Aplicar el Modelo de Amenazas para incluir la Seguridad en el Modelado de Sistemas

Marta Castellaro, Susana Romaniz, Juan Carlos Ramos,Carlos Feck, Ivana Gaspoz
Argentina, Universidad Tecnológica Nacional, Facultad Regional Santa Fe
mcastell@frsf.utn.edu.ar - sromaniz@frsf.utn.edu.ar - jramos@frsf.utn.edu.ar
Lavaise 610 – 3000 Santa Fe -Argentina, Tel, Fax: +54 .0342.4690348

Resumen. Hasta hace poco, la seguridad era concebida como una propiedad deseable del software y formaba parte del conjunto de atributos a considerar para determinar la calidad del mismo. El aumento de ataques a aplicaciones vulnerables ha conducido a un reconocimiento gradual que las protecciones a nivel de infraestructura (hoy de uso común para proteger sistemas accesibles por Internet) no son suficientes para tal objetivo. Se habla de "producir software seguro" y esto es concebido como una disciplina, que si bien se presenta menos madura que otras relacionadas con la calidad, cuenta con iniciativas, enfoques, metodologías y tecnologías emergentes. Este trabajo reúne algunos resultados de análisis respecto a terminología y metodologías sobre desarrollo de software seguro, y herramientas para instrumentarlo en las actividades de modelado y diseño de aplicaciones, cuando el equipo de desarrollo está constituido por profesionales no expertos en seguridad.

Palabras claves. Seguridad. Modelado de amenazas. Diseño de Software Seguro.

1.Introducción

Hasta hace poco tiempo, la seguridad era concebida con una propiedad deseable para el aseguramiento del software y formaba parte del conjunto de atributos a considerar para determinar la calidad de un software. El aumento de ataques de aplicaciones vulnerables ha conducido a un reconocimiento gradual respecto a que las protecciones a nivel de infraestructura (hoy de uso común para proteger sistemas accesibles por Internet) no son suficientes para tal objetivo. Se reconoce que las protecciones y mitigaciones de seguridad de las aplicaciones deben ser especificadas a nivel del sistema y su arquitectura, e implementadas durante el desarrollo de las aplicaciones y su operación.
En los últimos años esto ha evolucionado, y la seguridad del software es considerada como una característica del software que puede ser consistentemente demostrada [1]. La garantía de producir software seguro es concebida como una disciplina, y si bien se presenta menos madura que otras relacionadas con la calidad, se cuenta con iniciativas, enfoques, metodologías y tecnologías emergentes. Nuevos conceptos aparecen asociados al software en riesgo: vulnerabilidades, amenazas, ataques y ello conlleva a una taxonomía específica,

clasificaciones y definiciones de patrones.
Existen diferentes definiciones acerca de 'Aseguramiento del Software (Software Assurance)' establecidas por organismos reconocidos (CNSS-Committee on National Security Systems; NASA-National Aeronautics and Space Administration; DHS-Department of Homeland Security; NIST-National Institute of Standards and Technology), pero todas ellas presentan un criterio común: "La seguridad se presenta en todo el ciclo de vida del software. Un software seguro es diseñado, implementado, configurado y operado de manera que cumpla con las propiedades esenciales: continúe operando en presencia de ataques, aísle o limite los daños y se recupere lo antes posible".

2. Ingeniería de Sistemas Seguros

El software es típicamente un elemento o parte de un sistema más grande, sea este un sistema intensivo en software, o un sistema compuesto por elementos de hardware y software. Por ello la Ingeniería de Sistemas ha evolucionado desarrollando publicaciones y técnicas sólidas para sistemas, orientados a la construcción de sistemas seguros. Todo ello ha conformado la "Ingeniería de Sistemas Seguros", una disciplina que trata sobre la construcción de sistemas que deben permanecer funcionando como se espera ante la maldad, el error, o el azar. Como toda disciplina, se enfoca en los instrumentos, procesos, y métodos que se emplean para diseñar, poner en práctica, y probar sistemas completos, y adaptar sistemas existentes a su ambiente.
Las actividades de ingeniería de sistemas involucradas en la construcción de sistemas seguros, son: (1) El desarrollo de requerimientos de seguridad de sistemas; (2) El desarrollo de diseños de sistemas seguros; (3) La integración de componentes de subsistema; (4) Las pruebas de sistemas y subsistemas. En este trabajo nos centramos en las dos primeras actividades, incluyendo en el diseño las actividades de planificación de pruebas, por lo que luego las discutiremos en más detalle.

2.1 Desarrollo de requerimientos de software seguro

Se reconoce que muchos problemas de seguridad de software provienen de la especificación inadecuada o inexacta de los requerimientos o del desajuste entre la interpretación de los mismos durante la reconstrucción y su intención real.
Los requerimientos para las funcionalidades de seguridad en sistemas intensivos de software a menudo son confundidos con requerimientos para el software seguro.
La primera categoría incluye las funciones que ponen en práctica una política de seguridad (funciones de control de acceso, identificación, autenticación y autorización, y las funciones que realizan el cifrado y el descifrado, y la gestión de claves) y también las funciones que previenen la violación de las propiedades de seguridad del sistema o la información que procesa, como el acceso no autorizado, la modificación, la negación de servicio, descubrimiento, etc. Se conocen también como **'requerimientos de servicios de seguridad'.**
La segunda categoría de requerimientos afecta directamente a la probabilidad que el software en sí mismo sea seguro y, en general, son no funcionales (o propiedades). En conjunto, aseguran que el sistema ha de permanecer funcionando correctamente incluso cuando se encuentre bajo amenaza; están orientados a la reducción o eliminación de vulnerabilidades en el software y están vinculados principalmente al plan de desarrollo de software y al control de gestión del proyecto. Se conocen como '**objetivos de seguridad**' [2].

