Patrones de Ataque y de Seguridad como guía en el desarrollo de software

Ignacio Ramos

Universidad Tecnológica Nacional, Facultad Regional Santa Fe, Argentina
iramos@frsf.utn.edu.ar

Abstract. Se parte de una pregunta desde la que se va a intentar, en este trabajo, obtener una primera respuesta o acercamiento: Dado un conjunto bien definido de patrones de seguridad ¿Son estos adecuados para repeler o mitigar un conjunto concordante de patrones de ataque?
Mediante un recorrido cronológico sobre la investigación desarrollada en materia de seguridad, previa y presente, se va a introducir cada concepto asociado a esta pregunta tanto así como la importancia del tratamiento de esta información —su catalogación—, las dificultades que presentan a la hora de llevarse a la práctica y los avances que se están haciendo respecto de esto último.

Keywords: Patrones de Ataque. Patrones de Diseño. Seguridad. Software.

1 Introducción

1.1 Etapas previas

Teniendo en cuenta la presente popularización del uso de sistemas de software de manera cotidiana y los cambios constantes en todo aspecto del software, la necesidad de inmediatez, de la salida temprana de un producto al mercado ha ido creciendo, con sus implicaciones sobre la completitud sus pruebas de software, flexibilizando el control de calidad. La existencia de vulnerabilidades en el software derivadas tanto del diseño como de la programación está causada por dos factores primarios: *complejidad* y *motivación*. Los desarrolladores de software empujan para crear productos cada vez más complejos, y se trabaja constantemente sobre el límite de complejidad manejable. Por otra parte, aunque los vendedores fueran capaces de crear un software más seguro, la economía de la industria de software provee poco incentivo para ello; los consumidores generalmente recompensan a los vendedores por añadir rasgos y ser los primeros en colocarlos en el mercado. Estas dos motivaciones están en tensión directa con el objetivo de producir un software más seguro.

Al momento de iniciar mi trabajo como becario de investigación en 2012 ingresé en un proyecto que ya presentaba continuidad en este apartado del software desde 2009 y donde se planteó ver a la seguridad como un aspecto transversal que recorre a todas las etapas del proceso de desarrollo de software y al mismo tiempo genera realimentación para proyectos sucesivos. Estos trabajos previos fueron demostrando progresivamente

que ha habido grandes avances tanto técnicos como de concientización sobre la seguridad en los sistemas pero están muy detrás en lo que respecta a la velocidad de cambio del software antes mencionada.

En dichos trabajos se analizó a la seguridad, no solo como un requerimiento de calidad integral sino también como un requerimiento a considerar en todas las fases del ciclo de vida del desarrollo de software

Se definieron actividades integrables al ciclo de vida: 1. Identificar objetivos de seguridad; 2. Aplicar guías de diseño de seguridad; 3. Crear modelos de amenazas; 4. Conducir revisiones de seguridad de la arquitectura y el diseño; 5. Completar revisiones de seguridad de la implementación; 6. Ejecutar revisiones de seguridad del despliegue [1].

Se probaron diferentes herramientas y metodologías a aplicar en cada etapa del desarrollo, se elaboró una conclusión que dio sustento teórico al proyecto bajo el cual ingresé y abrió las puertas a una vacante, espacio a profundizar y probar, dentro de los patrones de diseño.

1.2 Inicio como becario

La primera instancia de mi participación fue de preparación, en la que participé de la elaboración de un glosario de términos y en un estudio generalizado sobre el material asociado a la seguridad, y con mi inclusión se comenzó a profundizar en lo que son los patrones de seguridad como una alternativa viable para acercar la teoría de seguridad —abundante y compleja en muchos niveles— a la puesta en práctica.

2 Patrones de seguridad

Los patrones de seguridad (PS) son una metodología que capta la experiencia en soluciones implementadas por expertos de seguridad, que son abstraídas de contexto y descritas bajo un conjunto de especificaciones [2].

