ISSN: 1646-9895

risti

Revista Ibérica de Sistemas e Tecnologias de Informação
Revista Ibérica de Sistemas y Tecnologías de Información

J u n h o 1 9 • J u n e 1 9

Nº 32

risti
Revista Ibérica de Sistemas e Tecnologias de Informação
Revista Ibérica de Sistemas y Tecnologías de Información

Edição / Edición
Nº 32, 06/2019

Tiragem / Tirage: 1000

Preço por número / Precio por número: 17,5€

Subscrição anual / Suscripción anual: 30€ (2 números)

ISSN: 1646-9895

Depósito legal:

Indexação / Indexación
Academic Journals Database, CiteFactor, Dialnet, DOAJ, DOI, EBSCO, GALE, IndexCopernicus, Index of Information Systems Journals, ISI Web of Knowledge, Latindex, ProQuest, QUALIS, SciELO, SCImago, Scopus, SIS, Ulrich's.

Propriedade e Publicação / Propriedad y Publicación
AISTI – Associação Ibérica de Sistemas e Tecnologias de Informação
Rua Quinta do Roseiral 76, 4435-209 Rio Tinto, Portugal
E-mail: aistic@gmail.com
Web: http://www.risti.xyz

risti
Revista Ibérica de Sistemas e Tecnologias de Informação
Revista Ibérica de Sistemas y Tecnologías de Información

Recebido/Submission: 09/01/2019
Aceitação/Acceptance: 23/03/2019

KE-SER: Un sistema basado en el conocimiento y la experiencia para dar soporte a arquitectos de software en aspectos de seguridad

María Celeste Carignano[1], Silvio Gonnet[1,2], Horacio Leone[1,2]

mcarigna@frsf.utn.edu.ar, sgonnet@santafe-conicet.gov.ar, hleone@santafe-conicet.gov.ar

[1] Universidad Tecnológica Nacional – Facultad Regional Santa Fe, Santa Fe, CP: 3000, Santa Fe, Argentina.

[2] INGAR – Instituto de Desarrollo y Diseño, CONICET-UTN, Santa Fe, CP: 3000, Santa Fe, Argentina.

DOI: 10.17013/risti.32.97–112

Resumen: En la actualidad, las organizaciones están expuestas a ataques cada vez más sofisticados, cuyo objetivo es comprometer o corromper la integridad, confidencialidad y disponibilidad de la información que manejan. La mayoría de los ataques, internos o externos, explotan vulnerabilidades de seguridad, las cuales pueden tener su origen en el desarrollo de los sistemas de software que emplean las organizaciones. Desarrollar sistemas que contemplen aspectos de seguridad es fundamental para disminuir las vulnerabilidades que exponen a las organizaciones. Sin embargo, las compañías de software pocas veces cuentan con expertos en seguridad que las asesoren a lo largo del ciclo de vida del desarrollo de software. En este trabajo se propone un sistema experto para brindar apoyo, desde el punto de vista de la seguridad, en una de las principales actividades del desarrollo como es el diseño arquitectónico.

Palabras-clave: sistema experto, seguridad, arquitectura de software, diseño, conocimiento.

KE-SER: A system based on knowledge and experience to support software architects in security aspects

Abstract: Nowadays, the organizations are exposed to increasingly sophisticated attacks, whose objective is to compromise or corrupt the integrity, confidentiality and availability of the information they handle. Most attacks exploit security vulnerabilities. The origin of these vulnerabilities can be in the development of the software systems that organizations employ. In order to reduce these vulnerabilities, it is fundamental to develop systems dealing with security aspects. However, software companies rarely have security experts to advise them throughout the software development life cycle. This paper proposes an expert system to provide support, from the point of view of security, in one of the main development activities such as architectural design.

Keywords*:* expert systems, security, software architecture, design, knowledge.

1. Introducción

La información siempre ha sido un recurso valioso para las organizaciones. En las últimas décadas, los avances tecnológicos han revolucionado la forma en la que se obtiene, procesa, comunica y almacena la información dentro y fuera de las organizaciones, generando ventajas competitivas y numerosos beneficios (Mejía & Muñoz, 2017). Sin embargo, estos avances también han colaborado con la aparición de ataques cada vez más sofisticados, cuyo objetivo es comprometer o corromper la integridad, confidencialidad y disponibilidad de la información que manejan dichas organizaciones.

La mayoría de los ataques, internos o externos, explotan vulnerabilidades de seguridad (Zambrano, Guarda, Valenzuela, & Quiña, 2019). La base de datos "*Common Vulnerabilities and Exposures"* (CVE, 2019) es una lista de vulnerabilidades y exposiciones de ciberseguridad divulgadas públicamente, ampliamente reconocida por la comunidad internacional de ciberseguridad, que define a una vulnerabilidad como:

> *una debilidad en la lógica computacional (por ejemplo, el código) que se encuentra en el software y algunos componentes de hardware (por ejemplo, el firmware) que, cuando se explota, produce un impacto negativo en la confidencialidad, la integridad o la disponibilidad. La mitigación de las vulnerabilidades en este contexto implica cambios en la codificación, pero también podría incluir cambios en las especificaciones o incluso la eliminación de las especificaciones (por ejemplo, la eliminación de los protocolos o la funcionalidad afectados en su totalidad).*

Teniendo en cuenta esto, resulta evidente la importancia de contemplar aspectos de seguridad durante el ciclo de vida del desarrollo de software (SDLC por sus siglas en inglés: "*Software Development Life Cycle*") con el objetivo de eliminar o disminuir las vulnerabilidades que podrían ser aprovechadas durante su ejecución. Cuanto más tarde se detecte el origen de una vulnerabilidad, mayor será el costo de su corrección y las consecuencias pueden llegar a ser catastróficas. Ejemplos de ello son los problemas ocasionados por los ataques "*Ransomware*" en la actualidad (Winkler & Treu Gomes, 2017).

