Catalogación como Apoyo al Uso de Patrones de Seguridad

Susana Romaniz, Juan Carlos Ramos, Marta Castellaro, Ignacio Ramos

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

Abstract. 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. La 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. Los patrones de seguridad 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 probada, permitiendo una transferencia eficiente de experiencia y de conocimientos. La descripción de un patrón debe ayudar a capturar de manera inmediata su esencia: cuál es el problema al que atiende y cuál es la solución propuesta. Los diferentes formatos existentes para su descripción y la multiplicidad de fuentes donde se encuentran disponibles, hacen que su descubrimiento demande esfuerzo que desalienta el uso sistemático por parte de los potenciales destinatarios. En este trabajo se presenta el prototipo de un catálogo que busca establecer un puente entre el conocimiento y la experiencia desarrollados por expertos en seguridad y las necesidades de conocimiento de los equipos de desarrollo de software.

Keywords: Patrones de seguridad. Inteligencia de seguridad. Catálogo de patrones de seguridad.

1 La seguridad en el proceso de desarrollo de software

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 [4]: 1) Identificar objetivos de seguridad, 2) Aplicar guías de diseño de seguridad, 3) Crear modelos de amenazas, 4) Conducir revisiones seguridad de la arquitectura y el diseño, 5) Completar revisiones de seguridad de la implementación, y 6) Ejecutar revisiones de seguridad del despliegue.

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 [5].

Paralelamente, observamos la activa evolución de herramientas y métodos para la realización de pruebas que permiten evaluar la robustez y resiliencia de productos de software y su infraestructura subyacente bajo condiciones de ataque (por ejemplo, los empleados en pruebas de penetración). Es decir, existe dominio de la inteligencia de ataques que explotan las vulnerabilidades y comprometen la seguridad de los sistemas basados en el uso intensivo de software. Todo esto produce un escenario complejo de abordar.

No obstante, existen criterios que ayudan a la producción de software seguro [6]:

- Tener presente que la seguridad y el costo de producción de un sistema de software dependen fuertemente del conocimiento que se tenga de sus requerimientos [7].
- Incluir el tratamiento de la seguridad en cada una de las diferentes etapas del ciclo de desarrollo de software es un criterio aceptado para mejorar la seguridad del producto final.[8]
- Incorporar los patrones de seguridad, ya que representan las mejores prácticas logradas por la industria a fin de detener o limitar ataques a la seguridad [9].

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.

Podemos inferir, entonces, que promover el uso de patrones de seguridad para que guíen y conduzcan en todo momento la construcción de modelos de desarrollo de soft-

ware seguro, partiendo de etapas tempranas del proceso, es un criterio que nos permitirá garantizar una mayor seguridad en el comportamiento de un producto de software.

2 Los atributos de seguridad del 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 los atributos mediante las que puedan ser descriptas estas características. Estos atributos 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 un software.

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. [10]

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 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; asegura que no se puede subvertir o eludir el atributo ‘responsabilización’.

3 Los 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 [11] 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 [12]. 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 fases de requerimientos, análisis, diseño, codificación, pruebas o 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 [13] "*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 [14]. De esta forma existen patrones para la fase de requerimiento, patrones para la fase diseño y patrones para la fase de implementación.

Dado que los objetivos de los patrones de seguridad son los de centrarse en la arquitectura de alto nivel y las cuestiones referentes a las organizaciones, al emplear el enfoque de patrón en todos los niveles resulta posible extender el abordaje incluyendo a la seguridad en una sola estructura común y terminología, y ayuda a la integración de todos los niveles. En términos generales, orientar el análisis mediante el contexto de patrones de seguridad permite integrar el problema que se aborda y las fuerzas presenten que lo determinan.

Específicamente, se destacan las siguientes ventajas en el empleo de un enfoque orientado al uso de patrones de en el tratamiento de la seguridad:

- expresan los conocimientos básicos de seguridad de una manera estructurada y comprensible.
- su representación es familiar para los desarrolladores de software y los ingenieros de sistemas, una parte clave de su audiencia.
- los patrones ya se utilizan para capturar el conocimiento acerca del sistema y de la organización, al utilizarlos para capturar conocimientos de seguridad ayuda a

mejorar la integración de la seguridad en los ciclos de vida de los sistemas, donde resultado claramente necesario.

En lo que respecta particularmente a su desarrollo, en lo últimos años se han ido especificando diferentes grupos de patrones de seguridad, como así también se han realizado esfuerzos para su clasificación [15, 16].

