¿Qué hay respecto a atender a la Seguridad durante el desarrollo de sistemas de información?

Castellaro Marta, Romaniz Susana, Ramos Juan Carlos
Universidad Tecnológica Nacional, Facultad Regional Santa Fe

Abstract.
"Esta comunicación presenta una reseña del trabajo realizado por un grupo de docentes investigadores de la Facultad respecto al Software Seguro. En distintos proyectos de investigación se fue abordando el tema en forma evolutiva, partiendo de la consigna de analizar la Seguridad como atributo organizacional transversal en los proyectos de Sistemas y Tecnologías de Información. Se analizó la taxonomía propia, y la adaptación de los procesos de ciclo de vida de desarrollo, mejorados para la seguridad. Se indagaron metodologías y herramientas para abordar la problemática de la seguridad, especialmente en las etapas tempranas del ciclo de desarrollo. También se trabajó sobre Patrones de Seguridad, elaborando una propuesta para ampliar su accesibilidad a los desarrolladores. Hoy se propone un trabajo enfocado en los Patrones de Ataque".

Palabras Clave
Software seguro, Seguridad, Patrones, Ciclo de Vida

Conceptos de Software Seguro

La seguridad siempre fue concebida como una propiedad deseable del software y forma parte del conjunto de atributos a considerar para determinar la calidad del mismo. El aumento de los ataques a aplicaciones vulnerables condujo al reconocimiento gradual por parte de la comunidad de especialistas, empresas e industria del software que las protecciones a nivel de infraestructura (hoy de uso común para proteger sistemas accesibles por Internet) no son suficientes para garantizar un comportamiento seguro de las mismas. En la actualidad, se reconoce que las protecciones y mitigaciones de seguridad de las aplicaciones se deben especificar a nivel del software y su arquitectura, e implementar tanto durante su proceso de desarrollo, y su despliegue y operación.

La seguridad se presenta cada vez más como un factor crítico a ser considerado por las organizaciones que proyectan la incorporación de Tecnologías de la Información (TI) o desarrollan e implementan Sistemas de Información (SI). Pero la modalidad generalmente empleada en las organizaciones para el tratamiento de la Seguridad en proyectos de TI y SI aborda la problemática de este atributo de forma fragmentada, estableciendo compartimentos estancos, en distintas áreas y en diferentes momentos durante la implementación e integración de proyectos; y en muchos casos a posteriori de su implementación [1].

La falta de una visión integral y transversal genera situaciones que implican mayores costos (muchos de ellos innecesarios), mayor trabajo, re-trabajo, rechazos de los participantes y, en muchos casos, hasta se incorporan nuevas vulnerabilidades.

En necesario plantear los objetivos de seguridad en el contexto de una *estrategia de la seguridad*, directamente ligados a los objetivos de la organización y alineados a casos de negocio, se los considere desde el inicio de los proyectos de SI, y que además evolucionen de manera lo más efectiva y eficientemente posible acompañando las transformaciones de la organización que los anida. En los últimos años la situación ha evolucionado, y la seguridad del software ya se considera como una característica que se la puede demostrar de manera consistente [2] [3].

Antes de determinar las características de un software para hacer de éste un *software seguro*, se deben establecer cuáles son los problemas de seguridad (criticidad del dominio de aplicación del software, vulnerabilidades derivadas de tecnologías utilizadas, *exploits* identificados) que se deberán traducir en requerimientos para ser atendidos durante su proceso de desarrollo.

Es necesario definir los atributos mediante los cuales se puedan describir estas características. Los ***Atributos de Seguridad del Software*** comprenden:

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

Centrando el análisis en los atributos fundamentales, se considera que los efectos de vulnerar la seguridad del software se pueden describir en términos de los efectos sobre estos atributos fundamentales, los cuales se enumeran a continuación [4]:

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 a 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 provocada 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 atributos adicionales, generalmente asociados 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; garantiza que no se puede subvertir o eludir el atributo 'responsabilización'.

En la Figura 1 se representan estos atributos:

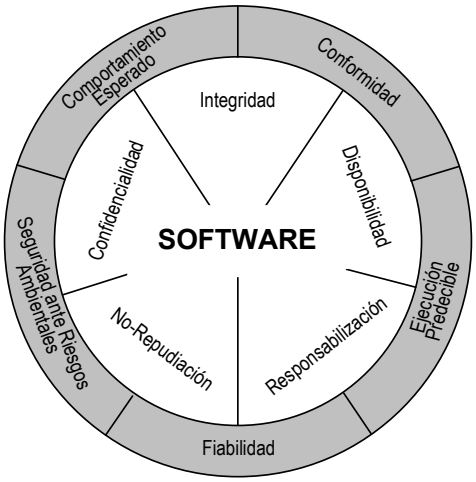

Figura 1. Propiedades fundamentales y conducentes en un software seguro.

La seguridad a lo largo de todo el ciclo de vida

A pesar de las medidas de seguridad perimetral adoptadas las aplicaciones siguen siendo muy vulnerables. Existen ya buenos argumentos para dar por agotada esta visión de la seguridad. Por una parte, plantear el aislamiento para protegerse va en sentido opuesto con los modelos de negocio de las empresas, que cada vez más tienden a la comunicación y colaboración con clientes y otros actores del negocio. Por otra parte, el objetivo de un posible atacante puede ser vulnerar la red, el host, o la aplicación, lo que exige para la prevención de estas amenazas un conjunto de defensas montadas en profundidad. No es que se deban descartar las medidas de seguridad perimetral, sino que se las debe considerar

sólo como una de las capas de un sistema de seguridad.

