Patrones de Seguridad aplicados a la función Autorización

Juan Carlos Ramos, Susana Romaniz, Marta Castellaro

Facultad Regional Santa Fe - Universidad Tecnológica Nacional
Lavaise 610 (S3004EWB) Santa Fe Argentina.
{jcramos, sromaniz, mcastell}@frsf.utn.edu.ar

Resumen. La característica de "software seguro" reside en la naturaleza de los procesos y las prácticas utilizadas para especificar, diseñar, desarrollar y desplegar el software. Un proceso mejorado para la seguridad incorpora prácticas para reducir el número de errores y debilidades explotables. Los patrones de seguridad constituyen un aporte para salvar el vacío entre teoría y práctica, y pueden emplearse en organizaciones con distinto grado de madurez. Este trabajo se centra en patrones de seguridad asociados al diseño y considera la inclusión de funciones relativas al control de acceso. En particular trata la autorización de los privilegios de usuarios: proceso en que se determina qué acciones puede ejecutar un usuario autenticado, tanto sobre las funciones propias del software como sobre los datos que éstas manipulan. Se analiza un prototipado del patrón ROLE-BASED ACCESS CONTROL (RBAC), aplicado a sistemas de información de la gestión universitaria.

Palabras claves: Software seguro. RBAC. Autorización

1 Introducción

La principal característica de un software seguro reside en la naturaleza de los procesos y las prácticas utilizadas para especificar, diseñar, desarrollar y desplegar el software [1]. Un proyecto que adopta un proceso de desarrollo de software mejorado para la seguridad incorpora un conjunto de prácticas que permiten reducir el número de errores y debilidades explotables. A lo largo del tiempo, estas prácticas se vuelven más sistemáticas, por lo que debería disminuir la probabilidad que tales vulnerabilidades estén presentes en el software en el momento en que se lo libera. Los resultados en el campo de la investigación y las experiencias en la industria indican la importancia de reducir tales vulnerabilidades potenciales tan temprano como sea posible dentro del ciclo de vida del desarrollo del software. La adopción de procesos y prácticas mejoradas para la seguridad resulta muchísimo más rentable que la solución tan difundida en la actualidad de desarrollar y liberar parches para el software operativo [2].

Esta atención temprana de la seguridad tiene que ver con la adopción de un conjunto de actividades que hacen posible la integración de la misma en el ciclo de vida de desarrollo de software [3], las que incluyen: i) identificar objetivos de seguridad, ii) aplicar guías de diseño de seguridad, iii) crear modelos de amenazas, iv) conducir

revisiones seguridad de la arquitectura y el diseño, v) completar revisiones de seguridad de la implementación, y vi) ejecutar revisiones de seguridad del despliegue [4].

El proceso de desarrollo de software seguro encuentra en los *patrones de seguridad* una vía para salvar el vacío existente entre teoría y práctica. Si bien existen abordajes teóricos, éstos se encuentran limitados a sistemas de relativa complejidad, además de requerir de un nivel de experiencia y conocimiento que no está disponible en el nivel necesario. A esto se agrega que el requerimiento de seguridad es uno más de los muchos que se deben atender durante el desarrollo de software, pudiendo observarse un abordaje ad-hoc.

2 ¿Qué hace seguro al software?

Antes de poder determinar las características de un software para hacer de éste un software seguro, se debe establecer cuáles son los problemas de seguridad que deberán ser atendidos durante su proceso de desarrollo, y seleccionar qué patrones de seguridad aplicar en las diferentes fases de dicho proceso. Es necesario definir las propiedades mediante las que puedan ser descriptas estas características. Estas propiedades comprenden:

1. un conjunto de propiedades fundamentales cuya presencia (o ausencia) son el terreno firme que hacen seguro al software (o no),
2. un conjunto de propiedades conducentes que no hacen seguro al software en forma directa, pero que permiten caracterizar cuán seguro es un software.

2.1 Taxonomía de las propiedades de seguridad

Centrando el análisis en las *propiedades fundamentales* (que se pueden tratar también como atributos de seguridad), se considera que los efectos de vulnerar la seguridad del software se pueden describir en términos de los efectos sobre estas propiedades fundamentales. Las mismas se enumeran a continuación. [5]

Confidencialidad. El software debe asegurar que cualquiera de sus características (incluidas sus relaciones con su ambiente de ejecución y sus usuarios), los activos que administra, y/o su contenido se encuentran enmascarados u ocultos de las entidades no autorizadas.