3.1 Descripción de los patrones de seguridad

Frecuentemente se hace referencia al modelo POSA (*Pattern-Oriented Software Architecture*) desarrollado por Buchman y otros [17] para describir el contexto y el uso de los patrones de seguridad. Pero también se puede observar la existencia de diferentes modelos descriptivos [18]; incluso, en muchísimos casos se los describe de manera que no se respeta estrictamente dicho modelo. Estos son analizados en detalle en [19].

Con respecto al formato adoptado para la descripción de los patrones de seguridad, en general, se adopta la definición de una *plantilla*, que incluye un conjunto de elementos con nombre y un alcance definido. Una buena descripción ayuda a capturar de manera inmediata la esencia de un patrón, es decir, cuál es el problema al que atiende y cuál es la solución propuesta. Además, ofrece todos los detalles necesarios para implementarlo y considerar las consecuencias de su aplicación.

Es importante tener en cuenta que no todos los campos son obligatorios. Por ejemplo, en el caso de algunos patrones resulta dificultoso o innecesario proporcionar descripciones detalladas de su estructura, el comportamiento y la implementación, porque toda la información se puede integrar bien en la descripción de la solución.

Así mismo, en [20, 21] se han definido otros formatos para la descripción de patrones de seguridad, también basados en plantillas definidas como conjuntos de elementos con nombre y alcance asociado. En un análisis preliminar de los diferentes tipos de plantillas, se observa que no existe una concordancia exacta entre los elementos y su alcance, si bien todos ellos capturan aproximadamente los mismos aspectos del patrón de seguridad para describirlo.

En [19] presentan los resultados obtenidos respecto de la variabilidad de los aspectos empleados para la definición formal de los patrones de seguridad, los que se obtuvieron a partir de una exhaustiva revisión de 364 patrones de seguridad recopilados desde diferentes fuentes, los que fueron publicados durante los años 1997-2012. A partir de dichos resultados, surge que los aspectos que más frecuentemente se emplean en las diferentes descripciones proporcionan las pistas más significativas para la selección y organización de los patrones de seguridad.

Dicha revisión además nos permitió observar la muy alta heterogeneidad en la manera de denominar los aspectos y que no existen criterios de correspondencia entre los diferentes nombres, tal como existe entre las descripciones de POSA y GoF [22]. Con respecto a los aspectos incluidos en las descripciones revisadas, se identificaron los nombres de los elementos más significativos, los cuales podremos utilizar para lograr

una visión del patrón de seguridad y poder compararlos entre sí, a los fines de establecer relaciones entre los mismos.

4 Una vía para capturar "inteligencia"

Si bien ya no se niega la necesidad de abordar el problema de la seguridad en los productos y servicios basados en software, los responsables de proyectos aún enfrentan serias dificultades en atender de manera efectiva las prioridades identificadas en cuanto a la seguridad. ¿Cuál es la causa de esta discrepancia? No podemos decir que sólo se deba al desconocimiento respecto de la existencia de atacantes que logran explotar vulnerabilidades existentes en el software. Aún así, frecuentemente los requerimientos de seguridad se expresan de manera vaga, y las arquitecturas de software demuestran importantes limitaciones en las decisiones adoptadas al respecto. De hecho, podemos observar una importante desconexión entre los expertos en seguridad y los equipos de desarrollo de software; los primeros están enfocados en la seguridad de un sistema, mientras que los segundos en la construcción de un sistema. Para estos últimos, la seguridad es uno de los objetivos no-funcionales con los guardan relación, pero sólo uno más de muchos.

Los patrones de seguridad constituyen una tecnología adoptada para *la descripción, la comunicación y la compartición de conocimiento*, y se proponen como un puente que busca reducir esta brecha al capturar la experiencia en seguridad en la forma de soluciones verificadas para problemas recurrentes. Se espera que sean comprendidos y utilizados por los diferentes integrantes de los equipos de desarrollo que no son expertos en seguridad. Debido a que el énfasis está puesto en seguridad, estos patrones capturan las fortalezas y debilidades de los diferentes enfoques con el fin de permitirles a los equipos de desarrollo tomar decisiones informadas que equilibren la seguridad con otros objetivos.