Existen diferentes definiciones acerca del *software seguro* establecidas por expertos y organizaciones reconocidas, pero todas ellas presentan un criterio común: "La seguridad se presenta en todo el ciclo de vida del software. Un software seguro se diseña, implementa, configura y opera de manera que cumpla con las propiedades esenciales, es decir, se espera que continúe operando en presencia de ataques, aísle o limite los daños y se recupere lo antes posible".
El software que ha sido desarrollado considerando a la Seguridad generalmente presenta las siguientes propiedades a lo largo de su ciclo de vida:
Ejecución predecible: existe la confianza justificada respecto a que, cuando se ejecuta, funciona de la manera esperada; está significativamente reducida o eliminada la habilidad de *inputs* maliciosos destinados a alterar la ejecución o de resultados producidos de forma que son favorables para el atacante.
Fiabilidad: se minimiza en el mayor grado posible el número de vulnerabilidades explotables de manera intencional; la meta es ninguna vulnerabilidad explotable.
Conformidad: actividades planificadas, sistemáticas y multidisciplinarias que aseguran que los componentes, productos y sistemas de software conforman los requerimientos y los estándares aplicables, y los procedimientos para los usos especificados.

Las vulnerabilidades inherentes al software (debido a errores o fallas en su diseño arquitectónico, en el código o en los algoritmos que implementan) no se pueden resolver completamente con medidas de seguridad externas. Dicho en otras palabras, una parte importante de los problemas de seguridad de los sistemas de información tiene su origen en defectos del software que éstos ejecutan, que se introducen durante su proceso de desarrollo; de allí, la importancia que se los pueda evitar y/o detectar durante dicho proceso.
El desarrollo de un software que pretenda considerar mejoras respecto a la seguridad implica ciertas prácticas y pruebas a realizar, y la adopción de enfoques y principios acordes en las distintas etapas del ciclo. Esto además exige un abordaje integrado, a lo largo de todas las etapas del ciclo de vida.

Resulta necesario analizar los modelos y métodos del ciclo de vida del desarrollo de software. Desarrollar software desde su inicio con la atención puesta en la Seguridad es mucho más efectivo, en órdenes de magnitud, que tratar de demostrar que el software es seguro, mediante pruebas y validación. *Un proceso de ciclo de vida de desarrollo mejorado para la seguridad -SDLC-* (al menos en algún grado) garantiza el cumplimiento de los requerimientos del software mediante el agregado de prácticas con las que se verifica su adecuación durante todas las fases del ciclo de vida del software. La idea que sustenta estas metodologías establece que se apliquen las mejores prácticas de seguridad del software a un conjunto de artefactos de software creados durante su proceso de desarrollo. Se propone, en la mayoría de los casos, que resulte un proceso neutral y, en consecuencia, se pueda utilizar con una amplia variedad de modelos de procesos de desarrollo.
Los controles de seguridad en el ciclo de vida del software no se deberán limitar a las fases de requerimientos, diseño, codificación y testeo. Resulta importante continuar realizando revisiones de código, pruebas de seguridad, controles de configuración, y aseguramiento de la calidad durante el despliegue y la operación, para garantizar que las actualizaciones y los parches no agregan vulnerabilidad de seguridad o lógica maliciosa al software en producción.
A lo largo de los últimos años, se han realizado diferentes esfuerzos para definir el modelo de proceso de desarrollo de

software más efectivo. En algunos casos, introduciendo modificaciones en las actividades tradicionales, mientras que otros insertando nuevas actividades, con el objetivo de reducir el número de vulnerabilidades en el software y en los servicios que se apoyan en él. En la actualidad, ya existen metodologías que están siendo empleadas con distintos grados de éxito en proyectos de desarrollo de productos o pilotos; ejemplos de éstas son:

Microsoft Trustworthy Computing SDL. Orientado a reducir las vulnerabilidades relacionadas a seguridad en el diseño, codificación y documentación, detectar y eliminar vulnerabilidades tan pronto como sea posible; y reducir la severidad del impacto de cualquier defecto residual [5].

Comprehensive, Lightweight Application Security Process (CLASP). Diseñado para insertar metodologías de seguridad en cada fase del ciclo de vida. Es un conjunto de actividades enfocadas en seguridad que pueden ser integradas en cualquier proceso de desarrollo de software [6].

Siete '*touchpoints*' para Seguridad de Software. Un conjunto de mejores prácticas a ser aplicadas en varios artefactos del desarrollo de software [7].

Estas metodologías se pueden comparar adoptando un conjunto de fases comunes a la mayoría de los SDLC [1]. De ello surge un conjunto común de artefactos indicados para la ejecución de muchas de las actividades propuestas, que se presentan en la Figura 2.

Modelos y Herramientas

Actualmente, se dispone de modelos y herramientas que dan soporte al abordaje de la problemática de la seguridad del software durante el Ciclo de Vida de Desarrollo.

En función del alcance definido para el trabajo, se centró la investigación en las etapas tempranas del proceso de desarrollo de software: Requerimientos, y Análisis y Diseño [8].

