente por su dominio de protección y los permisos otorgados explícitamente por el usuario.

3.3 Estado del Dispositivo

El estado del dispositivo se modela con el conjunto de suites instaladas, la suite activa, en caso de que haya una, y los permisos otorgados o revocados a todas las suites.

$$\begin{aligned} State =_{def} \{ \quad & suite & & : SuiteID \rightarrow Prop, \\ & session & & : option\ SessionInfo, \\ & granted, revoked & & : SuiteID \rightarrow Permission \rightarrow Prop \} \end{aligned} \quad (6)$$

La suite activa, junto con los permisos otorgados o revocados a ella en la sesión, se representa con el registro *SessionInfo*.

$$SessionInfo =_{def} \{\ id : SuiteID,\ granted,\ revoked : Permission \to Prop\ \} \quad (7)$$

La validez del estado del dispositivo se establece con una serie de condiciones que deben verificarse. Por ejemplo, para que una suite pueda instalarse los permisos requeridos en su descriptor deben ser un subconjunto de los permisos ofrecidos por el dominio de protección. La relación de compatibilidad $\approx$ entre el descriptor *des* y el dominio *dom* se especifica a continuación.

$$des \approx dom =_{def} \forall p: Permission, \quad (8)$$
$$des.required\ p \to allow\ dom\ p \vee \exists m: Mode, user\ dom\ p\ m$$

Una condición de validez del estado *s* es que toda suite instalada debe ser compatible con su dominio de protección asociado. Esta condición se especifica con el siguiente predicado.

$$SuiteCompatible =_{def} \forall ms : Suite, s.suite\ ms \to ms.descriptor \approx ms.domain \quad (9)$$

El detalle de las restantes condiciones de validez del estado se encuentra en [6] y se omiten aquí por razones de espacio.

3.4 Eventos y Propiedades de Seguridad

Los eventos relacionados con la seguridad se modelan como constructores del tipo *Event,* en la Tabla 1. En el evento *request*, la respuesta del usuario se representa con el tipo *UserAnswer*, donde *ua_allow* denota la aceptación y *ua_deny* el rechazo.

$$ua\_allow,\ ua\_deny : Mode \to UserAnswer \quad (10)$$

Tabla 1. Eventos relacionados con la seguridad

Nombre	Descripción	Tipo
install	*Instala una suite*	$SuiteID \to Descriptor \to Domain \to Event$
remove	*Remueve una suite*	$SuiteID \to Event$
start	*Inicia una sesión*	$SuiteID \to Event$
terminate	*Finaliza una sesión*	$Event$
request	*Solicita un permiso*	$Permission \to option\ UserAnswer \to Event$

La conducta de los eventos se especifica mediante pre y poscondiciones. Las precondiciones están definidas en términos del estado del dispositivo, mientras que las poscondiciones se definen en términos del estado anterior, el estado posterior y una respuesta opcional del dispositivo. La respuesta del dispositivo se representa con el tipo *Response*, donde el valor *allowed* denota la aceptación y *denied* el rechazo.

$$Response := allowed \mid denied \quad (11)$$

Los predicados *Pre* y *Pos* especifican respectivamente las pre y poscondiciones de los eventos. Por razones de espacio sólo se incluye su tipo, estando el detalle disponible en [6].

$$Pre : State \rightarrow Event \rightarrow Prop \quad (12)$$
$$Pos : State \rightarrow State \rightarrow option\ Response \rightarrow Event \rightarrow Prop$$

La conducta de cada evento se especifica mediante una *relación de ejecución en un paso*. Esta relación establece que el estado del dispositivo permanecerá inalterado, si la precondición del evento no se cumple; y cambiará según la poscondición del evento, si se satisface la precondición.

La relación de ejecución conduce al concepto de *sesión de ejecución*. Una sesión se define, a partir de un estado válido del dispositivo, como una secuencia de pasos que comienzan con el evento *start* y finalizan con el evento *terminate*.

La validez del estado del dispositivo es una propiedad que se demuestra invariante con respecto a la ejecución en un paso de cualquier evento, y también con respecto a una sesión de ejecución. Este resultado se generaliza para cualquier propiedad invariante respecto a un paso y a una sesión de ejecución. La aplicación de estos resultados permite verificar, entre otras propiedades, que si un permiso es revocado por un usuario, por el resto de una sesión, cualquier solicitud posterior del mismo permiso será también rechazada.

4 Formalización del Modelo de Autorización de MIDP 3.0

En esta sección se extiende la formalización descrita en la sección previa con los nuevos conceptos introducidos en MIDP 3.0. Se detallan los tipos de datos, una nueva condición de validez del estado y el evento de autorización. Asimismo, se analiza la conservación de las propiedades de seguridad demostradas en MIDP 2.0 y se mencionan nuevas propiedades referentes a la extensión.