Durante la obtención de requerimientos se debe recolectar "información relativa a la seguridad" tal como: (a) determinar y valorar los **activos,** definiendo qué se va a proteger (activos tangibles o intangibles) y determinando su valor (económico y no económico); (b) determinar los **actores**, dueños de los datos; (c) determinar los **casos de uso**, roles de cada actor; (d) determinar los requerimientos **legales y de negocios** (restricciones legales y de negocios, requerimientos de auditoría y no repudio, políticas de seguridad); (e) determinar las **amenazas** sobre los activos identificados, su probabilidad de ocurrencia y la política de manejo; (f) definir la **arquitectura** del sistema, entendiendo cómo se integran los mecanismos de seguridad al sistema, cómo se maneja la confianza, y cómo se comportará el sistema frente a un ataque (se debe especificar un modelo a alto nivel de amenazas y referencias a los documentos que describen los procedimientos de seguridad; también se deben enumerar las medidas de seguridad a implementar).
Durante el análisis de requerimientos se pueden identificar diversas características que derivan en los requerimientos de seguridad del software, por ejemplo: (a) arquitectura de la aplicación; (b) plataforma a emplear; (c) requisitos de cumplimiento de normativas y marcos regulatorios; (d) tipo de conectividad; (e) tipos de datos (confidenciales, públicos, etc.) que se almacenan o transmiten; (f) tipo de almacenamiento de datos (local o remoto, centralizado o distribuido, etc.); (g) roles de usuarios necesarios para la aplicación; (h) tipo de acceso a los datos por parte de cada rol; (i) acciones sobre el sistema que puede hacer cada rol (cambiar la configuración, arrancar o detener servicios, etc.); (j) tipo de registro que el sistema debe generar (de acceso a recursos, de cambio de configuraciones, de cambio de privilegios, etc.).
Luego es necesario 'identificar' los requerimientos de seguridad. Los usuarios pueden no ser totalmente conscientes de los riesgos de seguridad y las vulnerabilidades asociadas con su sistema. Para esta tarea, los ingenieros de sistemas en conjunto con los usuarios, pueden realizar un análisis descendente de los posibles fallos de seguridad que podrían causar riesgo a la organización, así como definir exigencias para identificar vulnerabilidades.
Una técnica que favorece este análisis descendente es el análisis de **árbol de amenaza** (a veces llamado árbol de ataque). En este análisis, se coloca el objetivo del atacante en lo alto del árbol y luego el analista identifica posibles alternativas para alcanzar dicho objetivo. Para cada alternativa, el analista recurrentemente puede añadir alternativas para alcanzar los sub-objetivos que componen el objetivo principal. Este proceso se repite para cada objetivo de atacante. Examinando los nodos de nivel más bajos del árbol de ataque resultante, el analista puede identificar las técnicas posibles para violar la seguridad del sistema, y así se pueden especificar como requerimientos de seguridad las prevenciones para estas técnicas.
Otras técnicas que apoyan el desarrollo de los requerimientos de seguridad son el **modelado de amenazas** y el desarrollo de **casos de mal uso y casos de abuso** [3].
Los resultados del análisis de requerimientos de seguridad del sistema se deben usar como la base para los escenarios de casos de prueba de seguridad que van a ser empleados durante la integración o pruebas de la aceptación.

2.2 Desarrollo de diseños de sistemas seguros

Respecto al modelado, en esta etapa se realiza una aproximación para descubrir y aprender sobre los requerimientos del software, lo que proporciona un modo de prever los funcionamientos y las interacciones del software propuesto dentro de su ambiente. Por

lo tanto, el desarrollo de software seguro requiere modelar las posibles amenazas explícitamente. Existen herramientas y técnicas que facilitan el modelado de amenazas, ataques y vulnerabilidades.
En el diseño de sistemas seguros, se deben incorporar varios rasgos claves de diseño para orientar la identificación de vulnerabilidades típicas de sistemas. Son ejemplos ilustrativos: el diseño de protocolo de seguridad, el diseño de control de acceso y contraseña, el control de publicaciones en sistemas distribuidos, el control de concurrencia, la tolerancia a fallos, y la recuperación de fallos; algunos están asociados a aspectos funcionales, pero existen otros que no están directamente relacionados con la funcionalidad de seguridad.
Para el diseño detallado se pueden identificar patrones de diseño. Así como existen patrones de diseño para problemas recurrentes de sistemas de información, existen también patrones de diseño de seguridad, que son esencialmente buenas prácticas puestas en forma de plantilla.
Los '**patrones de ataques**' se estudian y documentan para sistemas operativos, dispositivos de conectividad, sistemas de gestión de datos, usos y servicios de web, si bien aún no han sido establecidos para ataques específicos. Se emplean en la educación de desarrolladores, en la realización de evaluaciones de riesgo y la definición de modelos de amenaza para sistemas de software [4]. Hay numerosos trabajos destinados a enumerar patrones de ataques donde el objetivo es el software (más que los datos o las redes) [5].