El **Modelado de Amenazas** (*Threat Modeling* - TM) se presenta como una herramienta para evaluar los riesgos inherentes a una aplicación durante su desarrollo, principalmente en la etapa de diseño. Constituye un *framework* específico para el proceso de análisis de riesgo estructurado, que permite identificar las amenazas de una aplicación y cuantificar los riesgos a los que la misma estará expuesta. Es una metodología integrante del *Trustworthy Computing Security Development Life Cycle* de Microsoft, implementada desde hace más de una década [7]. Se trata de un esquema estructurado y repetible, para analizar riesgos y amenazas

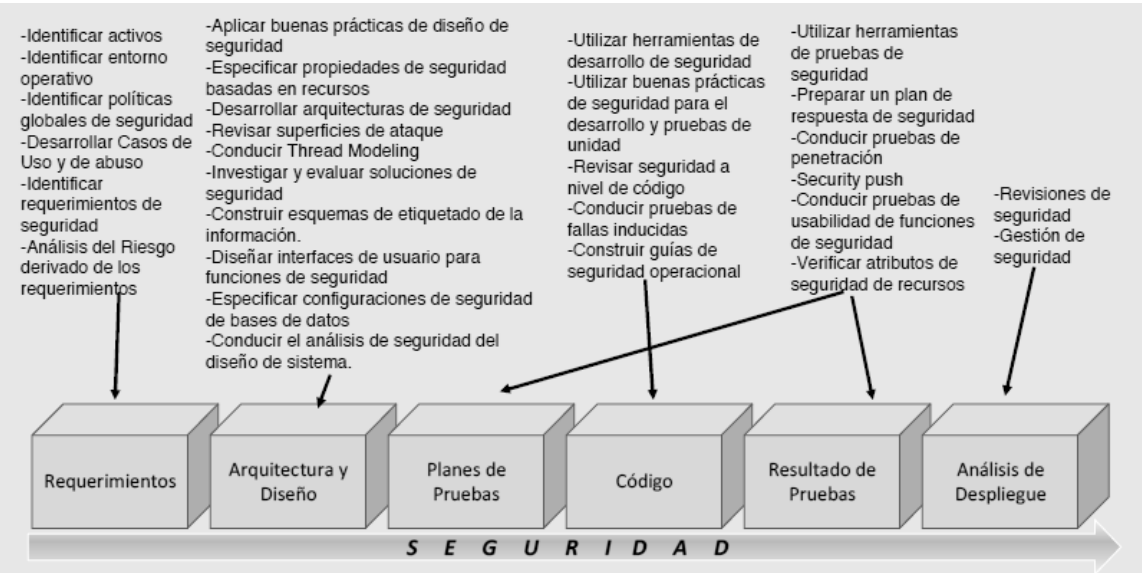

Figura 2. La seguridad a lo largo del Ciclo de Vida

en el software, que ha ido evolucionando a lo largo del tiempo, y su implementación se puede adaptar a diferentes proyectos u organizaciones, y la pueden utilizar equipos de desarrollo que no cuenten con expertos en seguridad.
TM aporta elementos al diseño que definen rasgos claves del futuro sistema: controles de acceso, mecanismos de autenticación, tolerancia y recuperación de fallas, por mencionar sólo algunos artefactos y atributos. No es una metodología fija o estática sino que la puede abordar desde diferentes enfoques:
- "*centrado en el activo*" (implica algún nivel de evaluación de riesgo, aproximación o clasificación),
- '*centrado en el atacante'* (implica la clasificación de riesgo o intenta estimar recursos, capacidades o motivaciones, y conlleva determinar escenarios de amenazas),
- '*centrado en el software*' (análisis de software, redes o sistemas de las organizaciones y conlleva el modelado de protocolos y de amenazas de red).
El enfoque elegido dependerá de los objetivos establecidos para el proyecto y del tipo de software que se desee obtener; si se está modelando un software de misión-crítica, el cuidado estará puesto en la protección de los activos a los que accede; en este caso, se dará preponderancia a un enfoque centrado en activos. En otro tipo de proyectos (por ejemplo, software de base) no se sabrá qué activos manipulará el software, y el acento estará en lograr una arquitectura y código seguros (por ejemplo bibliotecas del sistema operativo) dando preponderancia a un enfoque centrado en el software.

El proceso de Modelado de Amenazas comprende ***seis etapas*** [9]:
1) Conformar un grupo de análisis de riesgos (encargado de llevar adelante el modelado).
2) Descomponer el software e identificar componentes clave (incluyendo en este análisis las redes subyacentes y el software de base, para crear un 'perfil de seguridad' (vulnerabilidades conocidas).
3) Determinar las amenazas sobre cada componente (las propias y las derivadas de su interacción con otros componentes).
4) Asignar un valor a cada amenaza (calcular el riesgo asociado a cada amenaza, mediante alguna técnica elegida para tal fin).
5) Decidir cómo responder a las amenazas (eliminar la funcionalidad, no hacer nada y aceptar el riesgo, mitigarla mediante alguna contramedida, transferir el riesgo a una tercera parte).
6) Identificar las técnicas y tecnologías necesarias para mitigar los riesgos identificados (restricciones arquitectónicas, técnicas criptográficas, mecanismos de autenticación, algoritmos seguros, etc., que permitan solucionar los problemas planteados).

Para la etapa 3, se puede emplear el **Método STRIDE** (*Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, Elevation of privilege*).
Este método se aplica por cada elemento de la aplicación identificado en la etapa 2, analizando a cuáles de las categorías de amenazas mencionadas es sensible cada componente clave.
Las amenazas son el objetivo que persigue el atacante; considerando este enfoque, se puede emplear el **Método Análisis de Árboles de Amenazas**, que permite analizar en forma descendente y partiendo de una amenaza que el atacante puede concretar, las distintas alternativas que puede adoptar dicha amenaza para alcanzar ese objetivo; de esta manera, resulta posible identificar los puntos que se deben asegurar para mitigar la amenaza.
Para la etapa 4, se puede utilizar el **Método DREAD** (*Damage, Reproducibility, Exploitability, Affected, Discoverability*).