Debemos destacar que en este trabajo hemos adoptado el concepto *inteligencia* para hacer referencia al *conjunto de prácticas que dan lugar a una recopilación de conocimiento corporativo, el que se utiliza para llevar a cabo de manera proactiva las actividades de seguridad del software en toda la organización.* El empleo de patrones de seguridad, o de metodologías y herramientas derivadas, constituye una vía para la adquisición de conocimiento probado y reutilizable, lo que actualmente se considera como una buena práctica en un proceso de desarrollo de software seguro.

A modo de ejemplo, podemos mencionar a *Building Security In Maturity Model* (BSIMM) [23], un modelo que se ha venido desarrollando desde el año 2009 en base a estudios de iniciativas existentes de seguridad del software. Este modelo reúne aquellas prácticas observadas en el proceso de desarrollo de software de muchas organizaciones, y que están destinadas a atender los requerimientos de seguridad, y describe la base común compartida. BSIMM incluye un grupo de prácticas dentro de lo que denomina el *Dominio Inteligencia*, uno de los cuatro definidos.

Nos propusimos trabajar en el ordenamiento y accesibilidad de esa inteligencia, buscando generar una propuesta de catalogación de los patrones de seguridad. Así pues hemos considerado de especial importancia la posibilidad de contar con recursos que permitan acceder de manera sistematizada a un repositorio donde se referencie al conjunto de patrones de seguridad definidos. De esta manera, esperamos poner a disposición de los diferentes actores intervinientes en el proceso de desarrollo de software (líderes de proyecto, arquitectos, diseñadores, responsables de QA, programadores, testers) este conocimiento extraído del mundo real a partir de la experiencia de especialistas en seguridad.

5 La catalogación de patrones de seguridad

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.

Un *catálogo centralizado* 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, definimos 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 hemos 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, esperamos que el catálogo ofrezca información acerca del patrón de seguridad que ayude a capturar de manera inmediata sus aspectos esenciales.

6 Propuesta

Nuestro problema entonces 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, definimos el conjunto indicado en la Tabla 1.

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.

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 1. Atributos para la descripción de patrones de seguridad.

Con respecto a este *proceso de extracción de conceptos* durante la revisión de los patrones, encontramos una guía en [19]; 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, podemos destacar las siguientes consideraciones 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.

7 Proceso de catalogación.

A continuación presentamos mediante un ejemplo, la descripción del proceso de incorporación de un patrón de seguridad al catálogo. El patrón seleccionado se denomina *Authenticated Session* [24], el que reproducimos en forma resumida en la Figura 1.

Este proceso 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 2, los cuales resultan de las siguientes referencias (estas últimas dependerán del formato particular adoptado por los autores para la descripción del patrón de seguridad y de la calidad de la información asociada):

Authenticated Session

(a.k.a. Server-Side Cookies, Single Sign-On)

Abstract

An authenticated session allows a Web user to access multiple access-restricted pages on a Web site without having to re-authenticate on every page request. Most Web application development environments provide basic session mechanisms. This pattern incorporates user authentication into the basic session model.

Problem

HTTP is a stateless, transaction-oriented protocol. Every page request is a separate atomic transaction with the Web server. But most interesting Web applications require some sort of session model, in which multiple user page requests are combined into an interactive experience.

As a result, most Web application environments offer basic session semantics built atop the HTTP protocol. And the protocol itself has evolved to provide mechanisms—such as basic authentication and cookies—that allow session models to operate correctly across different Web browsers.

An obvious use for session semantics is to allow users to authenticate themselves once instead of every time they access a restricted page. However, great care must be taken when using session semantics in a trusted fashion. Most session mechanisms are perfectly adequate for tracking non-critical data and implementing innocuous transactions. In such cases, if an end user circumvents the session mechanism, no harm is caused. But it is easy to make mistakes when applying session mechanisms to situations where accountability, integrity, and privacy are critical.

Solution

An authenticated session keeps track of a user's authenticated identity through the duration of a Web session. It allows a Web user to access multiple protected pages on the Web site without having to re-authenticate him-/herself on every page request. It keeps track of the last page access time and causes the session to expire after a predetermined period of inactivity.

……..

Examples

Many significant Web banking and e-commerce applications rely on this pattern. Any site that enforces user authentication and does not store that information on the client uses something similar.

……..

Related Patterns

· *Network Address Blacklist* – a related pattern that demonstrates a procedure for blocking a network address from further access attempts if a session identifier guessing attack is conducted.

· *Password Authentication* – a related pattern that presents the secure management of passwords, which are almost always used as the authentication mechanism for this pattern.