2.3 Desarrollo de pruebas de sistemas seguros

En la referencia [6] se recomienda aumentar el equipo de pruebas de sistemas mediante la inclusión de pruebas independientes por un tercero desinteresado, dado que los atacantes no están en general influenciados por el conocimiento de diseño de sistemas o de mecanismos de protección de seguridad.
Las pruebas para descubrir defectos de diseño son difíciles de desarrollar. El grupo de pruebas (independiente o no) debe ser capaz de construir casos de prueba basados en el entendimiento de la psicología de los atacantes y el conocimiento de software típico, hardware, y otros tipos de defectos de sistemas. Las fuentes adicionales de información para el desarrollo de casos de prueba incluyen: (a) casos de mal uso y casos de abuso; (b) reportes del análisis de árbol de amenaza; (c) modelos de amenaza; (d) modelos de política de seguridad; (e) objetivos de seguridad; (f) requerimientos de seguridad del sistema.
La prueba de resistencia del diseño del sistema ante ataques debe incluir, como mínimo: (a) probar la capacidad del sistema para resistir a adivinación de contraseña, máscaras, etc.; (b) probar defectos transitorios, como una combinación insólita o la secuencia de acontecimientos, degradación del ambiente de operaciones (saturación temporal de la red, pérdida de control, cambios ambientales), o la pérdida temporal de sincronización inducida entre los componentes de un sistema; (c) pruebas "creativas".

3. Amenazas-Ataques

3.1 Categorías y Fuentes de Amenazas

Los ataques orientados a explotar vulnerabilidades de software han aumentado exponencialmente con la coincidencia de proliferación de sistemas intensivos de software, servicios accesibles a través de la Internet, dispositivos como teléfonos móviles y sistemas de

posicionamiento global. Se espera que todos estos sistemas funcionen continuamente, sin interrupciones. Las amenazas a este funcionamiento son muchas y de muy variadas fuentes. A continuación, en la Tabla 1 se presenta una categorización simple de las mismas, y luego un conjunto inicial de fuentes identificadas.

Tabla 1. Categorías de amenazas.

Categoría de amenaza	Descripción	Propiedad comprometida
Sabotaje	La ejecución del software se suspende o termina, o su funcionamiento se degrada, o el ejecutable se suprime o destruye	Disponibilidad
Cambio de versión	El software ejecutable se modifica intencionadamente (cambio o corrupción) o se substituye por parte no autorizada, o se inserta en el ejecutable lógica no autorizada (código malicioso)	Integridad
Intercepción	Una entidad no autorizada accede al software o a una función restringida dentro del mismo	Control de acceso
Descubrimiento	Se revelan aspectos tecnológicos del software y detalles de puesta en práctica empleando ingeniería reversa (por ej.: descompilación, desmontaje)	Confidencialidad

Las "fuentes de amenazas" al software se pueden agrupar en las siguientes categorías generales: (a) atacantes externos; (b) personas maliciosas que intencionadamente abusan del software; (c) usuarios, no maliciosos, que intencionadamente emplean mal el software, sea porque encuentran limitaciones del uso correcto para obtener lo que están buscando, o simplemente por curiosidad sobre cómo el software podría responder a ciertas entradas o acciones; (d) usuarios benignos que por casualidad (generalmente un malentendido) incluyen una entrada o realizan una acción que duplica un patrón de ataque (incidentes accidentales), y si bien son involuntarios, su resultado es el mismo que el de ataques intencionales.

3.2 El Modelado de Amenazas ('Threat Modeling' - TM)

La tarea de identificar los riesgos y las amenazas en el software requiere un esquema estructurado y repetible, que mejore el entendimiento de los involucrados en el desarrollo, ayude a identificar problemas de diseño e integración de componentes, y sirva como input para el equipo de pruebas. El TM 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.
El **proceso de TM** consta de las siguientes etapas [7]: (1) conformar un grupo de análisis de riesgo; (2) descomponer la aplicación e identificar componentes claves; (3) determinar las amenazas a cada componente de la aplicación; (4) asignar un valor a cada amenaza; (5) decidir cómo responder a las amenazas; (6) identificar las técnicas y tecnologías necesarias para mitigar los riesgos identificados.
Si bien este proceso constituye un esquema estructurado, se reconoce [8] que el TM se presenta como un conjunto complejo y multidimensional de compensaciones que podrían ser hechas para encontrar un juego de objetivos implícitos o explícitos.

Desde **distintas dimensiones** pueden observarse diferentes enfoques o empleos:

- *Respecto al término* '*amenaza*': se utiliza tanto para referirse a los posibles "ataques" intencionales, como al "riesgo" que se corre cuando alguien se equivoca.
- *Respecto a las acciones*: mediante el modelado de amenazas se puede referir a una técnica para obtener requerimientos (qué amenazas se van a modelar) o a técnicas de análisis y diseño (cómo se analizan y reflejan esas amenazas en el modelo)
- *Respecto al foco*: el modelado puede ser '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 a determinar escenarios de amenazas) o 'centrado en el software' (análisis de software, redes de organización o sistemas y conlleva al modelado de protocolos y de amenazas de red).
- *Respecto a los recursos humanos*: el modelado de amenaza puede ser hecho por expertos de seguridad, en colaboración de ingenieros con expertos, o por ingenieros sin expertos disponibles.