El estudio de la metodología llevó a investigar qué ***herramientas*** se encontraban disponibles para gestionar la aplicación del modelo. Se encontraron dos aplicaciones

provistas por Microsoft, que fueron concebidas para cubrir necesidades de diferentes equipos de desarrollo de la propia compañía, pero que a la fecha en que se llevó a cabo la investigación, estaban disponibles para descarga en forma gratuita:
- ***SDL Threat Modeling Tool*** (SDL) [10], utilizada en el desarrollo de productos como Windows Vista o SQL Server, que se clasifican como software de base.
- ***Threat Analysis and Modeling Tool*** (TAM) [11], orientada al desarrollo de aplicaciones de negocio, en las cuales están claramente definidos los objetivos de negocio, el patrón de despliegue, los datos y los controles de acceso.

En la Tabla 1 se presenta una comparación de las características de las dos herramientas. Del análisis, queda en claro que no hay una sola forma correcta de llevar adelante un modelado de amenazas sino que se pueden adoptar distintos enfoques válidos según las necesidades del equipo de trabajo y las características del producto a desarrollar.
En cuanto al equipo de análisis, *TAM tool* proporciona un repositorio de ataques y contramedidas; esto facilita el proceso de asociar cada amenaza con un conjunto de estas últimas. Por su parte, con *SDL tool* es el analista quien tiene que determinar las mitigaciones adecuadas para una amenaza, basándose en su propio conocimiento, por lo que es indispensable la presencia de expertos en seguridad.
En [8] se presentan los resultados de aplicar TAM a un Caso de Estudio: un sistema de catálogos por Internet, que pretende tener los elementos básicos de este tipo de portales Web. En la Figura 3 se muestra la captura que presenta la vista de árbol de los elementos del modelo que describen la aplicación y que son requeridos para el análisis de las amenazas.

Un posterior refinamiento de los casos de uso concluye la visión arquitectónica necesaria para definir el contexto de la aplicación. La herramienta genera automáticamente las amenazas a partir de las llamadas o interacciones definidas en cada caso de uso. Por ejemplo, para el CU "Autenticación del Usuario" de la Figura 3, a partir de la interacción entre un usuario anónimo que envía sus credenciales al Servidor Web, se generará una amenaza que luego se deberá analizar.
La herramienta provee vistas de los campos de información relativos al análisis de la amenaza, algunos de los cuales están destinados para las siguientes etapas del proceso de desarrollo del software.

Aplicación	**SDL Threat Modeling Tool**	**Threat Analysis and Modeling Tool**
Enfoque	Centrado en software	Centrado en activos
Características de las aplicaciones a desarrollar	Aplicaciones para las que el patrón final de despliegue no es conocido	Aplicaciones con objetivos de negocios definidos, patrones de despliegue, y datos bien conocidos
Objetivos	Garantizar la seguridad del código subyacente del software	Determinar el riesgo asociado a la aplicación, y las medidas necesarias para gestionar el riesgo
Técnicas para definir la arquitectura	Diagramas de Flujo de Datos	Diagramas de casos de uso y secuencia, descripción de componentes y roles
Presencia de Analistas expertos en seguridad	Muy necesaria	Puede modelarse sin la presencia de expertos
Identificación de amenazas	STRIDE	Basado en atributos de los componentes, roles y datos

Tabla 1. Comparativo SDL - TAM

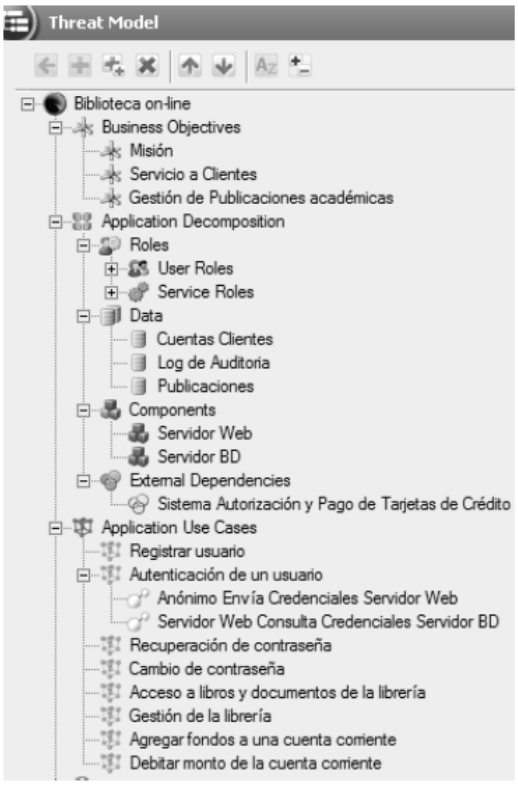

Figura 3. Vista de caso implementado en TAM

Patrones de Seguridad

En la última etapa del trabajo realizado por el equipo de investigadores, ganó importancia una línea de investigación orientada al uso de Patrones de Seguridad. En la Ingeniería de Software generalmente se está familiarizado con el concepto de patrones relacionados a la arquitectura de los sistemas o soluciones para su diseño de los mismos. No obstante, el concepto de patrón es más general, y se pueden encontrar diferentes tipos, como por ejemplo patrones de procesos, patrones de diseño, patrones de codificación, y patrones de seguridad. En los últimos años, diferentes autores han desarrollado patrones de seguridad para las actividades más importantes de cualquier ciclo de vida de desarrollo de software: requerimientos, análisis, diseño, codificación, y distribución; quienes, además han establecido clasificaciones y han generado repositorios para los patrones de seguridad conocidos [12].