Integridad. El software y los activos que administra son resistentes y flexibles a la subversión, la que se logra mediante modificaciones no autorizadas (del código, los activos administrados, la configuración o el comportamiento del software) por parte de entidades autorizadas, o cualquier modificación por entidades no autorizadas; se debe preservar tanto durante el desarrollo del software como durante su ejecución.

Disponibilidad. El software debe estar operativo y accesible a sus usuarios autorizados (humanos o procesos) siempre que se lo necesite; simultáneamente, su funcionalidad y sus privilegios deben ser inaccesibles a usuarios no autorizados (humanos y procesos) en todo momento. Para las entidades que actúan como usuarios se requieren dos propiedades adicionales, generalmente asociadas con los usuarios finales, que se indican a continuación.

Responsabilización. (en idioma inglés, *accountability*). Todas las acciones relevantes relacionadas con la seguridad del software que actúa como usuario se deben registrar y rastrear con atribución de responsabilidad; el rastreo debe ser posible tanto durante como a posteriori de la ocurrencia de las acciones registradas.
No repudiación. La habilidad de prevenir que el software que actúa como usuario desmienta o niegue la responsabilidad relativa a acciones que han sido ejecutadas; asegura que no se puede subvertir o eludir la propiedad 'responsabilización'.

3 Atributos requeridos en un software seguro

Con el propósito de proveer las propiedades fundamentales indicadas, es preciso incluir en el software seguro funciones relativas al *control de acceso. Estos requerimientos* están presentes tradicionalmente tanto en los componentes de la infraestructura como en las aplicaciones que conforman los sistemas de información basados en tecnologías de la información. Estas funciones hacen posible forzar el cumplimiento de políticas en base a las que los usuarios verificados ejecutan las diversas funciones provistas por el software conforme a su rol, y se evita que lleven a cabo aquéllas que no le competen. El control de acceso incluye: [6]

- *Autenticación de usuarios*
- *Autorización de sus privilegios*
- *Auditoría para controlar y registrar las acciones de los usuarios*

El presente trabajo centra el análisis en la segunda de las funciones; en particular, la *autorización de los privilegios de usuarios* comprende el proceso en que se determina qué acciones puede ejecutar un usuario autenticado, tanto en los que se refiere sobre las funciones propias del software como sobre los datos que éstas manipulan. A los fines de diseñar dicho proceso, se hace uso del conocimiento documentado vía *patrones de seguridad.*

4 Patrones de seguridad

El concepto de patrón es conocido por la comunidad como una solución a problemas comunes en el desarrollo de software, cuya efectividad se ha comprobado resolviendo problemas similares en ocasiones anteriores y tal que sea reutilizable (aplicable a diferentes problemas de diseño en distintas circunstancias). En el caso de los aspectos de seguridad del software, que se encuentran a lo largo de todas las fases del desarrollo, su definición original [7] establece que: "*Cada patrón es una regla de tres partes, expresada como una relación entre un determinado contexto, un determinado sistema de fuerzas que ocurren repetidamente en este contexto, y una determinada configuración de software que permite que estas fuerzas se resuelvan a sí mismas*."

Extendiendo el concepto a la seguridad del software, los patrones de seguridad documentan soluciones bien conocidas a problemas recurrentes de seguridad de la información, permitiendo una transferencia eficiente de experiencia y de conocimientos. Los mismos aplican el concepto de patrón al dominio de la seguridad, describiendo un problema particular de seguridad recurrente que ocurre en un contexto específico y presentando una solución genérica bien probada y aceptada por

la comunidad de expertos [8]. Al explicitar los supuestos bajo los que resultan aplicables sus soluciones, reducen el riesgo de su empleo inadecuado.