Se concluye que el modelado de amenaza abarca una amplia variedad de actividades en la definición de requerimientos de seguridad y análisis de diseños de seguridad. No hay un único modo 'mejor' o 'correcto' de modelado de amenaza.

El objetivo de nuestra investigación es la incorporación de la seguridad como atributo transversal, especialmente en el desarrollo de aplicaciones típicas, por equipos de trabajo pequeños, de profesionales no expertos en ataques. Por ello, nuestro trabajo de análisis realizó un recorte (basado en estos aspectos) definiendo como marco: a) las amenazas se refieren a riesgos; b) se enfoca en las etapas de análisis y diseño; c) está centrado en los activos; d) es ejecutado por equipos de trabajo que conocen la aplicación a desarrollar pero no cuentan con expertos.
Además, nos hemos centrado inicialmente en las fases de requerimientos, diseño arquitectónico y diseño detallado, que son etapas tempranas y donde consideramos que los profesionales no expertos deben contar con apoyo para contemplar la seguridad.

4. Seguridad y los procesos de ciclo de vida del software

Una metodología de desarrollo de software que incorpore mejoras en el sentido de incluir la seguridad, debe proporcionar un marco integrado, o en algunos casos la dirección fase-por-fase para promover el desarrollo de la seguridad de software en todas las fases de ciclo de vida. La integración de actividades de seguridad en los procesos de ciclo de vida de la ingeniería de software implica la adición de prácticas de seguridad (ejemplo: pruebas de penetración) y de principios (ejemplo: un diseño que hace cumplir la menor parte de privilegio), a las actividades y prácticas (como la ingeniería de requerimientos, el modelado, y pruebas a base de modelo) en cada fase del ciclo de vida de desarrollo de software tradicional. Si bien ha habido una investigación activa al respecto, no existe una metodología de la ingeniería de software, que permita asegurar que la seguridad existe en el desarrollo de sistemas de software en gran escala. Las propuestas corrientes indican que la seguridad debería ser considerada desde las etapas tempranas del ciclo de vida de software [9].
Entendemos por 'Ciclo de Vida del Desarrollo de Software' (**SDLC** – por sus siglas en inglés: ***Software Development Life Cycle***) al proceso de crear sistemas de infor-

mación, mediante una conjunto de modelos y metodologías específicos, que describe las etapas de análisis de requerimientos, diseño, implementación, prueba y mantenimiento. Del seguimiento de este proceso debería resultar un sistema de información de alta calidad, acorde a los requerimientos del cliente.

4.1 Metodologías y Técnicas para Seguridad

Han aparecido metodologías que modifican actividades tradicionales de los SDLCs, o insertan nuevas actividades, con el objetivo de reducir el número de debilidades y vulnerabilidades en software y su creciente dependencia ante amenazas [10]. Algunas metodologías han sido usadas satisfactoriamente en proyectos de desarrollo de productos o en proyectos pilotos académicos. Se pueden destacar: (1) *Microsoft Trustworthy Computing SDL* [11]; (2) *Oracle Software Security Assurance Process* [12]; (3) *Comprehensive, Lightweight Application Security Process* -CLASP- [13]; (4) *Seven Touchpoints for Software Security* [14]; y (5) *Team Software Process for Secure Software Development* - TSP-Secure - [15].

4.2 Microsoft Trustworthy Computing SDL

Para la experiencia que se describe en este trabajo se ha seleccionado la metodología *Microsoft Trustworthy Computing SDL* - MTC-SDL -, establecida formalmente por Microsoft para su proceso de desarrollo de software mejorado en seguridad, diseñado de manera que pueda ser empleada en un sentido amplio para diferentes desarrollos y diferentes tecnologías. El método SDL se aplica a lo largo de seis fases, trazando un mapa de tareas relevantes de seguridad y entregables en el SDLC existente: requerimientos, diseño, implementación y desarrollo, verificación, puesta en producción, soporte y mantenimiento. Se lo ha seleccionado por su completitud, por los materiales de guía y formación publicados, por su adaptabilidad a distinto tipo de proyectos y por su independencia tecnológica. Propone que el diseño sea conducido por el Modelado de Amenazas, que permite una descripción textual detallada y una descripción gráfica de las amenazas significativas para el sistema que se está modelando; este modelo captura los caminos a partir de los cuales las funciones del software y la arquitectura pueden ser apuntadas e identifican a los agentes de amenaza potenciales. En este trabajo nos hemos enfocado en dos técnicas:

1. ***STRIDE/DREAD Model-V1-2004***: ayuda a definir potenciales escenarios de amenazas desde la perspectiva del atacante, y se complementa con una metodología de cálculo de riesgo que encapsula respuestas para las potenciales preguntas sobre riesgo. Ayuda a ponderar y priorizar la importancia de sus contramedidas y mitigaciones [16].
2. ***Microsoft Threat Analysis and Modeling*:** simplifica y cambia la perspectiva del atacante al defensor, considerando que el usuario se identifica estrechamente con amenazas, más que con ataques, y que el defensor puede entender mejor las amenazas al sistema que el atacante. Para su implementación se han generado herramientas, de cuales hemos seleccionado para realizar el análisis:
   - ***Microsoft Threat Analysis & Modeling v2.1 (2006) - TAM***: Herramienta creada para facilitar la creación y asimilación de modelos de amenazas. Permite a los no expertos en seguridad generar un modelo basado en información ya conocida, presente

en el desarrollo de todo sistema: requerimientos de negocios y arquitectura de la aplicación. A partir de la información de estos modelos, la herramienta identifica automáticamente las amenazas ha ser consideradas y permite generar un conjunto de artefactos de seguridad de valor: matrices de control de acceso, diagramas de flujo de datos basados en casos de uso, superficies de ataque, y reportes variados.