En una primera aproximación al concepto de patrón, se lo puede definir como una solución a un problema que emerge en un contexto específico, la cual debe estar probada y ser repetible. En otras palabras, cada patrón describe un problema que surge frecuentemente en un cierto entorno, y presenta una solución general a ese problema, la cual puede utilizarse todas las veces que sea necesario, adaptando la solución para cada caso en particular.

La multiplicidad de fuentes desde donde se encuentran disponibles los diferentes grupos de patrones de seguridad definidos en la actualidad, hace que su descubrimiento demande un nivel de esfuerzo que, normalmente, desalienta el uso sistemático por parte de los potenciales destinatarios.

A partir de la identificación de este problema, se decidió trabajar en la definición y desarrollo de un *catálogo centralizado*, bajo la premisa que éste constituye una herramienta que actúa como punto de partida para la búsqueda e identificación de una o más soluciones a un problema de seguridad que se pretende resolver, expresadas mediante patrones de seguridad. De esta manera se busca establecer un puente entre la inteligencia desarrollada por expertos en seguridad y las necesidades de conocimiento de los equipos de desarrollo de software.

Para ello se definió, como principal requerimiento, que la información ofrecida por el catálogo resulte apropiada para "encontrar" el patrón de seguridad. Esta información se extrae o infiere de la descripción del propio patrón, e incluye extensiones que faciliten la categorización del patrón conforme criterios establecidos. En el actual diseño del catálogo se han adoptados *dos criterios*:

- el o los *atributos de seguridad* impactados por el problema descripto,
- la *fase del proceso de desarrollo de software* a la que aplica el patrón.

De esta manera, se espera que el catálogo ofrezca información acerca del patrón de seguridad que ayude a capturar de manera inmediata sus aspectos esenciales. Entonces, el problema a resolver es definir

una forma de estructurar e indexar un catálogo de patrones de seguridad, de manera de poder encontrar con cierta facilidad un patrón 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. Como a*tributos comunes* de los patrones de seguridad que nos permitan encontrar sus referencias, se definió el conjunto indicado en la Tabla 2.

Nombre	El nombre de la definición original del patrón de seguridad.
Objetivo	Problema que resuelve el patrón de seguridad. Es una respuesta a un problema de seguridad presente.
Clasificación	En base a la fase del proceso de desarrollo de software en la que normalmente se aplica el patrón: requerimientos, análisis, diseño, codificación, pruebas, implementación
Aspecto de seguridad afectado	Confidencialidad, integridad, disponibilidad, responsabilización, no-repudio
Palabras claves	Sirven como una referencia complementaria al patrón.
Referencia bibliográfica	Enlaces hacia los documentos y/o páginas web donde se encuentra la descripción detallada del patrón.

Tabla 2. Atributos para la descripción de patrones de seguridad.

Salvo el atributo 'Nombre', para los restantes atributos es necesario hacer un análisis de la descripción del patrón de seguridad en su fuente original, extraer los conceptos asociados a los atributos elegidos, y efectuar la catalogación del patrón.

Con respecto a este *proceso de extracción de conceptos* durante la revisión de los patrones, se encontró una guía en [14]; en este trabajo se resumen los resultados acerca de la *calidad de la información* incluida en los aspectos con que se describen los patrones, así como la frecuencia de uso de dichos aspectos utilizados a lo largo de 67 publicaciones de patrones de seguridad. Al respecto, se pueden destacar las consideraciones siguientes como guía para el proceso de extracción:

- *Solution* (presente en un 87% de las publicaciones), se utiliza para describir cuáles aspectos de seguridad atiende el patrón y cómo se lo podría implementar;
- *Problem* (presente en un 84% de las publicaciones), en muchos casos incluye descripciones abstractas o excesivamente simplificadas del problema.
- *Related Patterns* y *Consequences* (presentes en un 75% de las publicaciones), requieren de un buen conocimiento tanto de otros patrones de seguridad como del posible impacto en el campo de la seguridad para que se puedan utilizar como elementos de distinción.
- *Context* (presente en un 49% de las publicaciones), incluye generalmente una descripción muy breve, lo que hace difícil extraer el conocimiento suficiente o incluso una idea acerca de lo que es el patrón de seguridad.
- *Known Use* (presente en un 46% de las publicaciones), describe en qué casos de la vida real se puede utilizar el patrón y orienta respecto su dominio de aplicación.

Un ejemplo de aplicación del proceso propuesto se presenta aplicándolo al patrón *Authenticated Session* [15]. Dicho proceso de extracción consiste en la lectura y análisis de la descripción del patrón de seguridad a cargo de un experto de seguridad, y en la obtención de los datos indicados en la Tabla 3, los cuales resultan de las siguientes referencias (estas últimas

Nombre	**Authenticated Session**
Objetivo	Permitir el acceso a múltiples páginas con acceso restringido, sin tener que re-autenticarse cada vez. Mantener información de autenticación en la negación de un sistema
Clasificación	Diseño
Aspecto de seguridad afectado	Integridad. Responsabilización
Palabras claves	Authentication. Single Sign-On
Referencia bibliográfica	• Security Patterns Repository v1.0.pdf • Cunningham, C. "Session Management and Authentication with PHPLIB". http://www.phpbuilder.com/columns/chad19990414.php3,(Rev. Mayo 2013). • Kärkkäinen,S. "Session Management". *Unix Web Application Architectures*. http://webapparch.sourceforge.net/#23, October 2000. (Rev. Mayo 2013)