4.1 Seguridad a Nivel de Aplicación

Se introducen nuevos tipos de datos: *Vendor* denota los nombres de vendedores de suites, *Certificate* representa el conjunto de los certificados de clave pública y *Signature* la firma del archivo de la aplicación.

El descriptor de la suite se extiende con el nombre del vendedor y opcionalmente su certificado de clave pública y la firma del archivo. Una suite declara autorización de acceso con un predicado para cada conjunto de suites involucradas. El predicado *domainAuthorization* denota la autorización de acceso para todas las suites del mismo dominio. La autorización de acceso para un conjunto de suites firmadas con un certificado particular se establece con el predicado *signerAuthorization*. El predicado *vendorUnsignedAuthorization* hace lo propio para el conjunto de suites sin firmar de un

vendedor, mientras que *vendorSignedAuthorization* denota la autorización de acceso al conjunto de suites, firmadas con un certificado específico de un vendedor.

$$\begin{aligned} Descriptor =_{def} \{\ & required, optional && : Permission \to Prop, \\ & vendor && : Vendor, \\ & jarSignature && : option\ Signature, \\ & signerCertificate && : option\ Certificate, \\ & domainAuthorization && : Prop, \\ & signerAuthorization && : Certificate \to Prop, \\ & vendorUnsignedAuthorization && : Vendor \to Prop, \\ & vendorSignedAuthorization && : Vendor \to Certificate \to Prop\ \} \end{aligned} \quad (13)$$

4.2 Nuevo Estado del Dispositivo

El estado del dispositivo se extiende para representar las autorizaciones de acceso concedidas y denegadas por una suite a otra. Las autorizaciones concedidas se representan con *authorized* y las autorizaciones denegadas con *unauthorized*.

$$\begin{aligned} State =_{def} \{\ & suite && : SuiteID \to Prop, \\ & session && : option\ SessionInfo, \\ & granted, revoked && : SuiteID \to Permission \to Prop, \\ & authorized, unauthorized && : SuiteID \to SuiteID \to Prop\ \} \end{aligned} \quad (14)$$

Se agrega una condición de validez del estado del dispositivo, respecto de las autorizaciones de acceso. El predicado *ValidAuthorization* establece que una suite no puede estar autorizada y desautorizada, al mismo tiempo, por otra suite.

$$\begin{aligned} ValidAuthorization =_{def}\ & \forall idGrn, idReq: SuiteID, \\ & s.authorized\ idGrn\ idReq \to \neg\, s.unauthorized\ idGrn\ idReq \wedge \\ & s.unauthorized\ idGrn\ idReq \to \neg\, s.authorized\ idGrn\ idReq \end{aligned} \quad (15)$$

4.3 Solicitud de Autorización de una Aplicación

Un nuevo evento *authorization* modela la solicitud de autorización de acceso por parte de una suite a los recursos de la suite activa.

$$authorization : SuiteID \to Event \quad (16)$$

El evento tiene como precondición que en el estado *s* exista una suite activa, y que el identificador de la suite solicitante *idReq* corresponda al identificador de una suite instalada en el dispositivo.

$$\begin{aligned} & Pre\ s\ (authorization\ idReq) =_{def} \forall ses: SessionInfo \\ & s.session = Some\ ses \to \exists msReq: Suite, s.suite\ msReq \wedge msReq.id = idReq \end{aligned} \quad (17)$$

La poscondición establece las posibles respuestas del dispositivo y la relación entre el estado previo (*s*) y posterior (*s'*) a la ejecución del evento. Se analiza por casos, según se permita o se niegue el acceso.

Se permite el acceso si la suite solicitante *idReq* ya estaba autorizada, en cuyo caso el estado del dispositivo permanece inalterado. También se permite el acceso si *idReq* no estaba desautorizada y valida las reglas de autorización. En este caso el estado resulta modificado ya que *idReq* pasa a integrar el conjunto de suites autorizadas por la suite activa.

$$Pos\ s\ s'\ (Some\ allowed)\ (authorization\ idReq) =_{def} \forall ses: SessionInfo, \quad (18)$$
$$s.session = Some\ ses \rightarrow (s = s' \wedge s.authorized\ ses.id\ idReq) \vee$$
$$(\neg s.unauthorized\ ses.id\ idReq \wedge AccessAuthorization\ s\ ses.id\ idReq \wedge$$
$$s \equiv_{authorized} s' \wedge s'.authorized\ ses.id\ idReq)$$

Se niega el acceso si la suite solicitante *idReq* ya estaba desautorizada, en cuyo caso permanece inalterado el estado del dispositivo. También se niega el acceso si *idReq* no estaba autorizada y no valida las reglas de autorización. El estado del dispositivo en este caso cambia porque *idReq* se agrega al conjunto de suites desautorizadas por la suite activa.