Las mejores prácticas de desarrollo de software sugieren integrar los aspectos de seguridad en cada fase del SDLC. Existen varios enfoques que así lo proponen, como ser: Software Assurance Maturity Model (SAMM, 2019) de la organización Open Web Application Security Project (OWASP, 2019), Touchpoints de McGraw (MacGraw, 2006) y Security Development Lifecycle (SDL, 2019) de Microsoft, entre otros. Análisis comparativos de algunos de estos enfoques, o, en el caso de SAAM de su predecesor, pueden encontrarse en los trabajos de: De Win, Scandariato, Buyens, Grégoire, & Joosen (2009) y Mohammad, Alqatawna, & Abushariah (2017).

Sin embargo, las compañías de desarrollo de software no suelen contemplar la seguridad en el SDLC debido a que consideran que es muy costosa, que requiere de mucho tiempo y que no están suficientemente capacitadas para su realización, ya que, generalmente, no cuentan con expertos en seguridad entre sus miembros (Mohammad, Alqatawna, & Abushariah, 2017). En este contexto surge la motivación que da origen a la propuesta presentada en este trabajo.

Durante el diseño arquitectónico se toman las primeras decisiones de diseño que permiten establecer las bases para construir una solución de software que satisfaga los requerimientos de los stakeholders de un sistema (Bass, Clements, & Kazman, 2013; Clements, et al., 2010). Entre los requerimientos a satisfacer, se encuentran los requerimientos de seguridad. Estas decisiones tempranas son las más difíciles de corregir, las más difíciles de cambiar posteriormente en el SDLC, y tienen los efectos de mayor alcance (Bass, Clements, & Kazman, 2013). Por consiguiente, si los requerimientos de seguridad no son tenidos en cuenta de manera apropiada durante el diseño arquitectónico, el sistema resultante puede presentar vulnerabilidades que pongan en riesgo la seguridad del sistema desarrollado y la información de la organización que lo ejecuta.

Es por este motivo que se considera necesario brindar apoyo a los arquitectos de software para que puedan tomar decisiones durante el diseño arquitectónico de forma consciente e informada. En este trabajo, se describe un sistema capaz de realizar recomendaciones para satisfacer los requerimientos de seguridad de los sistemas a construir. Las recomendaciones pueden estar basadas en el conocimiento capturado de expertos en seguridad o en experiencias pasadas de arquitectos que diseñaron arquitecturas de sistemas con requerimientos similares.

A continuación, se presenta la propuesta con mayor nivel de detalle. En la sección 2, se describen las principales características del sistema experto, llamado KE-SER por sus siglas en inglés: "*Knowledge & Experience – SEcurity Recommendations*". En la sección 3, se profundizan los principales componentes de KE-SER. Luego, en la sección 4, se analizan los resultados de una evaluación del sistema experto propuesto. En la sección 6, se contextualiza la propuesta con referencia a trabajos relacionados. Finalmente, en la sección 5, se presentan las conclusiones y trabajos futuros.

2. KE-SER: conocimiento y experiencia

Los sistemas expertos cobran importancia y utilidad en aquellos dominios en los que resulta difícil y costoso disponer de expertos humanos para realizar consultas de manera personal, ilimitada e instantánea (Gupta & Singhal, 2013). Las compañías de desarrollo de software generalmente no cuentan con expertos en seguridad entre sus miembros (Mohammad, Alqatawna, & Abushariah, 2017), por lo que KE-SER puede ser de gran valor.

KE-SER es un sistema basado en el conocimiento y la experiencia que tiene como objetivo brindar soporte a los arquitectos de software en el proceso de diseño arquitectónico contemplando aspectos de seguridad. Los componentes de este sistema se presentan en la Fig. 1.

Los objetivos de KE-SER incluyen:

- responder a las consultas que podría realizar un arquitecto de software a un experto en seguridad para resolver los requerimientos de los stakeholders de los sistemas de software que deben ser construidos; y
- recordar las soluciones empleadas en el pasado para poder aplicarlas en el diseño de nuevas arquitecturas.

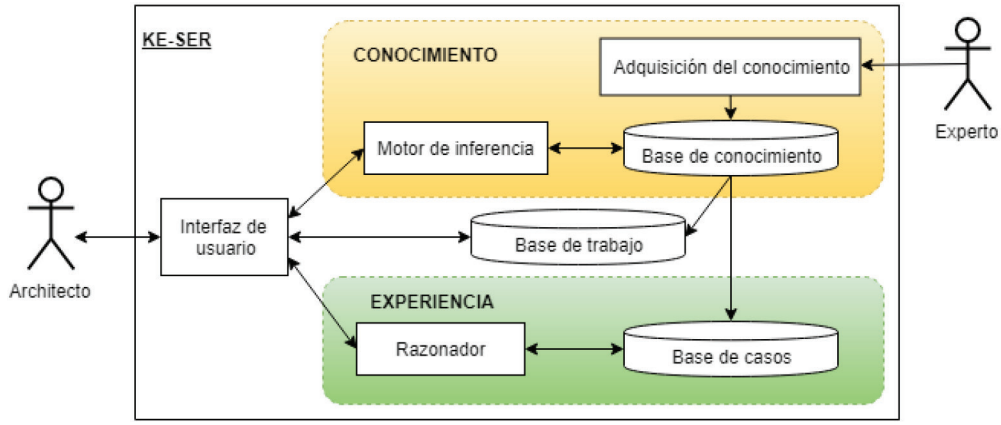

Figura 1 – Componentes de KE-SER

Para cumplir con el objetivo *I)* KE-SER emplea los componentes asociados al *CONOCIMIENTO* (en Fig. 1), y para cumplir con el objetivo *II)* KE-SER utiliza los componentes asociados a la *EXPERIENCIA* (en Fig. 1).