- ***Threat Modeling Tool (2009) - TMT**:* Herramienta que permite crear documentos de modelos de amenazas para aplicaciones, organizando la información relevante del modelo (puntos de entrada, activos, niveles de confianza, DFDs, amenazas y árboles de amenaza y vulnerabilidades) dentro de una vista simple e integrada. Permite exportar la documentación como XML, HTML y MHT.

Ambos son procesos iterativos, iniciando con un modelo de amenaza de alto nivel y con progresos de diseño detallado en las fases subsecuentes del ciclo de vida. Una de las versiones incorpora bibliotecas de ataque predefinidas que describen mitigaciones eficaces a cada tipo de ataque asociado con cada amenaza, autogenerando modelos de amenaza basados en un contexto definido de aplicación. El modelo entonces traza un mapa de aquellos modelos de amenaza a contramedidas relevantes.

5. Aplicación del MTC-SDL en el modelado de un caso

Para mostrar la aplicación del MTC-SDL y los aportes del mismo en las actividades de análisis y diseño, se ha seleccionado una aplicación web típica, a la que se la han realizado recortes a fin de poder exhibir resultados y mostrarlos de manera visible en las capturas de pantallas. El caso considerado en este análisis consiste en un Sistema Proveedor de Servicios de distribución de contenidos académicos (textos, apuntes de cátedra, artículos de revistas científicas, investigaciones, etc.), en formato digital con opciones de impresión. El sistema está basado en una arquitectura de tres capas, donde el cliente debe contar con un navegador de Internet ('browser') para poder interactuar con el mismo. También se requiere de espacio disponible en la Internet para montar dicho sitio y una aplicación de base de datos. Se debe contar con una base de datos de los clientes y de sus respectivas cuentas, y con una base de datos de publicaciones (libros y documentos) disponibles.
En el diagrama de componentes de esta aplicación se identifican dos Servidores que se comunican: un Servidor de Bases de Datos (que comprende la Base de Datos de Publicaciones y la Base de Datos de Clientes y Cuentas) y un Servidor Web.

En este análisis nos centraremos en la fase de requerimientos, identificando requerimientos funcionales principales y requerimientos de seguridad (funcionales y no funcionales), dejando a un lado otro tipo de requerimientos; y en la fase de diseño y modelado de amenazas de una porción del sistema completo de la librería on-line, específicamente la sección de 'Autenticación de un Usuario'.
En primer lugar, se presentan resultados obtenidos al emplear la herramienta TAM como ayuda para documentar los requerimientos de la aplicación y su arquitectura, para luego poder utilizarla en futuras etapas del desarrollo de software. Luego, se presentan alternativas usando la herramienta TMT.

5.1 Aplicando TAM

Para comprender el contexto de la aplicación, se deben definir los elementos individua-

les, así como la interacción entre ellos para definir el contexto de la aplicación. Esto hace posible la identificación de amenazas potenciales y la decisión de estrategias de seguridad efectivas. Al definir el contexto de la aplicación, es necesario definir los requerimientos y luego la arquitectura. Los primeros consisten en objetivos de negocios, roles de usuarios, datos y casos de uso, mientras que la arquitectura consiste en componentes, roles de servicios, dependencias externas y llamadas.
También se pueden asignar prioridades a cada una de las amenazas y especificar cada una de las respuestas, para así conformar la base de datos de amenazas.
A partir de los requerimientos y la descripción de la arquitectura, la herramienta trata de identificar de manera automática las amenazas, al tiempo que produce una serie de elementos como son: (1) matrices de acceso a datos; (2) casos de uso; (3) diagramas de flujos de datos, de llamada y de confianza; (4) superficie de ataque; y (5) informes.

A continuación se muestran capturas de pantalla obtenidas en distintos momentos de trabajo realizado sobre el caso de análisis con la herramienta TAM.