$$Pos\ s\ s'\ (Some\ denied)\ (authorization\ idReq) =_{def} \forall ses: SessionInfo, \quad (19)$$
$$s.session = Some\ ses \rightarrow (s = s' \wedge s.unauthorized\ ses.id\ idReq) \vee$$
$$(\neg s.authorized\ ses.id\ idReq \wedge \neg AccessAuthorization\ s\ ses.id\ idReq \wedge$$
$$s \equiv_{unauthorized} s' \wedge s'.unauthorized\ ses.id\ idReq)$$

El predicado *AccessAuthorization* representa la evaluación de las reglas de autorización de acceso en un estado del dispositivo.

$$AccessAuthorization: State \rightarrow SuiteID \rightarrow SuiteID \rightarrow Prop \quad (20)$$

4.4 Extensión Conservativa de Propiedades de Seguridad

La formalización propuesta para adecuar el modelo de seguridad de MIDP 2.0 a las novedades de MIDP 3.0 introduce varios cambios. Modifica el estado del dispositivo para llevar un registro de las suites autorizadas y desautorizadas; exige una nueva condición para la validez del estado del dispositivo; e introduce la autorización de acceso como un evento nuevo, relacionado con la seguridad.

Se demuestra que la validez del estado se conserva en la formalización propuesta. La validez se mantiene invariante con respecto a la ejecución en un paso de cualquier evento, y también es invariante con respecto a una sesión de ejecución. Esta propiedad permite verificar, entre otros teoremas, que si una suite ha sido desautorizada en una sesión de ejecución, cualquier solicitud de acceso de la misma suite, seguirá siendo rechazada. Todas estas propiedades fueron formalizadas y demostradas usando el asistente de pruebas Coq. Se omiten aquí las formalizaciones y las pruebas por razones de espacio.

5 Conclusiones y Trabajos Futuros

La reciente publicación del perfil MIDP 3.0 es una respuesta natural de la industria ante la creciente y demandante comunidad de usuarios y desarrolladores de aplicaciones para dispositivos móviles.

La formalización del modelo de seguridad de MIDP 2.0 planteada en [6] es pionera en su cometido y constituye una poderosa herramienta para entender las características y limitaciones del modelo. Dicha formalización plantea y demuestra propiedades de seguridad que debe cumplir cualquier implementación del perfil.

Este trabajo presenta una extensión al modelo de seguridad de MIDP 2.0, sin precedentes, que contempla los cambios introducidos en MIDP 3.0. En particular se modela una nueva dimensión de la seguridad, a nivel de aplicación, logrando que la extensión propuesta conserve las propiedades de seguridad en este nuevo contexto. De esta forma se actualiza la formalización, manteniendo su vigencia como una herramienta útil para razonar sobre el modelo de seguridad de MIDP, tanto a nivel de plataforma como de aplicación.

Como trabajo futuro, en proceso actual de realización, se plantea la obtención de un prototipo ejecutable de la especificación, en un lenguaje funcional de alto nivel. El modelo ejecutable podría utilizarse en una técnica complementaria de la verificación formal, como es el testing de caja negra, donde oficie de oráculo frente a casos de pruebas derivados de la especificación. Para que esto sea posible, la representación de alto orden de la formalización deberá ser refinada en una representación concreta, a la cual aplicar el mecanismo de extracción de programas de Coq, siguiendo por ejemplo la metodología propuesta por coautores de este artículo en [6].

Referencias

1. Java Platform Micro Edition, http://java.sun.com/javame/index.jsp, último acceso: Mayo de 2008.
2. JSR 37 Expert Group: Mobile Information Device Profile for Java Micro Edition. Version 1.0. Sun Microsystems, Inc. (2000)
3. JSR 118 Expert Group: Mobile Information Device Profile for Java Micro Edition. Version 2.0. Sun Microsystems, Inc. and Motorola, Inc. (2002)
4. JSR 271 Expert Group: Mobile Information Device Profile for Java Micro Edition.Version 3.0, Public Review Specification. Motorola, Inc. (2007)
5. Zanella Béguelin, S.: Especificación formal del modelo de seguridad de MIDP 2.0 en el Cálculo de Construcciones Inductivas. Master's thesis, Universidad Nacional de Rosario. (2006)
6. S. Zanella Béguelin, G. Betarte, y C. Luna. A formal specification of the MIDP 2.0 security model. T. Dimitrakos et al. (Eds.): FAST 2006, LNCS 4691, pp. 220–234, 2007. Springer-Verlag Berlin Heidelberg 2007.
7. Yves Bertot y Pierre Castéran. Interactive Theorem Proving and Program Development. Coq'Art: The Calculus of Inductive Constructions. Texts in Theoretical Computer Science. Springer-Verlag, 2004. ISBN 3-540-20854-2.
8. The Coq Development Team: The Coq Proof Assistant Reference Manual VersionV8.1. (2006)
9. Internet X.509 Public Key Infrastructure, http://www.ietf.org/rfc/rfc2459, último acceso: Mayo de 2008.