El sistema intenta ofrecer recomendaciones o consejos sobre seguridad ante preguntas como las siguientes: ¿Cuáles serían los aspectos a tener en cuenta para evitar el acceso no autorizado de individuos? ¿Cómo evitar que usuarios sin privilegios modifiquen los datos? ¿Cómo almacenar los datos para que no puedan ser leídos por personas ajenas al sistema? ¿Cuáles son los aspectos de seguridad que deberían ser tenidos en cuenta en aplicaciones como la que se está diseñando? ¿Cómo se diseñó la solución para un requerimiento de autenticación en aplicaciones anteriores?

Cabe destacar que KE-SER solo ofrece recomendaciones, con el objetivo de brindar información a los arquitectos de software para que tomen mejores decisiones durante el diseño arquitectónico, pero no ejecuta o genera ninguna decisión de manera automática.

La información que gestiona KE-SER está distribuida en tres bases de datos:

- La **Base de Conocimiento**, la cual contiene el conocimiento de los expertos en seguridad. El conocimiento está distribuido en objetos y su estructura se describe, mediante un diagrama de clases, en la sección 2.1.
- La **Base de trabajo**, la cual contiene la información de los sistemas que se están diseñando. En la sección 2.2 se describe la información que almacena esta base de datos.
- La **Base de Casos**, la cual contiene información sobre experiencias de diseño arquitectónico pasadas. Estas experiencias están representadas por casos y su estructura se describe, mediante diagramas de clases, en la sección 2.3.

El experto interactúa con KE-SER mediante el componente de **Adquisición de conocimiento**, el cual permite que el experto vuelque sus conocimientos en el sistema.

Para responder consultas tomando como base el conocimiento, se emplea el **Motor de Inferencia**, el cual simula, mediante un conjunto de algoritmos, la estrategia de resolución de problemas de un experto. En cambio, para responder consultas tomando como base la experiencia, se utiliza el **Razonador**, el cual utiliza razonamiento basado en casos para proponer estrategias de resolución basadas en el recuerdo de experiencias pasadas similares.

A partir del sistema experto propuesto, se ha construido una versión prototípica de una herramienta, llamada KE-T. KE-T ha sido implementada en JAVA (JAVA, 2019) como un complemento (*plug-in*) de Eclipse (Eclipse, 2019) utilizando SWT (siglas en inglés de Standard Widget Toolkit (SWT, 2019)) para construir la interfaz gráfica de la aplicación. Por restricciones en la longitud del trabajo no se presentará una descripción detallada de la misma. Pero resulta importante destacar, que ni el sistema experto, ni la herramienta presentada, pretenden automatizar el trabajo de los arquitectos, sino que el objetivo es brindar información que les permita tomar mejores decisiones durante el diseño de una arquitectura.

3. KE-SER: componentes

A continuación, se describen con mayor profundidad cada uno de estos componentes del sistema experto KE-SER.

3.1. Componentes asociados al conocimiento.

Los componentes asociados al conocimiento tienen como responsabilidad ofrecer recomendaciones a los arquitectos para que puedan resolver los requerimientos de seguridad de un sistema en particular. Con este fin, KE-SER mantiene una base de conocimiento (ver Fig. 1) que contiene el conocimiento volcado previamente por expertos en seguridad. La representación del conocimiento de los expertos se muestra en el diagrama de clases de la Fig. 2.

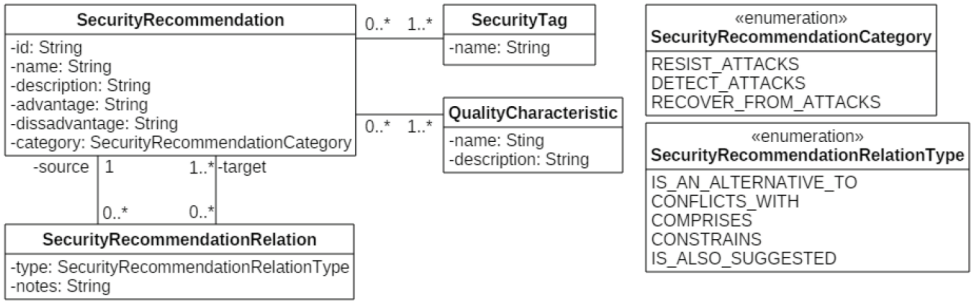

Figura 2 – Base de conocimiento de KE-SER

Las instancias de la clase *SecurityRecommendation* (ver Fig. 2) describen recomendaciones de seguridad que pueden ser útiles durante el diseño arquitectónico de un sistema. Ejemplos de recomendaciones son tácticas arquitectónicas, algoritmos específicos, o componentes arquitectónicos.

En general, las recomendaciones persiguen un objetivo específico, el cual puede ser resistir ataques de terceros, detectar ataques o recuperarse de ataques efectuados. La categoría de las recomendaciones (atributo *category* de *SecurityRecommendation* en Fig. 2) permite establecer los objetivos de cada recomendación documentada.

Además, los expertos deben proporcionar datos sobre las recomendaciones documentadas que permitan clasificarlas y recuperarlas. Un dato necesario es la característica de calidad que puede asociarse a cada recomendación (instancia de *QualityCharacteristic* en Fig. 2). Ejemplos de características de calidad están dados en la norma ISO/IEC 25010 (ISO 25010, 2011), como ser: confidencialidad, integridad, no-repudio, responsabilidad, y autenticidad; o en el trabajo de Bass, Clements & Kazman (2013): no-repudio, confidencialidad, integridad, garantía, disponibilidad, y auditoría. La restante información de clasificación puede ser documentada como etiquetas de la recomendación (instancias de *SecurityTag* en Fig. 2).