La Figura 1 muestra resultados del primer paso de descripción y modelado de la aplicación a desarrollar. En el panel de la izquierda puede observarse el árbol con el que esta aplicación organiza los elementos del modelo (roles de usuario - **a** -, almacenes de datos - **b** -, componentes - **c** -, dependencias externas y casos de uso - **d** -). A la derecha se encuentra el formato de la definición de un componente seleccionado ('Servidor Web'), donde el analista de requerimientos puede indicar datos básicos (nombre, descripción, roles asociados) así como también aspectos de 'alto nivel' relacionados a la implementación (tipo de servicio y tecnologías). Además, se pueden seleccionar (de una lista ofrecida como menú) las relevancias - **e** - del componente (atributo que define su comportamiento); que permite vincular los 'ataques' con los 'componentes' a través de librerías de ataques específicas y constituye el medio por el cual se comienza a incorporar la seguridad en la etapa temprana de relevamiento.
Luego de definir los componentes, se pasa a especificar los requerimientos mediante Casos de Uso, los cuales consisten de un conjunto de llamadas (calls). Cada llamada está compuesta por una entidad llamante (un rol de usuario o de servicio o un componente), y un componente que recibe la misma. Además, se permite especificar los datos enviados y recibidos, y el efecto sobre ellos. La definición precisa de las llamadas es muy importante, ya que la herramienta genera a partir de ellas una 'propuesta' de amenazas a evaluar, respecto a Confidencialidad, Integridad y Disponibilidad.
Para mostrar los aportes de la herramienta, como ejemplo analizamos el caso de uso 'Autenticación de un Usuario'. El proceso de autenticación lo inicia el usuario anónimo que llega al sitio y desea identificarse frente al mismo, para lo cual deberá proveer sus credenciales (en este caso, nombre de usuario y contraseña).

Definidas las llamadas, el caso de uso se visualiza como indica la Figura 2. Según se observa a la derecha, hay opciones para obtener una vista con el formato habitual de flujo de datos, o solicitar que se exhiban los aspectos de seguridad. Esta información se genera automáticamente a partir de los datos definidos anteriormente.
Una vez identificados los componentes claves de la aplicación, se debe examinar cuáles pueden ser las amenazas a las que está sujeto cada uno, para lo cual la metodología propone utilizar el método ***STRIDE per element*** para reconocer y categorizar las amenazas asociadas a la aplicación. La herramienta permite generar automáticamente las amenazas, que se clasifican en tres categorías: (a) amenazas de confidencialidad, (b) amenazas de integri-

dad, y (c) amenazas de disponibilidad. Las amenazas se pueden seleccionar de librerías predefinidas, o también crear librerías nuevas. Es posible entonces, definir automáticamente los ataques potenciales que se pueden usar para una amenaza dada. En la Figura 3 se presenta una vista de la definición de la amenaza 'Revelación no autorizada de las credenciales', identificada como una amenaza de Confidencialidad para el caso de uso seleccionado.

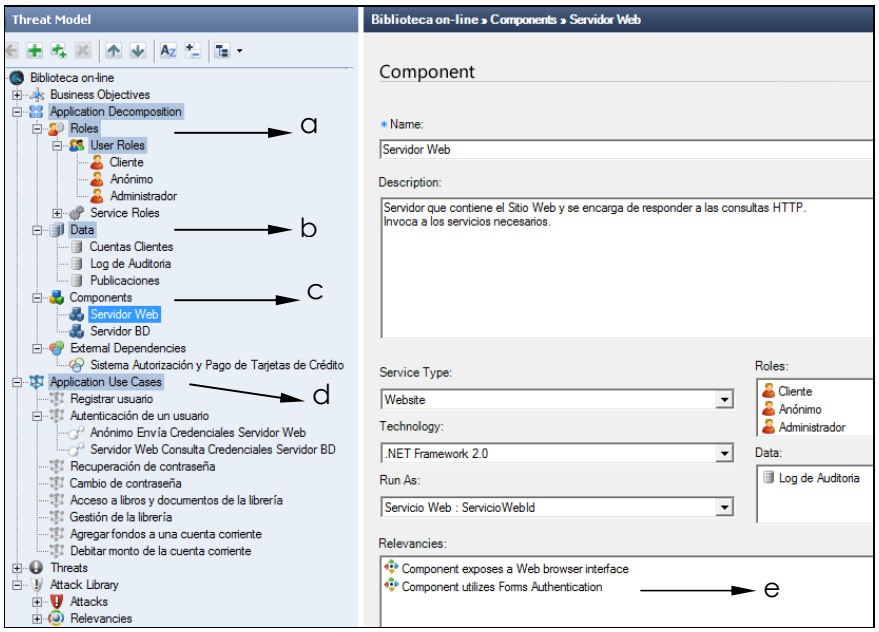

Fig. 1. Modelo-Componentes.

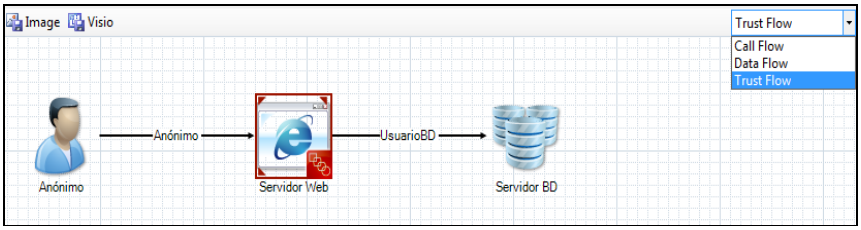

Fig. 2. Caso de Uso: 'Autenticación de un Usuario'

Además, al identificar una amenaza es necesario 'ponderarla' para poder: (1) establecer el riesgo de cada una de ellas, (2) planificar prioridades y establecer contramedidas, así como fijar criterios de aceptación de riesgos. La herramienta permite utilizar el ***Método DREAD*** que pondera a través de cinco propiedades evaluadas por separado (daño potencial, reproducibilidad, facilidad de explosión, usuarios afectados, facilidad de descu-

brimiento), y luego obtener el valor de riesgo de la amenaza como un promedio. En la Figura 3, ya se ha presionado el botón del ítem 'Calcular Riesgo' y se puede observar la ventana que se abre para cuantificar los atributos y realizar el cálculo. En cuanto a las contramedidas, las que figuran en esta vista son las asociadas automáticamente por la aplicación y están relacionadas a los componentes por las relevancias, pero el usuario puede manualmente agregar otras contramedidas de la librería que crea necesarias.