Tabla 3. Datos para la catalogación del patrón de seguridad *Authenticated Session*.

dependerán del formato particular adoptado por los autores para la descripción del patrón de seguridad y de la calidad de la información asociada):

- Objetivo: se infiere de la sección *Abstract* de la descripción.
- Clasificación: el experto de seguridad que realiza la lectura y análisis del patrón infiere que el mismo aplica a la Fase de Diseño, en el aspecto de 'autenticación de usuarios'.
- Aspecto de seguridad afectado: en el párrafo final de la sección *Problem* de la descripción hace referencia a que "*es fácil cometer errores cuando se aplican mecanismos de sesiones a situaciones donde responsabilización, integridad, y privacidad son críticas.*"; en consecuencia, el patrón procura resolver estas falencias.
- Palabras claves: se infieren de las secciones *Name* y *As-Know-As* (a.k.a).
- Referencia bibliográfica: a partir de las referencias incluidas en la sección *References*, se seleccionan aquellas fuentes cuya disponibilidad al momento de la catalogación se puede verificar.

Para concretar esta propuesta, analizamos herramientas alternativas de uso extendido que nos permitieran su implementación. Decidimos construir un prototipo utilizando una herramienta para gestión de referencias bibliográficas: 'JabRef'[16]. Ésta es una alternativa de muy bajo costo para probar la idea, que, si efectivamente funciona, se puede luego extender a herramientas de uso masivo, como por ejemplo una aplicación web con características 'wiki'.

JabRef es un software de gestión de referencias bibliográficas configurable. Utiliza como formato nativo BibTex (un formato de archivo basado en texto e independiente del estilo) para definir listas de elementos bibliográficos, artículos, tesis, etc. El formato determinado por BibText permite dividir cada tipo de entrada (*Entry Types*: Artículo, Tesis, Manual, etc.) en una serie de parámetros regulares a todas ellas (*Fields*: como nombre, autor, publicación) de forma que a la hora de su escritura o lectura podamos centrarnos en aquellos que conozcamos y al mismo tiempo dar una organización básica a los elementos mínimos y opcionales de toda entrada.

Para la generación del catálogo propuesto se aprovechó esta facilidad para el registro de referencias en este gestor; además, la existencia de los atributos distinguibles y específicos en cada patrón permitió generar un *entry type* especial (del tipo Patrón de Seguridad) con sus respectivos campos e información general y/o opcional.

La Figura 4 se presenta una pantalla de la interface de JabRef, donde:

- *Grupos* (*Groups*) (Sección 1). Se crear*on* grupos que corresponden a las fases del desarrollo de software en la

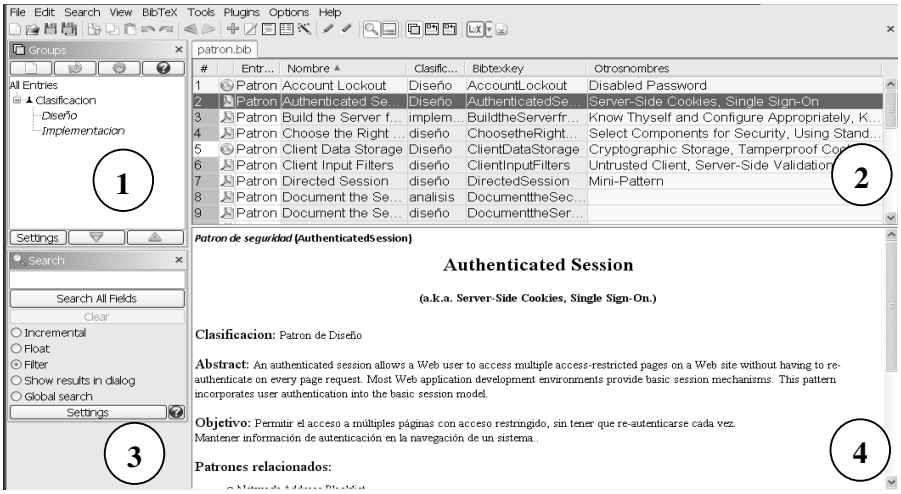

Figura 4. Interface del catálogo en JabRef.

Seleccionando un grupo, en la Sección 2 (*Entry*) aparecerán los patrones que correspondan a esa clasificación.

- *Principal (Entry)* (Sección 2). Las entradas cargadas en la base se presentan como una tabla con los campos como columnas, las que pueden utilizarse para seleccionar un criterio de orden al presionar la columna deseada. Se pueden ver todas las referencias bibliográficas o a archivos relacionados haciendo click derecho en la fila deseada de la columna "*File*" (segunda columna), plegándose una lista de valores. Si selecciona uno de estos, se dirigirá a la referencia solicitada, sea este un documento disponible localmente o internet.
- *Búsqueda (Search)* (Sección 3). Permite buscar de acuerdo a un atributo y filtros dentro de todo el catálogo. Luego de una búsqueda, y ya mostrando la se lección de patrones que contienen la frase o palabra buscada, al hacer doble click sobre estos y navegando por las diferentes pestañas, las palabras buscadas aparecerán resaltadas en azul.
- *Vista Preliminar (Sección 4).* Se modificó la versión original para que muestre por defecto la información del patrón seleccionado en formato PDF, que es una representación más amigable del contenido del patrón. Presionando el botón derecho de mouse se puede imprimir la vista preliminar.