Un ejemplo de recomendación de seguridad podría ser el siguiente:

- **ID**: SR-AIC
- **Name**: Autenticar usuarios por medio de identificador y contraseña
- **Description**: La autenticación es el proceso de validar la identidad de un usuario, en este caso, mediante un ID de usuario y una contraseña.
- **Advantage**: Fácil de implementar. Bajo costo. Fácil de utilizar
- **Dissadvantage**: El nivel de seguridad depende directamente de la complejidad de la contraseña. Contraseñas simples o débiles son fáciles de adivinar o inventar. Contraseñas demasiado complejas conducen a los usuarios a aplicar estrategias para gestionarlas que no siempre son correctas. Proporcionar una contraseña correcta no prueba que una persona es quien dice ser.
- **Quality Characteristic**: Autenticidad
- **Category**: RESIST ATTACKS
- **Security Tags**: Autenticación, Usuario, Password, Contraseña, Identificador, Nombre de usuario

Otros ejemplos de recomendaciones pueden observarse en la Tabla 1.

Las recomendaciones presentadas en la Tabla 1 forman parte de una vista parcial de una base de conocimiento construida con fines prácticos. Algunas de las fuentes de información formales consultadas para confeccionar la base de conocimiento fueron: Bass, Clements & Kazman (2013) y Johnsson, Deogun, & Sawano (2019). También fueron consultados sitios web y expertos en seguridad. Por cuestiones de extensión no se detalla la descripción, ventajas y desventajas de las recomendaciones presentadas.

Finalmente, los expertos pueden indicar si existe algún tipo de relación que vincula varias recomendaciones (*SecurityRecommendationRelation* en Fig. 2). Sin embargo, por limitaciones en la extensión del trabajo no serán abordadas en esta presentación.

3.2. Soluciones en producción

En el presente trabajo no se prescribe cómo debe ser documentada la arquitectura de software a diseñar. Solo se establece la necesidad de capturar información de dos

ID	Name	Quality Characteristic	Category	SecurityTag
SR-AIC	*Autenticar usuarios por medio de identificador y contraseña*	Autenticidad	RESIST ATTACKS	Autenticación, Usuario, Password, Contraseña, Identificador, Nombre de usuario
SR-AIB	*Autenticar usuarios por medio de identificaciones biométricas*	Autenticidad	RESIST ATTACKS	Autenticación, Usuario, Biométricas, Biometría
SR-AMF	*Autenticar usuarios por medio de múltiples factores*	Autenticidad	RESIST ATTACKS	Autenticación, Usuario, Biométricas, Biometría, Password, Contraseña, Identificador
SR-AT	*Mantener un "audit trail".*	Disponibilidad Auditoría No-repudio	RECOVER FROM ATTACKS	Audit trail, Log
SR-CD	*Mantener la confidencialidad de los datos*	Confidencialidad	RESIST ATTACKS	Datos, Protección, Acceso, Encriptar, Encriptación, Canales de comunicación, VPN, SSL

Tabla 1 – Ejemplos de recomendaciones

elementos: los intereses de los stakeholders y las decisiones de diseño de los arquitectos, ambos referidos con aspectos de seguridad de los sistemas de software a diseñar.

La información de seguridad relacionada con los sistemas que se están diseñando es almacenada en la **base de trabajo**. En la Fig. 3 se presenta la estructura de los conceptos involucrados en el diseño arquitectónico desde la perspectiva de la seguridad.

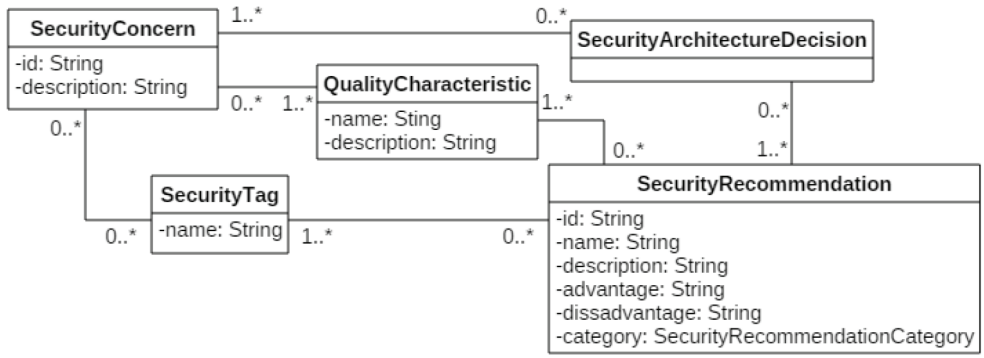

Figura 3 – Conceptos de KE-SER

Los intereses de los stakeholders referidos a seguridad están representados mediante la clase *SecurityConcern* (ver Fig. 3). Cada interés de seguridad se categoriza mediante las características de seguridad asociadas y las etiquetas de seguridad identificadas.

De esta forma, si un *stakeholder* expresa un interés expresado de la siguiente forma: "*Se deberá dejar registrado todo intento de acceder al sistema con un nombre de usuario no existente*", puede ser documentado como se muestra en la Tabla 2.

Las decisiones tomadas por los arquitectos, para satisfacer los intereses de seguridad, son documentadas mediante objetos de tipo *SecurityArchitectureDecision* (ver Fig. 3). Dichas decisiones pueden involucrar la adopción de una o más recomendaciones brindadas por el sistema experto (*SecurityRecommendation* en Fig. 3).

ID	Description	Quality Characteristic	SecurityTag
SC01	Se deberá dejar registrado todo intento de acceder al sistema con un nombre de usuario no existente	Autenticidad Auditoría	Autenticación. Usuario no autorizado, Nombre de usuario, Audit trail, Log

Tabla 2 – Ejemplo de instancia de *SecurityConcern*

Teniendo en cuenta las recomendaciones presentadas en la Tabla 1, y el interés de seguridad descrito en la Tabla 2, las decisiones de diseño tomadas podrían ser las mostradas en la Tabla 3.