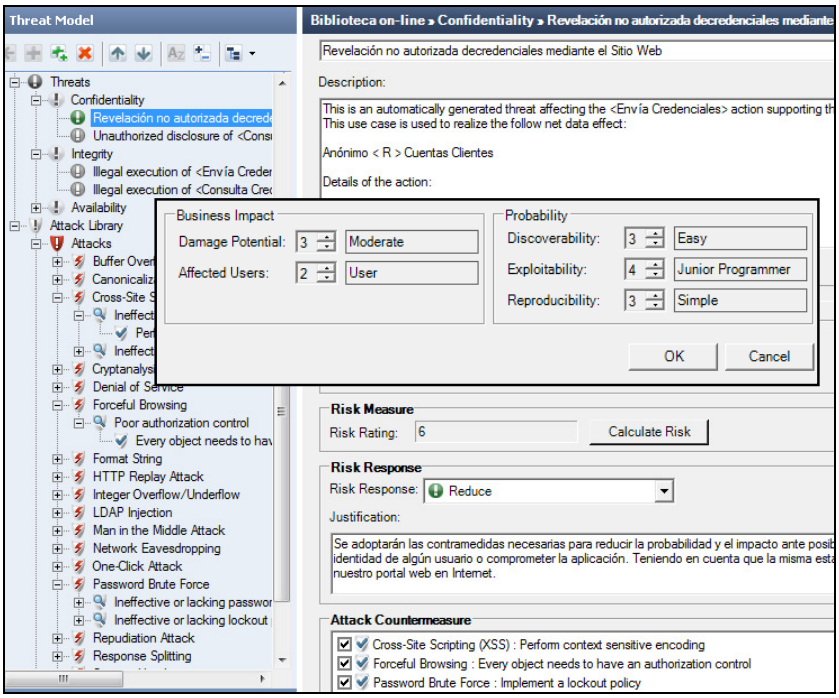

Fig. 3. Generación de la amenaza 'Contraseña débil'.

Es de destacar que la identificación de amenazas y la asignación de los valores para calcular el riesgo ponderado están sujetos a la capacidad del analista. El aporte de la herramienta, al disponer de librerías e ir ofreciendo ítems para seleccionar, facilita la tarea cuando se trata de un analista no experto, pero no suple su decisión, por lo que resulta conveniente la toma de conciencia y la capacitación de los equipos para trabajar en el desarrollo de software seguro. Se debe observar que hay espacios para indicar descripción y justificación, que permite documentar las decisiones y así facilitar la trazabilidad.

Con la información consignada al generar una amenaza, es posible obtener también el correspondiente Árbol de Ataques, donde la raíz es la amenaza que se seleccionó y cada rama constituye un camino para su potencial explosión. La Figura 4 constituye una vista parcial del árbol de amenaza generado automáticamente para la amenaza 'Revelación no autorizada de las credenciales'(se presenta más de 10 posibles ataques en el primer nivel de árbol y más de 25 contramedidas en el último).

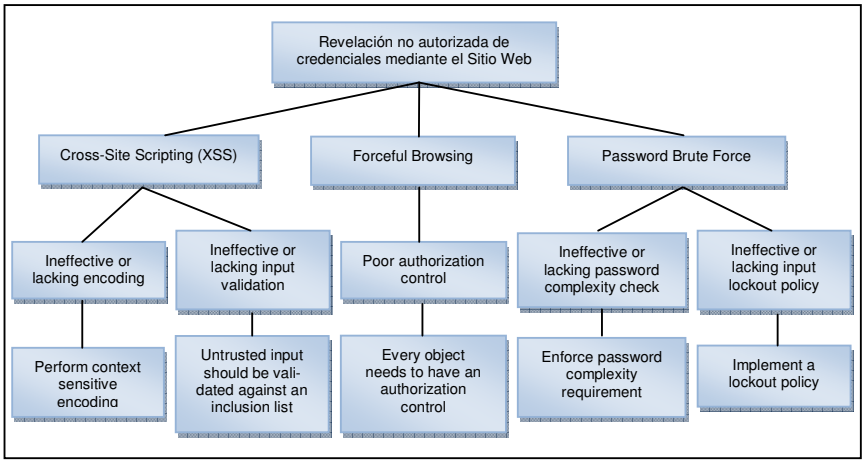

Fig. 4. Árbol de ataques de la amenaza 'Revelación no autorizada de las credenciales'

Una vez identificadas las amenazas, el paso siguiente es proponer contramedidas efectivas. Como se observa en el cuadro izquierdo de la Figura 3, al seleccionar una amenaza se presenta al final de la pantalla una lista de posibles contramedidas para mitigar ese potencial ataque. En este caso, se ha seleccionado la contramedida de realizar controles de acceso sobre todos los objetos ("Every object needs to have an authorization control") y se despliega automáticamente un cuadro que la describe, como en la Figura 5.