Lo que proponen ante las distintas etapas del ciclo de vida de desarrollo del software es evitar la implementación de soluciones propias, mediante la selección de un conjunto de patrones de seguridad ya definidos que sean contextualmente coherentes al entorno del sistema a construir, para luego realizar su profundización y aplicación, lo que puede marcar la diferencia en la defensa ante posibles amenazas. De esta manera no solo se puede tener un mejor manejo del dominio de la seguridad de un sistema sino que también se puede realimentar a las descripciones de los patrones para avanzar en conjunto en pos de mejores y más completas soluciones futuras.

Si bien cada patrón se focaliza en proveer una solución auto-contenida para resolver un problema específico, los patrones no son independientes unos de otros. Existen relaciones, y muchas veces la solución propuesta de cierto patrón se obtiene a partir de la implementación de un conjunto de otros patrones, de la misma manera en que un problema general es enmendado por la resolución de sub-problemas menores.

Los patrones de seguridad fueron definidos en múltiples trabajos con diferentes características y en diferente profundidad, sin haber “estándares” que definan su correcta descripción y difusión [3]. A fines de usar un ejemplo como aclaración, en la Tabla 1

se ve la catalogación de **un** patrón de seguridad simple, presente en el índice creado bajo este mismo proyecto y desarrollado más adelante.

Nombre	Account Lockout
Objetivo	Intento de captura de claves de usuario
Clasificación	Diseño
Aspecto de seguridad afectado	Confidencialidad, no-repudio
Palabras claves	Clave de usuario, contraseña, *guessing*
Referencia bibliográfica	Security Patterns Repository v1.0.pdf
	http://www.giac.org/paper/gsec/594/authentication-mechanisms-best/101431
	DoD 5200.28-STD, Trusted Computer System Evaluation Criteria. December 1985

Tabla 1. Ejemplo de catálogo de un Patrón de Seguridad típico, con sus Atributos asociados.

3 Proyecto de mapeo

Partiendo del trabajo anterior, se generó la propuesta comenzada en 2014, con el objetivo de analizar la contribución de los patrones para la generación de inteligencia asociada al desarrollo de software seguro.

Se distinguieron y refinaron un conjunto de criterios que ayudan a la producción de software seguro:

- Tener presente que la seguridad y el costo de desarrollo de un producto de software dependen fuertemente del conocimiento que se tenga de sus requisitos.
- Incluir el tratamiento de la seguridad en cada una de las diferentes etapas del proceso de desarrollo de software es un criterio aceptado para mejorar la seguridad del producto final [4] [5].
- Emplear los patrones de seguridad, ya que representan las mejores prácticas logradas por la industria, a fin de detener o limitar ataques que comprometen la seguridad del software [2] [3] [6].
- Emplear los patrones de ataque en la descripción de los métodos utilizados para la explotación del software.

Este proyecto pone foco en los últimos dos puntos dado que se considera que son los que más aportan a la captura de inteligencia de seguridad, de una manera sistemática y comunicable [7] [8].

4 Patrones de ataque

Los patrones de ataque (PA) constituyen también una herramienta de conocimiento para el diseño, desarrollo y despliegue de software seguro, que permite capturar y comunicar la perspectiva del atacante (completan la definición de una amenaza).

Conforman la descripción de los métodos utilizados para explotar software, orientados (desde el punto de vista de diseño) a un objetivo relativamente "destructivo" y que son un elemento clave para identificar y comprender las diferentes perspectivas en las que una amenaza se puede convertir en una vulnerabilidad. Ofrecen información que permite determinar factores de riesgo, evaluaciones de impacto, detección proactiva de ataques y apoyo en las decisiones de seguridad [9].

De esta manera, brindan una forma coherente de enseñar a diseñadores y desarrolladores como sus sistemas pueden ser atacados y como pueden efectivamente defenderse.