SecurityConcern		SecurityRecommendation	
ID	**Description**	**ID**	**Description**
SC01	Se deberá dejar registrado todo intento de acceder al sistema por parte de un usuario no autorizado	*SR-AIC*	*Autenticar usuarios por medio de identificador y contraseña*
		SR-AT	*Mantener un "audit trail"*

Tabla 3 – Ejemplo de instancia de *SecurityArchitectureDecision*

3.3. Componentes asociados a la experiencia

Los componentes asociados a la experiencia tienen como responsabilidad recuperar las soluciones de seguridad empleadas en el pasado para proponerlas a los arquitectos durante el diseño de nuevas arquitecturas. Para ello KE-SER aplica razonamiento basado en casos (CBR, por sus siglas en inglés *Case-based reasoning*). Razonamiento Basado en Casos es un paradigma de resolución de problemas (Aamodt & Plaza, 1994) que involucra el uso de experiencias pasadas para comprender y resolver nuevas situaciones (Kolodner, 1992). Un ***caso*** denota una situación experimentada previamente, que ha sido capturada y aprendida de forma tal que puede ser reutilizada en la resolución de problemas futuros (Aamodt & Plaza, 1994). Suele estar compuesto por el *problema*, que describe el estado del mundo cuando ocurre el caso; y la *solución*, que establece la solución encontrada (Watson & Marir, 1994).

En el contexto de este trabajo, un caso describe las decisiones de diseño tomadas durante la definición de una arquitectura de software que involucran aspectos de seguridad y es llamado ***caso arquitectónico de seguridad***. En la Fig. 4 se presenta la estructura definida para los *casos arquitectónicos de seguridad*.

Un *caso arquitectónico de seguridad* es representado mediante la clase *SecurityArchitectureCase* (ver Fig. 4). El *problema* (*ProblemDescription* en Fig. 4) está definido en función de los intereses de seguridad de los stakeholders (*SecurityConcern*

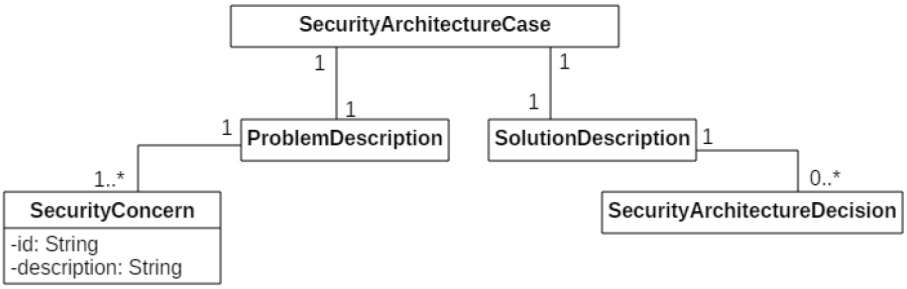

Figura 4 – Estructura de un caso arquitectónico

en Fig. 4) y la solución (*SolutionDescription* en Fig. 4) se compone de decisiones de diseño (*SecurityArchitectureDecision* en Fig. 4) tomadas por los arquitectos de software aceptando recomendaciones de diseño propuestas.

Los *casos arquitectónicos de seguridad* se almacenan en la base de casos (ver Fig. 1) y contienen la información de los sistemas que se encontraban en la base de trabajo, una vez que se finaliza con el diseño de su arquitectura.

3.4. Motor de inferencia

El motor de inferencia simula la estrategia de resolución de problemas de un experto. Para ello ejecuta un proceso que tiene como entrada un conjunto de intereses de seguridad (instancias de *SecurityConcern*) del sistema a construir y como salida cero o más recomendaciones de seguridad (instancias de *SecurityRecommendation*) para cada uno de los intereses de entrada. La cantidad de recomendaciones propuestas dependerá de la información almacenada en la base de conocimiento y de la calidad descriptiva de las etiquetas de seguridad empleadas en la definición de los intereses de seguridad y las recomendaciones de seguridad.

El proceso de inferencia analiza cada uno de los intereses de seguridad de entrada (llamado *ISE*) aplicando la siguiente regla:

> **SI** (existe un conjunto de recomendaciones de seguridad, llamado *CR*, que poseen características de calidad (instancias de *QualityCharacteristic*) comunes con *ISE* y también poseen etiquetas de seguridad (instancias de *SecurityTag*) comunes con *ISE*) **ENTONCES** proponer las recomendaciones del conjunto *CR* para el interés de seguridad *ISE*.

Una vez finalizado el procesamiento de todos los intereses de seguridad del conjunto de entrada, se presentan al arquitecto las recomendaciones de seguridad obtenidas para cada uno de ellos.

Para determinar si una recomendación de seguridad (llamada ***RS***) posee características de calidad comunes con un interés de seguridad (llamado ***IS***), el sistema experto realiza el siguiente cálculo (I):

*CIS = características de calidad de **IS*** *CR = características de calidad asociadas a **RS*** *IC = intersección de CIS y CR* *X = cantidad de características en IC/cantidad de características en CIS*	(I)

Si ***X > U_C*** entonces se considera que ***RS*** posee características de calidad comunes con ***IS***, en donde U_C es un valor umbral previamente establecido que indica un límite inferior de inclusión de las recomendaciones.

Por otro lado, para determinar si una recomendación de seguridad (llamada ***RS***) posee etiquetas de seguridad comunes con un interés de seguridad (llamado ***IS***), el sistema experto realiza el siguiente cálculo (II):

*EIS = etiquetas de seguridad de **IS*** *ER = etiquetas de seguridad asociadas a **RS*** *IE = intersección de EIS y ER* *Y = cantidad de etiquetas en IE/cantidad de etiquetas de EIS*	(II)

Si ***Y > U_E*** entonces se considera que ***RS*** posee etiquetas de seguridad comunes con ***IS***, en donde U_E es un valor umbral previamente establecido que indica un límite inferior de inclusión de las recomendaciones.