La solución propuesta por un patrón de seguridad consiste en un conjunto de roles interactuantes que pueden ser organizados en múltiples estructuras (aplicables a la fase de Requerimientos, Diseño o de Implementación, según corresponda el patrón) concretas, así como también un proceso para crear una estructura particular en éstas [9]. De acuerdo a lo expresado en [10] "*Un patrón define un proceso y una cosa: la 'cosa' es creada por el 'proceso'*." Los patrones de seguridad se pueden categorizar de acuerdo a un punto de vista asociado a un ciclo de vida de desarrollo de software [11]. De esta forma existen patrones para la fase de requerimiento, patrones para la fase diseño, y patrones para la fase de implementación. Este trabajo se centra en el empleo de *patrones de seguridad para el diseño*.

5 Empleo de patrones de seguridad para el diseño de la función Autorización

En las organizaciones suele haber muchos sistemas, los que son utilizados por múltiples usuarios, los que a su vez tienen diferentes privilegios para usar los sistemas y las diferentes funciones y datos que estos gestionan. Los privilegios forman diferentes subconjuntos los que pueden ser asignados a diferentes usuarios (personas realizando el mismo trabajo o similar). El problema que se plantea entonces es resolver la *asignación de privilegios de usos* (*autorización*) de una manera ágil y flexible.

Una primera solución a este problema se conoció como el patrón *ROLE* [12]. Esto evolucionó, incorporando el concepto de *ROLE-BASED ACCESS CONTROL* (*RBAC*) [13], donde el problema se enfoca en las funciones de trabajo que las personas tienen que realizar en su actividad diaria. RBAC está basado en el principio de acceso con el menor privilegio [14]: un rol sólo concede los mínimos privilegios requeridos por un individuo para realizar su trabajo.

El principio básico de RBAC (tal como se muestra en la Figura 1 [15]) es que un sujeto puede ejecutar una transacción sólo si tiene asignado un rol con los permisos requeridos para hacerlo. Los roles pueden ser asignados estática o dinámicamente, mientras que una jerarquía de roles simplifica el modelado de roles habilitando roles superiores para heredar los privilegios a roles inferiores. Se pueden aplicar restricciones a las asignaciones y jerarquías de roles para prevenir que se asignen roles conflictivos a los usuarios, o que se asigne demasiada autoridad a un usuario.

5.1 Proceso de implementación

Si bien la teoría de base de RBAC es relativamente simple, introducirlo en una organización madura es un proyecto de trabajo intensivo. Por esta razón, se presentará un proceso sugerido para incorporarlo a una organización, y luego se lo redefinirá para poder comenzar con el prototipado de implementación del mismo a un caso concreto.

En [16] los autores sugieren el siguiente proceso para una planificación de Roles:

1. **Establecer el Caso de Negocio**. Se debe seleccionar un caso de negocio bien comprendido para obtener patrocinio ejecutivo y fondos; se definen métricas que permiten evaluar los resultados.

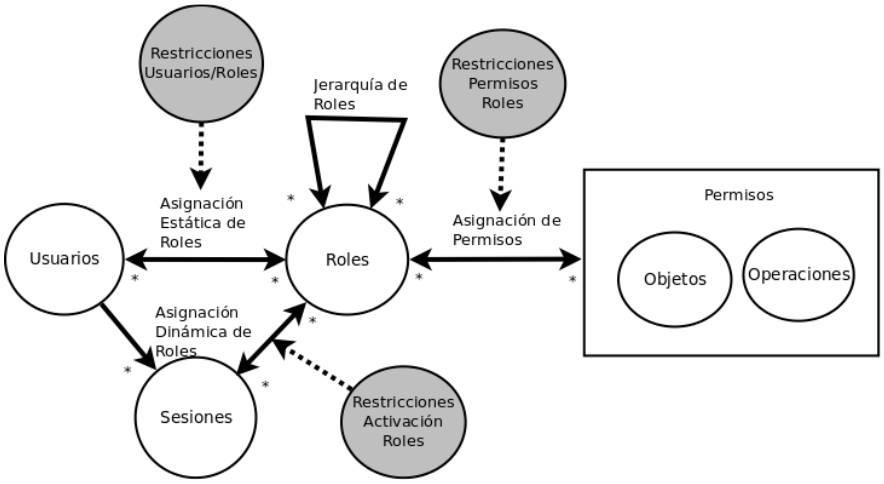

Fig. 1. Estándar RBAC.