En esta figura se muestra la manera como se visualiza la información catalogada relativa al patrón *Authenticated Session* [15], donde se observan los atributos propuestos para categorizar un patrón de seguridad.

Modelo de Madurez en la Seguridad

Cabe destacar el desarrollo de abordajes de Aseguramiento de la Calidad (QA), para las prácticas de desarrollo de sistemas y de software seguro. Se deben respetar las pautas del QA en las principales actividades relacionadas con la Seguridad durante el ciclo de vida de desarrollo a los fines de proveer las garantías de que los requerimientos de seguridad se han definido adecuadamente, desarrollado los controles de satisfacción de la seguridad, y satisfecho los requerimientos de seguridad.
En del año 2009 se publicó el modelo de madurez "*Building Security In Maturity Model* (BSIMM) diseñado para ayudar a entender y planificar una iniciativa de seguridad de software. Se trata de un modelo creado por un proceso de entendimiento y análisis de datos de un conjunto de iniciativas de seguridad de software destacadas: CLASP de OWASP, Microsoft SDL, o *Touchpoints* de Cigital),. Durante dicho proceso se relevaron y analizaron datos de estas experiencias, habiéndose identificado que, si bien las metodologías particulares aplicadas se diferencian, existen muchos puntos en común en estas iniciativas. Estos puntos son capturados y descritos en BSIMM.

BSIMM no es una guía completa de "cómo manejar" la seguridad de software, sino una colección de buenas prácticas y actividades que están en uso actualmente.
El modelo se apoya en el *Software Security Framework*, que proporciona un andamio conceptual para BSIMM. Correctamente usado, BSIMM puede le ayuda a una organización a compararse en lo que concierne a iniciativas mundiales de seguridad de software, y definir los pasos que puede tomar para lograr un abordaje más eficaz en pos de la producción de software seguro.
Dicho *framework* reconoce Doce Prácticas organizadas en Cuatro Dominios:
Gobernanza: Aquellas prácticas que ayudan a organizar, manejar y medir una iniciativa de seguridad de software. El desarrollo de personal es también una práctica de central.
Inteligencia: Las prácticas que recolectan el conocimiento corporativo usado en realizar actividades de seguridad de software a lo largo de toda la organización. Las colecciones incluyen tanto la guía proactiva de seguridad como el modelado de amenaza de la organización.

Puntos de contacto con el SSDL: Prácticas asociadas con el análisis y el aseguramiento de artefactos de desarrollo de software particulares y procesos.
Despliegue: Prácticas que hacen a lo que realizan las organizaciones con la configuración y mantenimiento del software, y otras cuestiones del ambiente que tienen el impacto directo sobre la seguridad de software.
El modelo de madurez identifica para cada Práctica, Tres Niveles de madurez, los que proporcionan una progresión natural a lo largo de las actividades asociadas con cada práctica. No obstante, no resulta necesario realizar todas las actividades para un nivel dado antes de avanzar con las actividades en un nivel superior dentro de la misma práctica.
De esta manera, mediante la identificación de los Objetivos y las Actividades correspondientes a cada Práctica que se aplique, y garantizando un balance apropiado con respecto a los dominios, es posible crear un plan estratégico para cada iniciativa de software seguro en particular.

Acciones relacionadas
Además de las actividades directamente vinculadas con la ejecución de diferentes proyectos de investigación, cabe destacar otras acciones derivadas de los avances logrados en el dominio del software seguro.
En el año 2012, se codirigió una tesis de Maestría (Conti J., "Estudio e implementación de una biblioteca que permita detectar vulnerabilidades en aplicaciones web escritas en Python mediante el análisis de manchas")[17]. En dicha tesis, se elaboró una técnica destinada a evitar ataques por inyección de código malicioso en aplicaciones Web, que se puede integrar a *frameworks* de codificación segura. De esta manera, resultó posible abordar aspectos relacionados a la etapa de Codificación, con algunos aportes para esta fase del SDLC.
Por otra parte, durante el año 2013, integrantes del equipo de docentes investigadores han llevado a cabo pruebas pilotos para evaluar la madurez de la seguridad en el proceso de desarrollo de software en organizaciones públicas y privadas localizadas en la zona de la ciudad de Santa Fe. En esta primera experiencia, se hizo uso de la metodología BSIMM adaptada al ámbito local.

Conclusión
La seguridad en los sistemas de información es una preocupación constante en todas las organizaciones que basan sus estrategias de negocio en el uso de los mismos, y sigue siendo un problema latente en todas ellas. Los diferentes avances en el tratamiento de soluciones al problema de la seguridad en el software procuran proveer de soluciones a un tema complejo. Como se pudo observar a lo largo de los diferentes proyectos de investigación llevados adelante, son múltiples los aspectos que se deben considerar desde la concepción de un producto de software, hasta su puesta en marcha y uso cotidiano. Asimismo, se identificó que existen diferentes aspectos que se deben considerar para atacar este problema y, también, distintas soluciones a los mismos [18].
En la última etapa, se focalizó el interés en los patrones de seguridad, y cómo estos pueden ser utilizados como un instrumento para compartir conocimiento de expertos en seguridad [19]. Atendiendo a este último aspecto, se ha trabajado en el desarrollo de un prototipo que facilite su descubrimiento y acceso por parte de la comunidad, de manera que puedan ser aprovechados en el trabajo cotidiano. Uno de los próximos objetivos a encarar es evolucionar dicho prototipo hacia una solución más completa como soporte a los Patrones de Seguridad'.
Con tal propósito, se elaboró un proyecto de investigación para desarrollar en los próximos dos años, en el cual además, se propone articular el uso de los patrones de seguridad con los Patrones de Ataque, otro instrumento que permite capturar y modelar los problemas de seguridad a los

que se ven expuestos los sistemas en general, y el software en particular.