3.5. Razonador

El ***Razonador*** utiliza razonamiento basado en casos para proponer recomendaciones de seguridad basadas en el recuerdo de experiencias pasadas similares. Ante la descripción de los intereses de seguridad (instancias *SecurityConcern*) del sistema a ser diseñado, se recuperan de la base de casos aquellos *casos arquitectónicos de seguridad* (instancias de *SecurityArchitectureCase*) que tienen intereses similares y se proponen para el nuevo diseño arquitectónico las recomendaciones de seguridad consideradas en dichos casos.

Para esto, KE-SER recorre todos los casos de la *base de casos*, calculando para cada uno de ellos el porcentaje de intereses de seguridad del sistema a construir que tienen similitud con los intereses del caso que está siendo analizado. Aquellos casos, cuyo porcentaje supera un valor umbral previamente establecido (llamado ***U_I***), pasan a formar parte del conjunto de casos recuperados.

Como resultado, el arquitecto puede consultar para cada uno de los casos recuperados cuáles fueron las recomendaciones que se siguieron para satisfacer los intereses de seguridad que son similares a los intereses del sistema a construir.

Para establecer si dos intereses de seguridad son similares (llamando ***ISS*** al interés de seguridad del sistema a construir y ***ISC*** al interés de seguridad del caso de seguridad arquitectónico), se tienen en cuenta los siguientes aspectos:

Por un lado, las características de calidad de ambos intereses, realizando el cálculo presentado en (III):

CIS_1 = características de calidad de ISS *CIS_2 = características de calidad de ISC* *IC = intersección de CIS_1 y CIS_2* *X = cantidad de características en IC/cantidad de características de CIS_1*	(III)

Por el otro, las etiquetas de seguridad de ambos intereses, realizando el cálculo presentado en (IV):

EIS_1 = etiquetas de seguridad de ISS *EIS_2 = etiquetas de seguridad de ISC* *IE = intersección de EIS_1 y EIS_2* *Y = cantidad de etiquetas en IE/cantidad de etiquetas de EIS_1*	(IV)

De esta forma, si ***ISC*** cumple con la restricción presentada en (V) pasa a ser considerado por KE-SER como similar a ***ISS***, por lo que las recomendaciones de seguridad contempladas al momento de resolver a ***ISC*** serán propuestas al arquitecto para resolver a ***ISS***.

$X > U\_IC \;\; \& \;\; Y > U\_IE$	(V)

En donde U_IC y U_IE son valores umbrales previamente establecidos.

4. Evaluación y resultados

Para realizar la evaluación del modelo propuesto se utilizó un dataset público llamado PROMISE (PROMISE, 2019). Este dataset consta de 625 sentencias de requerimientos que pertenecen a 15 proyectos de desarrollo, de las cuales 66 corresponden a requerimientos de seguridad. En la evaluación se tuvieron en cuenta 57 sentencias de requerimientos de seguridad, debido a que 9 fueron descartadas por considerarse mal clasificadas o por carecer de suficiente información como para trabajar con ellas. Las sentencias de requerimientos de seguridad fueron consideradas como intereses de seguridad (instancias de *SecurityConcern* en Fig. 3).

La evaluación se llevó a cabo en tres pasos: 1) se clasificaron los intereses de calidad (asociándolos a instancias de *QualityCharacteristic*); 2) se etiquetaron los intereses de seguridad (asociándolos a instancias de *SecurityTag*); y 3) se empleó KE-SER para recuperar recomendaciones de seguridad.

En la Tabla 4 se presentan los resultados del primer paso discriminando los intereses de seguridad por proyecto.

Durante el segundo paso se asignaron etiquetas de seguridad a los intereses de seguridad analizados. Las etiquetas utilizadas se mencionan a continuación, indicándose entre paréntesis la cantidad de intereses de seguridad a las que fueron asignadas: Acceso (9), Usuario autenticado (9), Autorización (9), Rol usuario (18), Usuario autorizado (34), Autorización (34), Datos (7), Datos correctos (7), Datos válidos (7), Integridad (10), Acceso información (17), Acceso servicios (16), Nivel de acceso (1), Informe de usuario (2), Auditoría (4), Nombre de usuario (2), Contraseña (3), Acceso subsistemas (1),

Características de calidad	Proyectos															Total
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	
Confidencialidad	0	0	6	7	4	1	2	7	0	1	3	0	0	1	2	34
Autenticidad	1	1	0	1	0	1	0	0	0	0	0	0	2	1	0	7
Integridad	0	1	1	0	1	2	0	3	0	0	0	1	0	0	0	*9*
Auditoría	0	0	0	2	0	0	0	1	0	0	0	1	0	0	0	*4*
Confidencialidad + Autenticidad	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	*2*
Confidencialidad + integridad	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	*1*
TOTAL	1	3	7	10	5	4	2	12	0	1	3	3	2	2	2	*57*

Tabla 4 – Métricas de intereses de seguridad según características de calidad asociadas

Encriptación (3), Detección ataques (2), Notificaciones (1), Verificación integridad (1), Seguimiento transacciones (1), No cookies (1), Archivos de datos (2), Base de datos (2).

Durante el tercer paso, se llevaron a cabo distintas consultas al sistema experto KE-SER.

Por un lado, se realizaron consultas individuales al sistema experto para obtener recomendaciones basadas en el conocimiento. Para ello se llevaron a cabo 15 consultas. En cada i-ésima consulta se le solicitó a KE-SER conocer las recomendaciones basadas en el conocimiento para los intereses de seguridad del i-ésimo proyecto. Debido a que la Base de Conocimiento había sido nutrida con suficiente conocimiento, se obtuvieron recomendaciones para todas las sentencias de seguridad de todos los proyectos consultados. Por restricciones en la longitud del trabajo no se presentan detalles de la Base de conocimiento empleada para realizar la evaluación.