Podemos resumir los beneficios que se obtienen de su uso de la siguiente manera:

- Ayudan a categorizar los ataques de una forma significativa, haciendo posible un análisis efectivo de los problemas y las soluciones; en lugar de adoptar un enfoque ad-hoc para la seguridad del software, los PA permiten identificar los tipos de ataques conocidos a los que una aplicación puede estar expuesta y así construir en la propia aplicación las mitigaciones adecuadas;
- Contiene el nivel de detalle suficiente acerca de cómo tienen lugar los ataques de tal forma que los desarrolladores pueden ayudar a prevenirlos. Sin embargo, por lo general, no contienen detalles específicos inapropiados acerca de los exploits reales a fin de no favorecer el aprendizaje no deseado por parte de atacantes menos expertos. La idea de fondo es que no se los pueda utilizar directamente para crear exploits automatizados.

Los Patrones de ataque cumplen un rol entre tantos dentro del "conocimiento en seguridad de la arquitectura de software". Otros artefactos asociados pueden ser: casos de desuso/abuso, requerimientos de seguridad, modelos de amenazas, árboles de ataque, etc.

5 La catalogación de patrones

5.1 Catalogación de Patrones de Seguridad

Desde el proyecto de investigación se desarrolló un trabajo considerando la consigna de que la información ofrecida por un catálogo resulte apropiada para "encontrar" el patrón de seguridad, que fue presentado en [10]. Esta información se extrae o infiere de la descripción del propio patrón, e incluye extensiones que faciliten su categorización conforme a dos criterios establecidos: a) el o los atributos de seguridad impactados por el problema descrito; b) la fase del proceso de desarrollo de software a la que aplica el patrón.

El trabajo se orientó al ordenamiento y accesibilidad de esa inteligencia, buscando generar una propuesta de catalogación de los PS; para ello, se definió una forma de estructurar e indexar un catálogo, de manera de poder encontrar con cierta facilidad un

PS que proponga una solución a una situación de seguridad identificada, y de allí ir a la referencia original completa de la descripción del patrón de seguridad.

Los atributos comunes de los PS que se tuvieron en cuenta fueron: a) nombre; b) objetivo; c) clasificación (requisitos, análisis, diseño, codificación, pruebas, implementación), d) aspecto de seguridad afectado (confidencialidad, integridad, disponibilidad, responsabilización, no-repudio), e) referencia bibliográfica (enlaces hacia los documentos y/o páginas web donde se encuentra la descripción detallada del patrón).

5.2 Catalogación de Patrones de Ataque

En la actualidad, la iniciativa *Common Attack Pattern Enumeration and Classification* -CAPEC- (TM) ofrece un catálogo disponible en línea en forma pública de los PA, junto con un esquema de uso general para la definición de entidades y una taxonomía [9]. Asimismo, está disponible el listado completo y actualizado de las entradas. Se facilita la búsqueda en línea de un PA específico mediante una lista de palabras claves o el ID-CAPEC.

Además, a fin de facilitar la navegación, es posible seleccionar alguno de los métodos de revisión definidos: a) por representación jerárquica, b) por relaciones con factores externos o c) por relaciones con atributos específicos; cada uno de estos métodos ofrece una vista única que ayuda a encontrar un PA específico o que muestra las relaciones entre diferentes PA. También está disponible documentación con las descripciones de los diferentes elementos del *CAPEC Schema* oficial, que hace posible la comprensión de las estructuras de datos, y que se puede utilizar para el desarrollo de nuevas entradas en el catálogo o el agregado de contenido en entradas existentes.

6 Evaluación de la relación Patrones de Ataque / Patrones de Seguridad

Cuando se planifica la construcción de software seguro, se debería definir cuáles son los ataques que el sistema tiene que resistir y, por tanto, los requisitos de seguridad que el sistema tiene que cumplir. Dado este conjunto de posibles ataques, los arquitectos y desarrolladores deberían ser capaces de seleccionar patrones apropiados y considerarlos en el diseño del mismo. Este proceso se representa en la Figura 1.