References

[1] Coggeshall, J. "Session Authentication".
http://www.zend.com/zend/spotlight/sessionauth7may.php, May 2001.
[2] Cunningham, C. "Session Management and Authentication with PHPLIB".
http://www.phpbuilder.com/columns/chad19990414.php3?page=2, April 1999.
[3] Kärkkäinen, S. "Session Management". *Unix Web Application Architectures*.
http://webapparch.sourceforge.net/#23, October 2000.

Fig. 1. Patrón de seguridad seleccionado para catalogar: *Authenticated Session* [24].

- 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.

Nombre	**Authenticated Session**
Objetivo	Permitir el acceso a múltiples páginas con acceso restringido, sin tener que re-autenticarse carda vez. Mantener información de autenticación en la navegación de un sistema.
Clasificación	Diseño
Aspecto de seguridad afectado	Responsabilización, Integridad.
Palabras claves	Autenticación, 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 2. Datos para la catalogación del patrón de seguridad *Authenticated Session* [24].

Obtenidos estos datos, procedemos a la catalogación del patrón de seguridad seleccionado para el ejemplo, haciendo uso del prototipo que describimos en la siguiente la Sección.

8 Prototipo

Para concretar esta propuesta, analizamos herramientas alternativas de uso extendido que nos permitieran su implementación. Definimos construir un prototipo utilizando una herramienta para gestión de referencias bibliográficas: 'JabRef'. 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, Patente, 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 aprovechamos esta facilidad para el registro de referencias en este gestor; además, la existencia de los atributos distinguibles y específicos en cada patrón nos permite generar un *entry type* especial (del tipo Patrón de Seguridad) con sus respectivos campos e información general y/o opcional.

Otra características importante de JabRef es el uso de LaTeX , el cual nos permite transferir automáticamente y sin mayores dificultades (dado el uso de parámetros bien definidos) el catálogo actual a cualquier otro tipo de software que permita el ingreso del mismo lenguaje: bases de datos, páginas de Wikipedia, otro gestor específico o de propósito general, etc.

La Figura 2 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 cual el patrón es normalmente aplicable. 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-

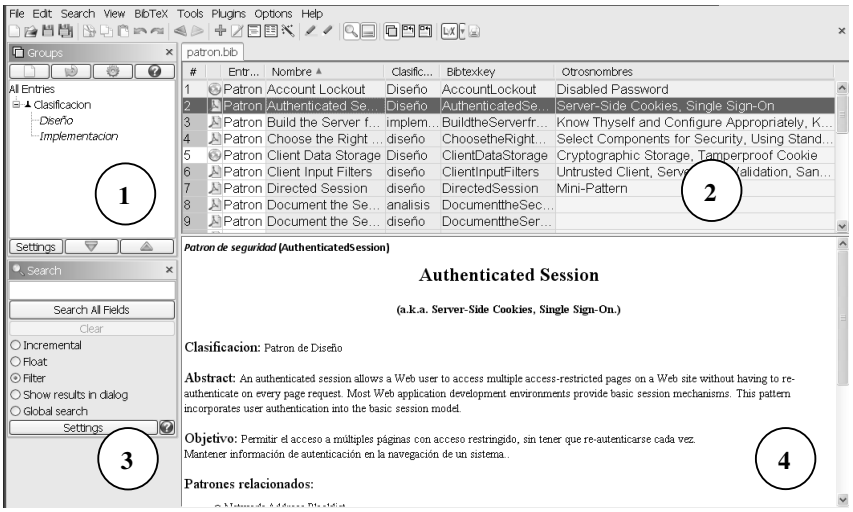


Fig. 2. Interface del catálogo en JabRef.

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* [24], donde se observan los atributos propuestos para categorizar un patrón de seguridad.

8.1 Ejemplo de catalogación de un patrón de seguridad

La Figura 3 muestra algunas de las facilidades que ofrece el prototipo desarrollado para incorporar información de los atributos indicados en la Tabla 2:

- Figura 3.a): Carga de los 'Campos requeridos' (obligatorios): nombre, objetivo, clasificación, aspectos (de seguridad), campos claves (keywords) y Bibtexkey (una particularidad de JabRef y referencias bibtex, que se aprovecha para referencias al patrón de seguridad),
- Figura 3.b): Se complementa con información de patrones relacionados y otros nombres conocidos (en ambos casos, si los hubiere),
- Figura 3.c): Se definen las 'Palabras Claves' (keywords) y las 'Referencias' al patrón de seguridad, aprovechando la facilidad de JabRef de poder vincular archivos locales y extrenos como 'Files'.