2. **Evaluar las capacidad para RBAC**. Determinar cuál es la experiencia y herramientas disponibles en procesos de asignación de autorización.
3. **Seleccionar herramientas y colaboradores para la implementación**. De acuerdo a los niveles y características a implementar de RBAC, seleccionar la herramienta. Es importante incorporar al menos un colaborador con experiencia.
4. **Establecer línea base de Roles**. Se establece una base de roles, a partir de la cual se realizará la definición de roles de granularidad fina.
5. **Identificar participantes e interesados del proyecto**.
6. **Articular metas y objetivos para el proyecto**. Los interesados técnicos y de negocio deben acordar las metas y objetivos del proyecto: enfoques y estrategias de implementación, definición de métricas, planificación del proyecto, etc.
7. **Implementación iterativa**. La estrategia de implementación debe ser realizada en fases para que la misma sea exitosa.

Este proceso es válido para llevar a cabo una implementación de Roles teniendo cierta experiencia previa (actividades 2 y 4, especialmente), pero en nuestro caso, se plantea hacer una experiencia piloto (actividad 7) por lo que se adapta el mismo para seguir con nuestro proyecto. El proceso a seguir es el siguiente:

1. Establecer un Caso de Negocio
2. Definir un esquema de roles para el caso de negocio.
3. Establecer la línea base de Roles

5.1.1. Caso de Negocio

Como caso para realizar el prototipado señalado se seleccionó trabajar con los sistemas de información que se aplican a la gestión de una entidad universitaria (en particular, una facultad). Para establecer la definición de roles se eligió una de las secretarías que conforman la estructura organizativa de una Institución Universitaria, a la se denomina simplemente “Secretaría”; la misma tiene una estructura jerárquica clásica

(ver Figura 2), conformada de la siguiente manera: Secretario, Direcciones (Académica, Departamentos, Posgrado), Áreas (Educación a Distancia, Acceso a la Universidad). La Dirección de Departamentos, a su vez, se compone de un Departamento por cada carrera; y la Dirección Académica se conforma por cuatro Departamentos (Alumnos, Biblioteca, Legajos, y Títulos y Egresados). Además, cada unidad organizacional está compuesta por 'Administrativos', que colaboran en la realización de las funciones de la unidad.

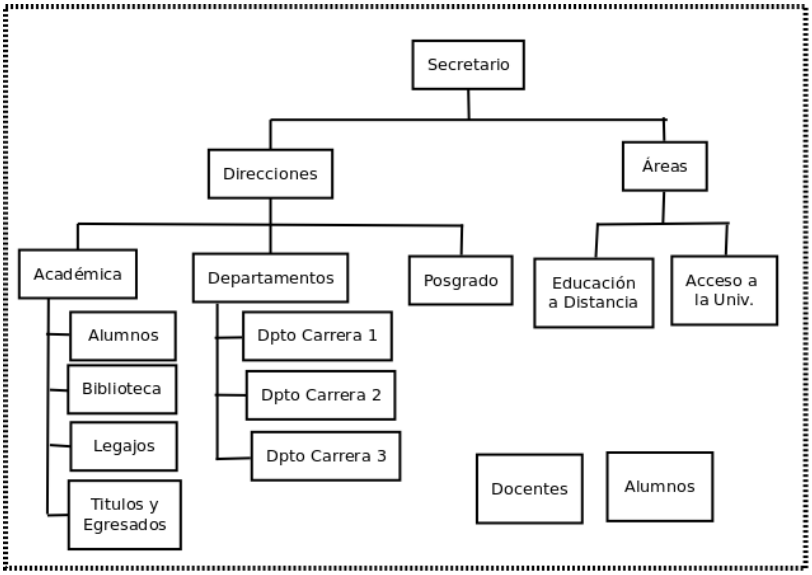

Fig. 2. Estructura organizativa del caso de estudio.

Los principales *Objetos protegidos* (*OP*) considerados para este caso son (no es la lista exhaustiva, y sólo se considera a nivel de 'sistema'): a) Sistema de Gestión Académica, b) Sistema de Autogestión Docentes y Alumnos, c) Sistemas de Información Gerencial, d) Información de Ingresantes, y e) Sistema de Gestión de Biblioteca.