En el sitio web antes mencionado de CAPEC [9], se presenta un catálogo de PA y se proporciona un árbol de clasificación de patrones de ataque, basado en el método STRIDE (*Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege*).

Las actividades a mi cargo consistieron en alinear los PA del catálogo CAPEC y sugerir una taxonomía basada en la descripción de los PA, el propósito e intención de los PS.

La ventaja de una taxonomía de este tipo es que los usuarios pueden ver fácilmente los PS pertinentes al identificar PA específicos, y ayuda a identificar PS similares.

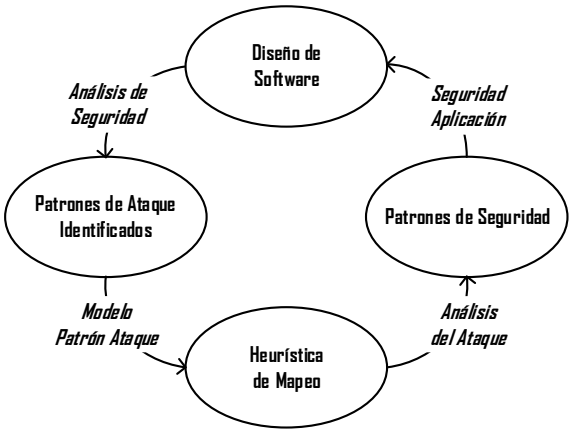


Fig. 1. Patrones en el diseño de software seguro.

Como primer acercamiento, si tenemos en cuenta los atributos definidos en el catálogo de patrones de ataque, existe un conjunto de ellos que puede —después de una etapa de estudio y correlación— relacionarse con patrones de seguridad.

Específicamente los atributos definidos "*Solutions and mitigations*", "*Relevant Security Requirements*", "*Related Security Principles*" y "*Related Guidelines*", definen vagamente la forma de encarar una solución para establecer controles de seguridad ante tal ataque, incluyendo qué principio de seguridad se vería violado y a qué directrices de seguridad afecta, todo esto simplemente enunciado. Si bien no se encuentra descripto pormenorizadamente, ayuda a contrastarlo con los atributos y características definidos en el catálogo de patrones de seguridad.

Para hacer más clara la forma de proceder podemos seleccionar un patrón de ataque de CAPEC que defina una amenaza probable en un hipotético sistema cliente-servidor, como puede ser la inyección SQL (CAPEC-66). Este PA se aprovecha del software que construye sentencias SQL basadas en cadenas ingresadas por el usuario. El atacante construye una cadena no esperada que ejecute acciones diferentes a las cuales la aplicación prevé.

Atributos definidos:
Solutions and mitigations: Validar de forma estricta las entradas, usar procedimientos almacenados, usar páginas de error personalizadas.
Relevant Security Requirements: Escapar los caracteres especiales, solo usar consultas almacenadas ya parametrizadas, revalidar la entrada de información en tales consultas y las páginas de error no deben mostrar información que ayude a posibles ataques.

Teniendo en cuenta lo definido en tales atributos, una búsqueda de palabras clave en el catálogo de patrones de seguridad desarrollado por equipo del proyecto de investigación, nos da un conjunto aplicable.
Como se ve en la Figura 2, una búsqueda en el catálogo por "validar" nos da tres patrones, de los cuales dos son PS de diseño y aplicables.

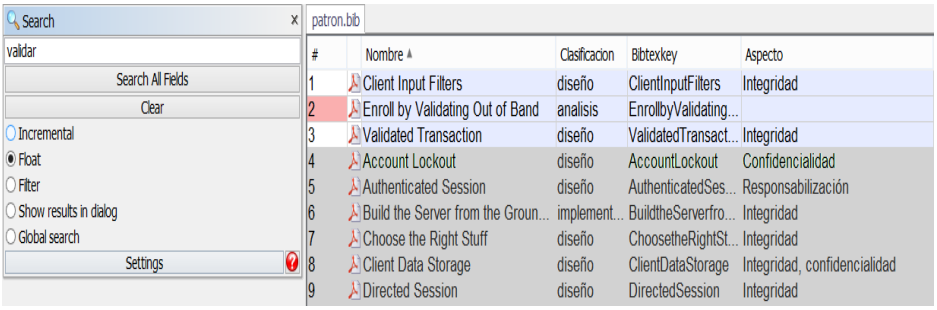


Fig. 2. Resultado de buscar "validar" en el catálogo de PS desarrollado, en azul los coincidentes