8.2 Ejemplo de búsqueda en el catálogo de patrones de seguridad

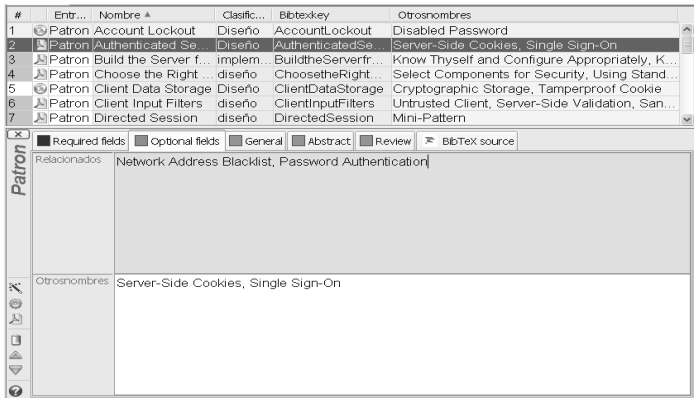
La Figura 4 muestra el resultado de una búsqueda por referencias a patrones de seguridad aplicables a la fase de diseño y que atiendan a la confidencialidad como atributo de seguridad.

8.3 Detalles de la implementación

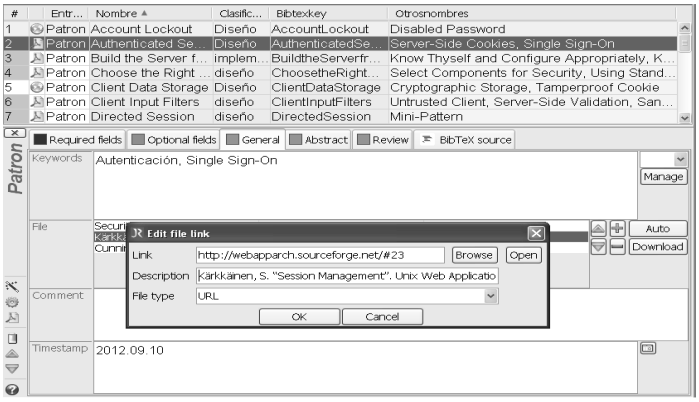
Para poder compartir la base de patrones documentados, se realiza una configuración de JabRef que debe incluir los siguientes elementos:

- JabRef.exe.
- jabref.xml (Configuración completa del software).
- JabRef-2.8.1.jar.
- patron.bib (la biblioteca generada).
- documentosIndexados (carpeta en la que JabRef va a buscar los archivos referenciados).

a) Campos Requeridos.



b) Campos Adicionales.



c) Carga de las 'Palabras Claves' y 'Referencias'.

Fig. 3. Interfaces para la carga de los datos correspondientes a los atributos de catalogación del patrón de seguridad *Authenticated Session* [24].

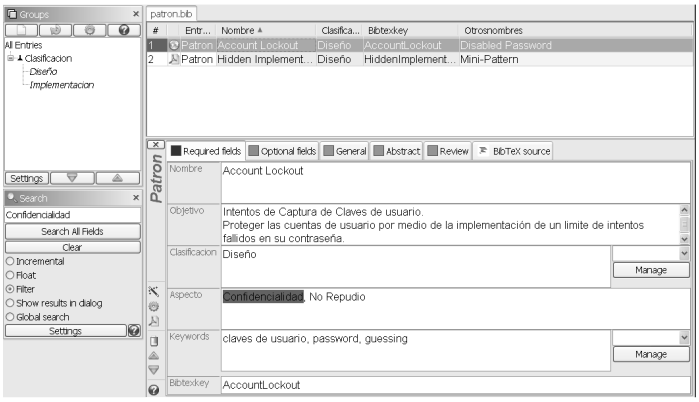


Fig. 4. Resultado de una búsqueda en el catálogo de patrones de seguridad.

• Conclusiones y Trabajo futuro

Los productos de software seguro plantean la necesidad de atender a atributos de seguridad en todas las fases de su proceso de desarrollo, y los problemas a resolver deben ser abordados con las soluciones adecuadas. Los patrones de seguridad son soluciones probadas a problemas recurrentes, y consideramos que disponer de una referencia rápida a los mismos será de suma utilidad para la comunidad de desarrollo de software.