5.1.2 Esquemas de roles.

La estructura y objetos definidos en 5.1.1 permiten definir los siguientes *Roles de Negocio* (*RN*): a) Secretario, b) Director Académico, c) Director de Departamento Carrera, d) Director de Posgrado, e) Responsable Área Educación a Distancia, f) Coordinador Acceso a la Universidad, g) Director Dpto. Alumnos, h) Director Dpto. Biblioteca, i) Director Dpto. Legajos, j) Director Dpto. Títulos y Egresados, k) Administrativo Departamento Carrera, l) Administrativo Posgrado, m) Administrativo Educación a Distancia, n) Administrativo Acceso a la Universidad, o) Administrativo Alumnos, p) Administrativo Biblioteca, q) Administrativo Legajos, r) Administrativo Títulos y Egresados, s) Docentes, y t) Alumnos.

Una característica destacable de la Institución es la existencia de usuarios que, debido a que participan en ella con diferentes roles (por ejemplo: Secretario, Docente

y Alumno) simultáneamente, deben, en consecuencia, contar con distintos privilegios para acceder a los sistemas de información. Esto conlleva un importante grado de complejidad en definición del modelo a adoptar para el patrón de seguridad RBAC.

Para explicitar los privilegios que se pueden otorgar sobre los OP, sin asociarlos directamente a Usuarios o a RN, se definen los *Roles Técnicos* (*RT*), planteados en el alto nivel. En la Tabla 1 se muestra sólo un subconjunto representativo, vinculado a los Privilegios que estos tienen sobre los OP. Finalmente, y según las definiciones anteriores, se hace la primera asignación de RT a RN, como se muestra en Tabla 2.

Rol Técnico	**Privilegio**	**Objeto Protegido**
Secretario Académico	Análisis de la información de carreras, cursos y alumnos	Sistema Información Gerencial
Administrador Sistema Académico	Configuración del sistema	Sistema Académico
Administrativo Sistema Académico	Carga de operaciones diarias	Sistema Académico
Docente	Consultas de cursos y actualización de datos de cursos	Sistema Autogestión
Estudiante	Consultas de información propia e inscripciones	Sistema Autogestión
Director Dpto. Carrera	Análisis de Información de carrera, cursos y alumnos	Sistema Información Gerencial
Administrativo Dpto. Carrera	Consultas evolución alumnos carrera	Información Alumnos (Historia Académica)
Administrativo Ingresantes	Consultas y actualización	Información Ingresantes
Administrador Biblioteca	Configuración	Sistema Gestión Bibliotecas
Administrativo Biblioteca	Consultas y actualización	Sistema Gestión Bibliotecas
Socio Biblioteca	Consultas y préstamos	Sistema Gestión Bibliotecas

Nota: Esta definición relaciona un privilegio directamente con un OP (relación 1 a 1).

Tabla 1. Definición del Rol Técnico - Privilegio - Objeto Protegido.

5.1.3 Línea base de roles.

Si bien los RN sirven para asignar un rol a algún Usuario que integra una estructura organizativa, son los RT los que en definitiva permitirán hacer la asignación de Privilegios sobre los OP. Por lo tanto, se utilizan los RT para realizar la definición de la Línea Base de Roles. La Tabla 2 es una vista parcial de la Línea Base resultante para el caso (sólo se consideran algunos ítem como ejemplos).

El primer nivel de asignación de roles a Usuarios que se establece emplea 'RBAC Flat' (correspondiente al *Level 1* de los cuatro niveles definidos por los estándares RBAC [15]). En la Tabla 3, a cada Usuario se le asigna un RN, el que a su vez le direcciona a un RT. Los privilegios asociados son los que se corresponden a los definidos en la Tabla 1, y el modelo RBAC para este caso se muestra en la Figura 3.

Analizando la Tabla 3, se observa que es posible establecer niveles de herencia y/o composición entre algunos RT específicos. Por ejemplo, un *Administrador Sistema*

Rol de Negocio	Rol Técnico
Secretario	Secretario Académico
Director Académico	Administrador Sistema Académico
Administrativo Alumnos	Administrativo Sistema Académico
Docente	Docente
Estudiante	Estudiante
Director de Dpto. Carrera	Director Dpto. Carreras
Administrativo Dpto. Carrera	Administrativo Dpto. Carrera
Administrativo Acceso a la Universidad	Administrativo Ingresantes
Director Dpto. Biblioteca	Administrador Biblioteca
Administrativo Dpto. Biblioteca	Administrativo Biblioteca
Docente	Socio Biblioteca
Estudiante	Socio Biblioteca

Tabla 2. Asignación de Rol Técnico a un Rol de Negocio.