Si analizamos las entradas de los dos PS identificados para la fase de diseño, encontramos que ambos objetivos **(**Figuras 3 y 4**)** hacen referencia a posibles mitigaciones del ataque inicial, y ambos patrones poseen patrones relacionados que se pueden trabajar para lograr en conjunto una solución integral.

Los pasos a seguir pueden ser profundizar en cada uno para eliminar los que no aplican al contexto y/o arquitectura del sistema y estudiar sus aplicaciones prácticas ya definidas y probadas, en vez de inventar soluciones propias. De la misma manera se puede continuar con los patrones de ataque relacionados que define CAPEC y construir un árbol que relacione PA a PS aplicables.

Patron de seguridad **(ValidatedTransaction)**

Validated Transaction

(a.k.a. Mini-Pattern.)

Clasificacion: Patron de diseño

Abstract: The Validated Transaction pattern puts all of the security-relevant validation for a specific transaction into one page request. A developer can create any number of supporting pages without having to worry about attackers using them to circumvent security. And users can navigate freely among the pages, filling in different sections in whatever order they choose. The transaction itself will ensure the integrity of all information submitted.

Objetivo: Evitar la manipulación de información ante un conjunto de pasos críticos de validación mediante una re-validación completa en el paso final.

Aspecto: Integridad.

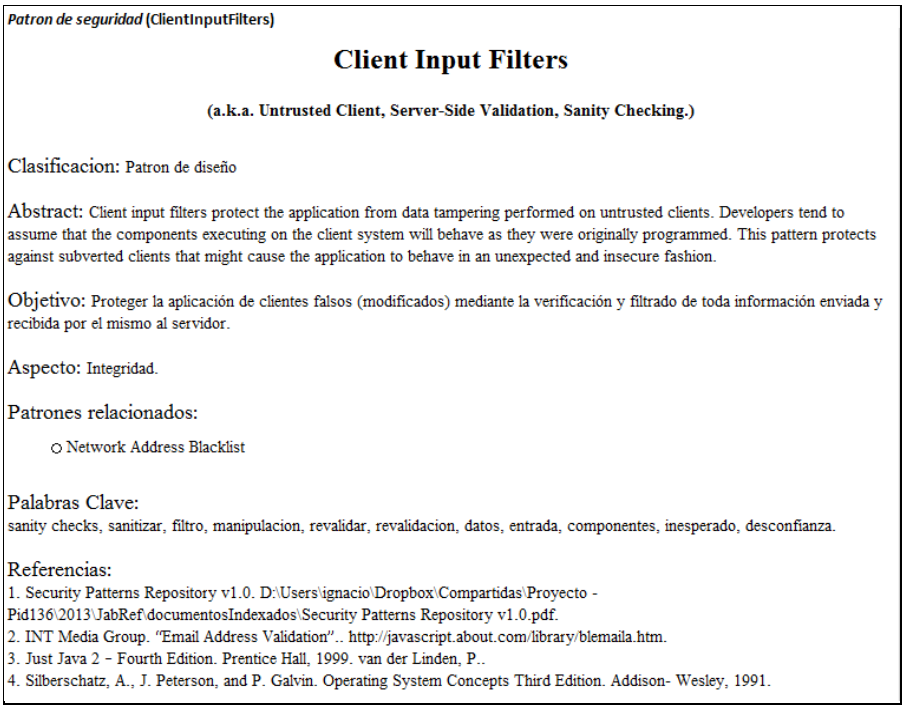
Patrones relacionados:

- Client Input Filters
- Directed Session

Palabras Clave:
Revalidar, Transaccion validada, mini patrón, ecommerce, comercio electrónico, critical steps, single point commit, validation check, comprobación.