Los resultados de este tipo de consultas a KE-SER dependen del conocimiento volcado en la base de conocimiento. Si el conocimiento y la clasificación de las etiquetas es escaso, es poco probable que se obtengan respuestas útiles del sistema experto. Este hecho es análogo a las respuestas que se podrían obtener de expertos humanos, si no tienen mucho conocimiento no pueden realizar recomendaciones útiles y de calidad.

Por otro lado, se realizaron consultas al sistema experto para obtener recomendaciones basadas en la experiencia. Para ello, se llevaron a cabo 15 iteraciones. En cada i-ésima iteración se realizaron las siguientes actividades:

i. consulta a KE-SER de las recomendaciones basadas en la experiencia para los intereses de seguridad del i-ésimo proyecto;
ii. creación del correspondiente caso arquitectónico de seguridad (*SecurityArchitectureCase* en Fig. 4) con la información del problema (dado por las sentencias de seguridad del i-ésimo proyecto) y de la solución, basada en las recomendaciones recibidas. En aquellos casos en los que no se obtuvieron recomendaciones útiles para un interés de seguridad, se crearon recomendaciones basadas en el conocimiento para dar solución a dichos intereses y poder continuar con el proceso;
iii. almacenamiento del i-ésimo caso arquitectónico en la base de casos.

Los resultados obtenidos se presentan en la Tabla 5. Como puede observarse en la Tabla, un 79% de los intereses de seguridad planteados como consulta obtuvieron recomendaciones basadas en la experiencia. Los resultados de este tipo de consulta a KE-SER dependen de la experiencia que tenga el sistema experto, es decir, de la cantidad de casos en la Base de Casos.

	Proyectos															**Total**
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	
Casos en la Base de casos al momento de consultar por proyecto	Ø	1	1-2	1-3	1-4	1-5	1-6	1-7	1-8	1-9	1-10	1-11	1-12	1-13	1-14	
Intereses de seguridad	1	3	7	10	5	4	2	12	0	1	3	3	2	2	2	57
Intereses de seguridad con recomendaciones aplicadas surgidas de la experiencia	0	2	4	6	5	4	2	9	0	1	3	3	2	2	2	45

Tabla 5 – Métricas de sentencias de seguridad con recomendaciones basadas en la experiencia según proyecto

Los primeros proyectos analizados (como ser 1, 2 y 3) obtuvieron peores resultados que los últimos, debido a que al momento de ser tratados la base de casos no contaba con experiencias pasadas para recuperar. Este comportamiento también es análogo al de expertos humanos, aquellos expertos que cuentan con más experiencia tienen más probabilidades de responder consultas con recomendaciones útiles y de calidad.

5. Trabajos relacionados

En la actualidad, en el contexto del diseño arquitectónico existen varias herramientas que brindan asistencia a los arquitectos de software para la realización de sus tareas. En Rodríguez, Díaz Pace, & Soria (2018) y Vazquez, Díaz Pace, & Campo (2010) se describen enfoques que emplean en CBR para explorer alternativas de diseño de arquitecturas orientadas al servicio (SOA) y arquitecturas a nivel general, respectivamente. En ambos trabajos los requerimientos de calidad toman un papel importante, pero el objetivo es dar soporte al diseño detallado o implementación de las arquitecturas de entrada.

En Carignano, Gonnet, & Leone (2016) se describe una herramienta que utiliza CBR para dar soporte al diseño de arquitecturas de software, a partir de un conjunto de requerimientos de calidad y restricciones de diseño. Esta herramienta contempla todo tipo de requerimiento de calidad de manera general.

En Gomes & Leitäo (2006) también se propone una herramienta que aplica CBR para reutilizar conocimiento de diseño de software, empleando diagramas de clases UML para describir el sistema a diseñar.

Las herramientas y enfoques mencionados hasta aquí contemplan los requerimientos de calidad de manera general, siendo la seguridad una parte más de ellos. La diferencia con el enfoque propuesto se puede describir con base en dos pilares. Por un lado, el enfoque propuesto está especializado en los requerimientos de calidad de seguridad, por lo que puede ser abordado con mayor nivel de detalle. Por el otro, KE-SER no solo trabaja desde la perspectiva de la experiencia, como las herramientas previamente mencionadas, sino que también integra la perspectiva del conocimiento proveyendo los beneficios de contar con un experto con el conocimiento necesario para resolver problemas cuando la experiencia no es suficiente.

En el ámbito de seguridad, varios trabajos han propuesto el empleo de sistemas expertos, por ejemplo: Yang, Hu, and Chen (2004), o Rani and Goel (2015). Sin embargo, no se han encontrado trabajos relacionados a la utilización de sistemas expertos para dar soporte al diseño de arquitecturas de software.

6. Conclusiones y trabajos futuros

Las mejores prácticas de desarrollo de software sugieren integrar los aspectos de seguridad en cada fase del SDLC. Por lo que en este trabajo se presentaron las características generales de un sistema experto, llamado KE-SER, cuyo objetivo es dar soporte a los arquitectos de software en sus tareas en aspectos relacionados con seguridad.

El sistema experto KE-SER cuenta con la posibilidad de ofrecer recomendaciones de seguridad desde el punto de vista de información experta o de la experiencia. De forma tal que el arquitecto pueda seleccionar aquellas que considere más adecuadas para su aplicación. Además, en este trabajo se describió brevemente una herramienta que permite materializar el sistema experto propuesto, llamada KE-T.