Además, se elaboró otro proyecto de investigación para desarrollar en el mismo período, cuyo objetivo es ampliar y actualizar la base de conocimiento acerca de los modelos de madurez y marcos de referencia para la producción de software seguro existentes y herramientas que los soportan, así como evaluar su adecuación al contexto nacional. De esta manera, se espera realizar nuevos aportes en el campo del desarrollo de software seguro, en este caso orientados a facilitarle a las empresas que desarrollan software tanto para comercializar como para uso interno introducir mejoras en la calidad respecto de la seguridad en sus productos.

Referencias

[1] Romaniz S., Castellaro M., Ramos J. "La Seguridad como aspecto organizacional y transversal en Proyectos de Sistemas de Información". 38 JAIIO - Workshop Seguridad Informática. Mar del Plata, Argentina. 2009.

[2] Romaniz S., Castellaro M., Ramos J., Feck C., Gaspoz I., "Aplicar el Modelo de Amenazas para incluir la Seguridad en el Modelado de Sistemas". V Congreso Iberoamericano de Seguridad Informática-CIBSI 2009, Montevideo, Uruguay. 16 a 18 de noviembre de 2009. Trabajos Completos- ISBN: 978-9974-0-0593-8, Pag 118.

[3] Goertzel K., Winograd T., McKinley H., Oh L., Colon M., McGibbon T., Fedchak E., Vienneau R. "Software Security Assurance -State-of-the-Art Report (SOAR)", Information Assurance Technology Analysis Center (IATAC). 2007.

[4] Avizienis, A et al.: "Basic and Concepts and Taxonomy of Dependable and Secure Computing". IEEE Transactions on Dependable and Secure Computing. Vol.1 No.1. (2004).

[5] Lipner S., Howard M. "El ciclo de vida de desarrollo de seguridad de Trustworthy Computing". Unidad tecnológica y empresarial de seguridad-Microsoft Corporation, 2004 Annual Computer Security Applications Conference, copatrocinada por IEEE. USA. 2004.

[6] CLASP. www.securesoftware.com/CLASP

[7] McGraw, G. Software Security: Building Security In. Addison-Wesley. 2006.

[8] Feck C. "Modelado de Amenazas, una herramienta para el tratamiento de la seguridad en el diseño de sistemas". 4º Congreso Nacional de Estudiantes de Ingeniería en Sistemas de Información (CNEISI 2010). Santa Fe. http://www.frsf.utn.edu.ar/cneisi2010/archivos/22-Modelado_de_Amenazas_en_el_diseño_de_sistemas.pdf

[9] Milano P., "Seguridad en el ciclo de vida del desarrollo de Software", Seminario de Seguridad en el Desarrollo de Software, CYBSEC Security Systems, Argentina, 27 de agosto de 2008.

[10] "The Microsoft SDL Threat Modeling Tool", Microsoft,http://www.microsoft.com/security/sdl/getstarted/threatmodeling.aspx, accedido junio de 2010.

[11] ACE (Assessment, Consulting & Engineering) Team,"Threat Analysis and Modeling v3.0", Microsoft,http://code.msdn.microsoft.com/tam/Release/ProjectReleases.aspx?ReleaseId=4180, accedido junio de 2010.

[12] Ramos J., Romaniz S., Castellaro M.. "Patrones de seguridad aplicados a la función autorización". Anales del XVIII Congreso Argentino de Ciencias de la Computación CACIC 2012- Workshop de seguridad informática (WSI), organizado por Red de Universidades con Carreras en Informática (RedUNCI) - http://sedici.unlp.edu.ar/handle/10915/23841. 2012. ISBN 978-987-1648-34-4.Bahía Blanca. Argentina.

[13] Fortify Software. [http://www.fortify.com/]

[14] Cigital. [http://www.cigital.com/]

[15] Darrell M. Kienzle et al. "Security Patterns Repository Version 1.0". Disponible en http://www.scrypt.net/~celer/securitypatterns/repository.pdf

[16] http://jabref.sourceforge.net/

[17] Conti J., Russo A.; " A Taint Mode for Python via a Library"; OWASP Europe AppSec Research 2010; Estocolmo Suecia; http://www.owasp.org/index.php?title=OWASP_AppSec_Research_2010_-_Stockholm,_Sweden&setlang=es#A_Taint_Mode_for_Python_via_a_Library

[18] Castellaro M.; Pessolani P.; Ramos J.; Romaniz S. "Hacia la Ingeniería de Software Seguro". XV Congreso Argentino de Ciencias de la Computación – CACIC, Jujuy, Argentina . 2009. Trabajos Completos- ISBN: 978-897-24068-4-1 . Volumen WIS, pág 1257

[19] Ramos J.; Romaniz S.; Castellaro M.; "Patrones de Seguridad aplicados a un Sistema de Toma de decisión". Proceeding del Simposio del IIITEC Simposio Internacional de Innovación y Tecnología ISIT2011-ISBN: 978-612-45917-1-6, Pág. 43-48. Perú. 2011.

Datos de Contacto

Marta Castellaro. Universidad Tecnológica Nacional - Facultad Regional Santa Fe. Lavaise 610 (S3004EWB) Santa Fe. Argentina. mcastell@frsf.utn.edu.ar.