Referencias:
1. Security Patterns Repository v1.0. D:\Users\ignacio\Dropbox\Compartidas\Proyecto - Pid136\2013\JabRef\documentosIndexados\Security Patterns Repository v1.0.pdf.

Fig. 3. Entrada de catálogo del Patrón *Validated Transaction*.



Patron de seguridad (ClientInputFilters)

Client Input Filters

(a.k.a. Untrusted Client, Server-Side Validation, Sanity Checking.)

Clasificacion: Patron de diseño

Abstract: Client input filters protect the application from data tampering performed on untrusted clients. Developers tend to assume that the components executing on the client system will behave as they were originally programmed. This pattern protects against subverted clients that might cause the application to behave in an unexpected and insecure fashion.

Objetivo: Proteger la aplicación de clientes falsos (modificados) mediante la verificación y filtrado de toda información enviada y recibida por el mismo al servidor.

Aspecto: Integridad.

Patrones relacionados:

- Network Address Blacklist

Palabras Clave:
sanity checks, sanitizar, filtro, manipulacion, revalidar, revalidacion, datos, entrada, componentes, inesperado, desconfianza.

Referencias:
1. Security Patterns Repository v1.0. D:\Users\ignacio\Dropbox\Compartidas\Proyecto - Pid136\2013\JabRef\documentosIndexados\Security Patterns Repository v1.0.pdf.
2. INT Media Group. "Email Address Validation".. http://javascript.about.com/library/blemaila.htm.
3. Just Java 2 - Fourth Edition. Prentice Hall, 1999. van der Linden, P..
4. Silberschatz, A., J. Peterson, and P. Galvin. Operating System Concepts Third Edition. Addison- Wesley, 1991.

Fig. 4. Entrada de catálogo del Patrón *Client Input Filters*.

7 Conclusiones y Trabajos Futuros

En los últimos años, las iniciativas han centrado su objetivo en relacionar los dos tipos de patrones, de manera que la aplicación de patrones de ataque se ligue a la aplicación de patrones de seguridad. Ha surgido una nueva taxonomía basada en PA con el fin de mejorar la aplicabilidad de los PS. Las diferentes clasificaciones de PS publicadas tienen desventajas y en general no son adecuados para los no expertos con el fin de seleccionar patrones en situaciones específicas, por ello se trabajó para establecer un esquema de clasificación de los PS que se basa en los PA [11].

Actualmente estamos ejecutando un proyecto de aplicación de estos modelos a sistemas basados en web, que tiene como objetivo general analizar la contribución de los patrones para la generación de inteligencia asociada al desarrollo de software seguro, y como objetivos específicos a) analizar la evolución en los últimos años de las formas de descripción y clasificación de los patrones de seguridad, b) identificar y comprender los conceptos, taxonomías y aspectos claves para la identificación y descripción de los patrones de ataque, c) localizar modelos y herramientas para la aplicación de los patrones en el tratamiento de los ataques sobre software, d) aplicar modelos y herramientas al estudio de casos, y e) elaborar pautas que ayuden a las organizaciones a incorporar PS y PA a sus procesos de desarrollo de software seguro.

En este trabajo futuro se debe considerar la validación de la correspondencia (mapeo PA/PS) y la utilidad de la clasificación, es decir, probar si un PS de diseño en realidad ayuda a impedir con éxito un ataque descrito por un PA. También es necesario comprobar si un no-experto será capaz de seleccionar adecuados patrones en situaciones específicas. Se propone trabajar con el enfoque detallado en [11] y reflejado en la Figura 5.