Como trabajo futuro, se plantea el enriquecimiento de las etiquetas de seguridad (*SecurityTag*) para que reconozca sinónimos, antónimos y homónimos. Las etiquetas de seguridad son conceptos muy importantes, en el contexto de KE-SER y una definición detallada y rica de ellas permite un mejor funcionamiento del sistema experto.

Referencias

Aamodt, A., & Plaza, P. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. AI Communications. IOS Press, 7(1):39–59, 1994.

Bass, L., Clements, P., & Kazman, R. (2013). Software Architecture in Practice, 3rd Edition. Boston: Addison-Wesley. DOI: 10.1145/2693208.2693252

Carignano, M.C., Gonnet, S., & Leone, H. (2016). RADS: una herramienta para reutilizar estrategias en diseños de arquitecturas de software. In Simposio Argentino de Ingeniería de Software (ASSE 2016) - JAIIO 45. SSN: 2451–7593, pp. 147-158

Clements, P., Bachmann, F., Bass, L., Garlan, D., Ivers, J., Little, R., Merson, P., Nord, R., & Stafford, J. (2010). Documenting Software Architectures: Views and Beyond (2nd Edition). Boston: Addison-Wesley.

CVE (2019). Common Vulnerabilities and Exposures. Recuperado de: cve.mitre.org/about Accedido: 02/2019.

Yang, D., Hu, C., & Chen, Y. (2004). A framework of cooperating intrusion detection based on clustering analysis and expert system. In Proceedings of the 3rd international conference on Information security (InfoSecu '04). ACM, New York, NY, USA, 150–154. DOI: 10.1145/1046290.1046321

De Win, B., Scandariato, R., Buyens, K., Grégoire, J., & Joosen, W. (2009). On the secure software development process: CLASP, SDL and Touchpoints compared. *Information and Software Technology*, 51(7), 1152–1171. DOI: 10.1016/j.infsof.2008.01.010.

Eclipse (2019). Eclipse Foundation. Eclipse. Recuperado de: www.eclipse.org. Accedido: 22-02-2019.

Gomes, P., & Leitão, A. (2006). A tool for management and reuse of software design knowledge. In: Staab S., Svátek V. (eds) Managing Knowledge in a World of Networks. EKAW 2006. Lecture Notes in Computer Science, vol 4248. Berlin, Heidelberg: Springer. DOI: 10.1007/11891451_3

Gupta, S., & Singhal, R. (2013). Fundamentals and Characteristics of an Expert System. International Journal on Recent and Innovation Trends in Computing and Communication, 1(3), 110–113.

ISO 25010 (2011). ISO/IEC 25010:2011 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models. International Organization for Standardization/ International Electrotechnical Commission.

ISO 42010 (2011). ISO/IEC/IEEE 42010: International Organization for Standardization and International Electrotechnical Commission. Systems and software engineering - architecture description. Edition 01-12-2011.

Java (2019). Oracle Corporation. Java. Recuperado de: www.oracle.com/technetwork/java/index.html. Accedido: 22-02-2019.

Johnsson, D., Deogun, D., & Sawano, D. (2019). Secure by Design. Shelter Island, NY: Manning Publications

Kolodner, J. (1992). An introduction to case-based reasoning. Artificial Intelligence Review, 6, 3–34. DOI: 10.1007/BF00155578

MacGraw, G. (2006). Software security: building security in. Boston: Addison-Wesley.

Mejía, J., & Muñoz, M. (2017). Tendencias en Tecnologías de Información y Comunicación. RISTI - Revista Ibérica de Sistemas y Tecnología de información, (21), 51–66. Doi: 10.17013/risti.21.51-66.

Mohammad, A., Alqatawna, J., & Abushariah, M. (2017). Secure Software Engineering: Evaluation of Emerging Trends. 8th International Conference on Information Technology (ICIT), 814–818, DOI:10.1109/ICITECH.2017.8079952

OWASP (2019). The OWASP Foundation. Recuperado de: www.owasp.org. Accedido: 02/2019

PROMISE (2019). PROMISE Software Engineering Repository. Recuperado de: promise.site.uottawa.ca/SERepository/. Accedido: 05/2019.

Rani C., & Goel, S. (2015). CSAAES: An expert system for cyber security attack awareness. In *International Conference on Computing, Communication & Automation*, Noida, 2015, pp. 242–245. DOI: 10.1109/CCAA.2015.7148381

Rodríguez, G., Díaz-Pace, J. A., & Soria, A. (2018). A case-based reasoning approach to reuse quality-driven designs in service-oriented architectures. Information Systems, 77(2018), 167–189. DOI: 10.1016/j.is.2018.06.003.

SAMM (2019). Software Assurance Maturity Model. Recuperado de: www.owasp.org/index.php/OWASP_SAMM_Project. Accedido: 22-02-2019.

SDL (2019). Microsoft Security Development Lifecycle, Microsoft. Recuperado de: www.microsoft.com/en-us/sdl. Accedido: 22-02-2019.

SWT (2019). Eclipse Foundation. Standard widget toolkit. Recuperado de: www.eclipse.org/swt. Accedido: 22-02-2019.

Watson, I., & Marir, F. (1994). Case-based reasoning: A review. The Knowledge Engineering Review, 9(4), 327–354.

Winkler, I. & Treu Gomes, A. (2017). Chapter 7 - Adversary Infrastructure. In I. Winkler & A. Treu Gomes (Eds.). *Advanced Persistent Security*, (pp. 67 – 79). Rockland, MA: Syngress. DOI: 10.1016/B978-0-12-809316-0.00007-5

Zambrano, A., Guarda, T., Valenzuela, E.V.H., & Quiña, G. N. (2019). Técnicas de mitigación para principales vulnerabilidades de seguridad en aplicaciones web. RISTI - Revista Iberica de Sistemas e Tecnologias de Informacao, (E17), 299-308.