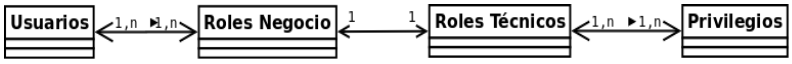

Fig. 3. RBAC Flat adaptado.

Usuario	Rol de Negocio	Rol Técnico
Juan P.	Secretario	Secretario Académico
	Docente	Docente
	Docente	Socio Biblioteca
María V.	Director Académico	Administrador Sistema Académico
Patricia Z.	Administrativo Alumnos	Administrativo Sistema Académico
Horacio L.	Director de Departamento Carrera	Director Dpto. Carreras
	Docente	Docente
	Docente	Socio Biblioteca
Nora R.	Administrativo Departamento Carrera	Administrativo Dpto. Carrera
Lucía M.	Director Dpto. Biblioteca	Administrador Biblioteca
Alejandra A.	Administrativo Dpto. Biblioteca	Administrativo Biblioteca
Luis J.	Administrativo Acceso a la Univ.	Administrativo Ingresantes
Susana R.	Docente	Docente
	Docente	Socio Biblioteca
Juan R.	Estudiante	Estudiante
	Estudiante	Socio Biblioteca

Tabla 3. Asignación de Rol Técnico a Usuarios conforme su Rol de Negocio.

Académico puede heredar los privilegios de un *Administrativo Sistema Académico*, y de esta manera aumentar sus privilegios; un *Secretario Académico* puede agregar a sus privilegios los de un *Docente;* un *Administrador Biblioteca* puede heredar los

privilegios de un *Administrativo Biblioteca*; un *Docente* puede agregar los privilegios de un *Socio Biblioteca*; etc. De esta manera, se presentan relaciones de herencia y composición entre roles. Para modelar esta relación de herencia y composición se aplica un patrón de diseño *COMPOSITE* [16], resultando el modelo representado en la Figura 4 a) Herencia de RN, y b) Herencia y Composición de RT, en el que se muestra hasta un nivel 3 de herencia).

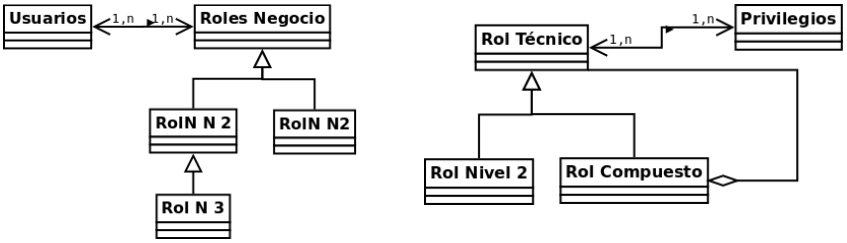

Fig. 4. RBAC con herencia y composición.

Para una mayor comprensión, se instancian algunas de las clases propuestas (solamente a nivel de roles). Entonces, por ejemplo: "AdministradorSistemaAcadémico:RolNivel2" es un "AdministrativoSistemaAcadémico:RolTécnico"; "AdministradorBiblioteca:RolNivel2" es un "AdministrativoBiblioteca:RolTécnico"; "Docente: RolCompuesto" agrega "SocioBiblioteca:RolTécnico".

De esta manera, con el modelo anterior, es posible representar hasta tres niveles de herencia de roles, y la estructura compuesta es aplicable a cualquier nivel. Es necesario incorporar instancias de roles no asignables cuando se requiere representar roles simplemente compuestos por otros roles (por la relación de herencia entre Rol Técnico y Rol Compuesto, Figura 4 b). El modelo definido constituye una primera fase del proceso de implementación de RBAC descripto en el punto 5.1, el cual puede ser extendido en fases posteriores, refinando las relaciones encontradas y/o incorporando nuevas relaciones.