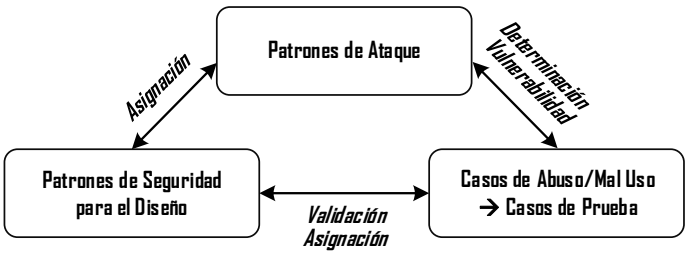


Fig. 5. Proceso de validación del mapeo PA/PS.

En este marco, y como parte del trabajo para una beca bianual que recientemente se me ha asignado, se elaboró un plan destinado a la construcción de una prueba de concepto que permita validar relaciones entre PS y PA sobre una aplicación web seleccionada.

Finalmente, expreso una conclusión personal de esta actividad como becario. Con muy pocos conocimientos de seguridad ingresé en el año 2012 a un equipo de trabajo, pero fui incrementando mis conocimientos y a la vez desarrollando habilidades asociadas a la investigación y la comunicación. En el año 2013 pude participar como expositor en un evento internacional (CIBSI). En año 2014 trabajé en el proyecto PID "Uso de patrones para la inteligencia de seguridad" donde desarrollé las actividades que se describen en este trabajo. Además fui incorporando lo visto a mis actividades laborales. El tema despertó en mí mucho interés y he solicitado una beca del gobierno nacional bianual, en el marco de este equipo de investigación, que será de apoyo para la finalización de mi carrera.

Por último expreso mi reconocimiento a los docentes investigadores que me guiaron: Marta Castellaro, Susana Romaniz, Juan Carlos Ramos.

8 Referencias

1. Ozment, A., "Software Security Growth Modeling: Examining Vulnerabilities with Reliability Growth Models". Quality of Protection Springer. (2006). Disponible http://pdf.aminer.org/000/220/165/modelling_software_operational_reliability_via_input_domain_based_reliability_growth.pdf.
2. Schumacher, M. et al., "Security Patterns: Integrating Security and Systems Engineering". John Willey & Sons Inc. EEUU. (2006).
3. Schumacher, M.: "Security engineering with patterns-origins, theoretical model, and new applications". Springer-Verlag. (2003).

4. Castellaro, M. y otros, “Hacia la Ingeniería de Software Seguro”. XV Congreso Argentino de Ciencias de la Computación, CACIC 2009. Argentina. (2009).
5. Romaniz, S. y otros, “La seguridad como aspecto organizacional y transversal en proyectos de Sistemas de Información”. 38 Jornadas Argentinas de Informática 38JAIIO. Argentina. (2009).
6. Meier, J. et al., “Security engineering explained”. (2005). Disponible en http://www.microsoft.com/download/en/confirmation.aspx?id=20528.
7. Yoshioka, N. et al.: “A survey on security patterns”. Progress in Informatics. No. 5 pp. 35–47, (2008).
8. Bunke, M. et al.: “Application Domain Classification for Security Patterns”. PATTERNS 2011: The Third International Conferences on Pervasive Patterns and Applications Copyright (c) IARIA, (2011).
9. CAPEC Common Attack Pattern Enumeration and Classification: “CAPEC List Version 2.0”, Desarrollado en forma comunitaria, disponible públicamente. (2013). Disponible en http://capec.mitre.org/.
10. J. Ramos, S. Romaniz, M. Castellaro e I. Ramos. “Compartir Inteligencia: Construcción de un Catálogo de Patrones de Seguridad”. CIBSI2013. (2013) Disponible en: http://www.cibsi.utp.ac.pa/documentos/Catalogo_de_Patrones_de_Seguridad.pdf.
11. Wiesauer, Sametinger: A security design patterns taxonomy based on attack patterns; Findings of a Systematic Literature Review. International Conference on Security and Cryptography-SECRYPT (2009). Paper 83