Nuestro objetivo es compartir inteligencia sobre patrones de seguridad. Esta propuesta tiene como finalidad proveerle a toda esta comunidad de una referencia rápida, e ir realimentándola a medida que sea utilizada. La herramienta elegida para construir el catálogo de referencias nos sirve para probar de una manera simple, y eventualmente modificar, la propuesta de estructura y criterios de búsqueda, y consideramos la posibilidad de reemplazarla por otra con mayores prestaciones y estilo. Este prototipo nospermite definir las bases para hacer una extensión hacia una herramienta web del estilo wiki, la que podrá estar disponible para toda la comunidad, y además ser actualizada por ella.

Además de la herramienta web, será necesario proponer criterios para incorporar y publicar nuevas referencias al catálogo, de manera que todas las entradas sigan estos criterios comunes acordados. Este es el trabajo que nos proponemos encarar en una próxima etapa.

9 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”. (2005). Disponible en http://www.microsoft.com/download/en/confirmation.aspx?id=20528.
4. Castellaro, M. y otros, “Hacia la Ingeniería de Software Seguro”. XV Congreso Argentino de Ciencias de la Computación, CACIC2009. Argentina. (2009).
5. Ozment, A., “Software Security Growth Modeling: Examining Vulnerabilities with Reliability Growth Models”. Quality of Protection Springer. (2006). Disponible en http://pdf.aminer.org/000/220/165/modelling_software_operational_reliability_via_input_domain_based_reliability_growth.pdf
6. Solinas, M., “Elicitación y trazabilidad de requerimientos utilizando patrones de seguridad”. Universidad Nacional de La Plata. Argentina. (2012). Disponible http://sedici.unlp.edu.ar/bitstream/handle/10915/421/Documento_completo.pdf?sequence=1.
7. Garvin, D., “What does product quality really mean?”, Sloan Management Review, Vol 26, No 1. (1984).
8. 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).
9. Schumacher, M. et al., “Security Patterns: Integrating Security and Systems Engineering”. John Willey & Sons Inc. EEEUU. (2006).
10. Avizienis, A et al.: “Basic concepts and taxonomy of dependable and secure computing”. IEEE Transactions on Dependable and Secure Computing. Vol.1 No.1. (2004).
11. Coplien, J.: “Design Pattern Definition - Software Patterns”. Disponible en http://www.hillside .net/component/content/article/50-patterns/222-design-pattern-definition.
12. Schumacher, M.: “Security engineering with patterns-origins, theroretical model, and new applications”. Springer-Verlag. (2003).
13. Alexander, C.: “The Timeless Way of Building”. Oxford University Press. EEUU. (1979).
14. Yoshioka, N. et al.: “A survey on security patterns”. Progress in Informatics. (2008).
15. Fernandez, E. et al. “Classifying Security Patterns”. Progress in WWW Research and Development Volume 4976. (2008).
16. Washizaki, H. et al. “Improving the Classification of Security Patterns”. 20th International Workshop on Database and Expert Systems Application, DEXA’09. Austria. (2009).
17. Buschmann, F. et al. “Pattern-Oriented Software Architecture: A System of Patterns”. Chichester, UK: Wiley, 1996.
18. Schumacher, M. et al., “Security engineering with patterns”. Proceedings of the Conference on Pattern Languages of Programs, pp. 1–17. (2001).
19. Bunke, M. et al. “Organizating Security Patterns Related to Security and Pattern Recognition Requirements”. International Journal on Advances in Security, vol 5 no 1 & 2 (2012).
20. Kienzle, D.: “Security Patterns Template and Tutorial version 1.0”. (2008). Disponible en http://www.scrypt.net/~celer/securitypatterns/template%20and%20tutorial.pdf
21. Blakley, B. et al. “Security Design Patterns”. The Open Group. (2004). Disponible en https://www2.opengroup.org/ogsys/catalog/g031.
22. Henninger, V. et al. “Software pattern communities: Current practices and challenges”. Proceedings of the Conference on Pattern Languages of Programs PLOP. New York, NY, USA. ACM, 2007, pp. 14:1–14:19.
23. “Building In Security Maturity Model 4 - BSIMM4”. Disponible en http://www.bsimm.com/download
24. Darrell M. Kienzle et al. "Security Patterns Repository Version 1.0”. Disponible en http://www.scrypt.net/~celer/securitypatterns/repository.pdf