6 Conclusiones

El 'Caso de Negocio' se seleccionó con el propósito de atender a requerimientos relativos al mejoramiento del proceso de diseño de los componentes de software de los sistemas de información que prestan servicios a las diferentes unidades funcionales de la Institución. Estos sistemas, cuya criticidad, número y universo de usuarios ha venido evolucionando con un constante aumento, plantean nuevos desafíos en cuanto a la atención de los requerimientos funcionales relacionados con el control de acceso a los recursos que proveen los diferentes componentes de software y, en particular, con la autorización de los privilegios de usuarios. El objetivo último es lograr que dichos sistemas empleen componentes de software que posean mejores propiedades de seguridad.

El relevamiento preliminar realizado ha permitido observar que los diferentes sistemas de información atienden a estos requerimientos aplicando criterios ad-hoc,

fuertemente determinados por la plataforma tecnológica que los soporta. Así, uno de los requerimientos planteados es la necesidad de contar con *guías de diseño* para la implementación de funciones relacionadas con la autorización, las que se aplicarán en el desarrollo de nuevos sistemas e integración de nuevos componentes. Además, en el caso que el nivel de impacto sea aceptable, se utilizarán para introducir mejoras en los sistemas existentes que lo permitan.

Este trabajo ha sido precedido de otras actividades conducentes a la atención de la seguridad de la información de manera integral, tal como la creación de un Comité de Seguridad a nivel institucional, así como la definición y difusión de Políticas de Seguridad. Actualmente se está completando el relevamiento de OP y refinando el modelo que se propone en este trabajo. También se están diseñando instrumentos que permiten gestionar (procesos de Alta/Baja/Modificaciones) de autorizaciones de acceso.

Este prototipo que aplica RBAC ha permitido difundir los Patrones de Seguridad como una herramienta disponible y que se puede emplear como uno de los primeros pasos para comenzar a "atender a la Seguridad" en organizaciones que emplean sistemas centrados en software.

Referencias

1. McGraw, G.: Software Security: Building Security In. Addison-Wedsley. EEUU (2006).
2. Romaniz, S.: Buenas prácticas de elicitación de los requerimientos de seguridad. IV Congreso Iberoamericano de Seguridad Informática -CIBSI2007-, Argentina (2007).
3. Meier, J. et al.: "Security engineering explained", http://www.microsoft.com/download/en/confirmation.aspx?id=20528, (2005). (visitado en Julio 2012).
4. Castellaro, M. y otros: Hacia la Ingeniería de Software Seguro. XV Congreso Argentino de Ciencias de la Computación -CACIC2009-, Argentina (2009).
5. Avizienis, A. et al.: Basic Concepts and Taxonomy of Dependable and SecureComputing. IEEE Transactions on Dependable and Secure Computing. Vol.1 No.1 (2004).
6. Housley, R.: Guidance for Authentication, Authorization, and Accounting (AAA) Key Management, RFC-4962, The IETF Trust (2007)
7. Coplien, J.: Design Pattern Definition - Software Patterns, http://www.hillside.net/component/content/article/50-patterns/222-design-pattern-definition (visitado en Julio de 2012).
8. Schumacher, M.: Security engineering with patterns–origins, theoretical model, and new applications. Springer (2003).
9. Schumacher, M. et al.: Security Patterns: Integrating Security and Systems Engineering. John Wiley & Sons Inc, EEUU (2006).
10. Alexander, C.: The Timeless Way of Building. Oxford University Press, EEUU (1979).
11. Nobukazu Yoshioka and Hironori Washizaki, K. M.: A survey on security patterns. Progress in Informatics (2008).
12. Yoder, J.: Architectural Patterns for Enabling Application Security, PLoP`97 (1998).
13. Fernandez, E. and Pan, R.: A pattern language for security models. 8th Conference of Pattern Languages of Programs (PloP) (2001).
14. Saltzer, J. and Schroeder, M.: The Protection of information in computer systems, Proceedings of the IEEE, vol. 63, no. 9 (Sept 1975), pp. 1278-1308.
15. Ferraiolo, D. et al.: Proposed NIST standard for role-based access control. ACM Transactions on Information and System Security, 4(3):224–274 (August 2001).
16. Emden, T.: Implementing Role-Based Access Controls in the Enterprise, Qubera Solutions (2011).
17. Gamma, E. et al.: Design Patterns - Elements of Reusable Object-Oriented Software, Addison-Wesley (1995).