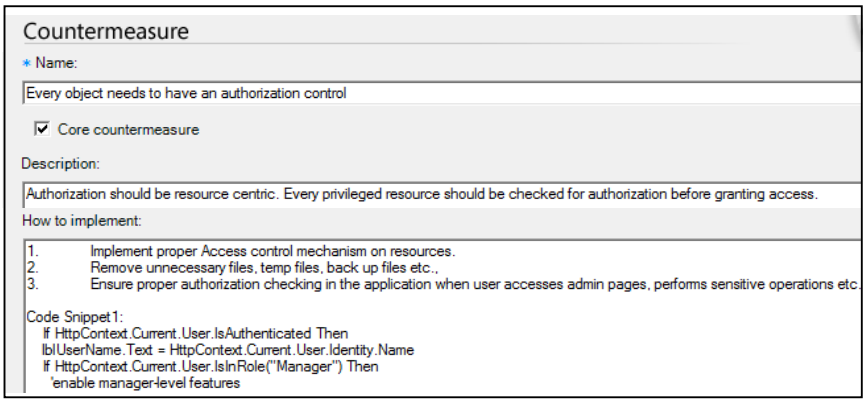

Fig. 5. Mitigación de riesgos/Contramedidas.

Es posible también **generar reportes**, destinados a diferentes miembros del equipo, que constituyen documentación de aporte para las etapas siguientes del ciclo. Éstos pueden ser reportes destinados a: responsables de riesgos (amenazas y respuesta a las mismas), el equipo de diseño, el equipo de desarrollo (implementaciones sugeridas para las contramedidas), el equipo de testing (procedimientos o casos para testear los componentes, según los ataques a los que están expuestos y las amenazas definidas en

el modelo) y para el equipo de operaciones. Estos reportes pueden "customizarse" y proporcionan información resultante del modelo de amenazas, resumida y organizada, que orienta a diferentes actividades del proceso de desarrollo del software.

5.2 Aplicación de TMT

Una alternativa de modelado puede ser el uso de DFD, una representación gráfica que agiliza el proceso de modelado de requerimientos y al mismo tiempo es bien conocido por los profesionales de la ingeniería de software. La herramienta TMT da soporte para ello, con adaptaciones de manera que: los componentes se representan como Burbujas de Proceso, los roles de usuarios y dependencias externas como Entidades, y los almacenes de datos en forma directa. Además de los componentes típicos se han agregado los límites de confianza. A fin de determinar cuáles son las amenazas es preciso conocer: (a) cuáles son los puntos de entrada, (b) los niveles de confianza, y (c) los activos de mayor interés. En nuestro caso de análisis, el Diagrama del Modelo, enriquecido con los límites de confianza de datos y de equipos, tiene la vista que se muestra en la Figura 6.

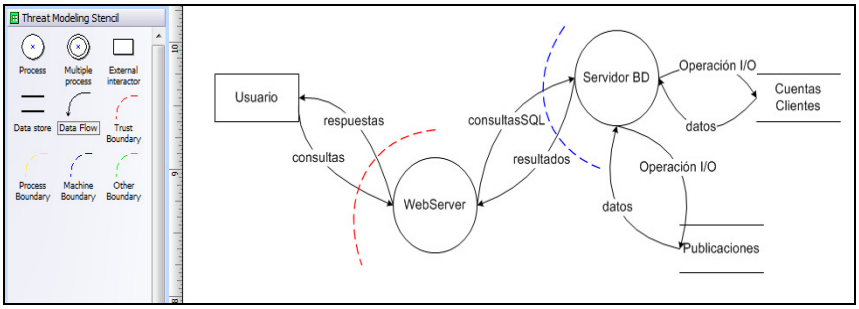

Fig. 6. DFD con límites de confianza.

Elemento	**S**poofing Identity	**T**ampering with Data	**R**epudia-tion	**I**nformation Disclosure	**D**enial of Service	**E**levation of Privilege
Entidad Externa	*		*			
Proceso	*	*	*	*	*	*
Almacén de datos		*	X	*	*	
Flujo de datos		*		*	*	

Fig. 7. Matriz de STRIDE.

A partir del diagrama, la herramienta produce un análisis automático por cada elemento de la lista de componentes definidos en el diseño, mediante el método STRIDE, tomando como base la siguiente matriz de 'Identificación STRIDE de amenazas por tipo de elemento' (Ver Figura 7).

Para el diagrama de la Figura 6, si se selecciona que se genere el Modelo de Análisis, resulta una vista como la que se muestra en la Figura 8, donde a la izquierda se tiene un

árbol completo de los elementos que caracterizan el sistema (entidades, procesos, almacenes y flujos), y para cada uno de ellos una expansión (en un nivel) de las categorías de amenazas provistas por STRIDE. Seleccionando un ítem (hoja del árbol) a la derecha se presenta automáticamente un conjunto de preguntas que el analista puede formularse como ayuda para identificar las amenazas. Además, dispone de una plantilla para poder definir una amenaza e indicar el impacto y la mitigación, de manera textual descriptiva. Si bien los detalles de la amenaza son menos precisos, la ayuda para su identificación es de nivel básico para un analista no experto y permite documentar las cuestiones de seguridad desde el inicio y de una forma simple que pueda ser comunicada al equipo.

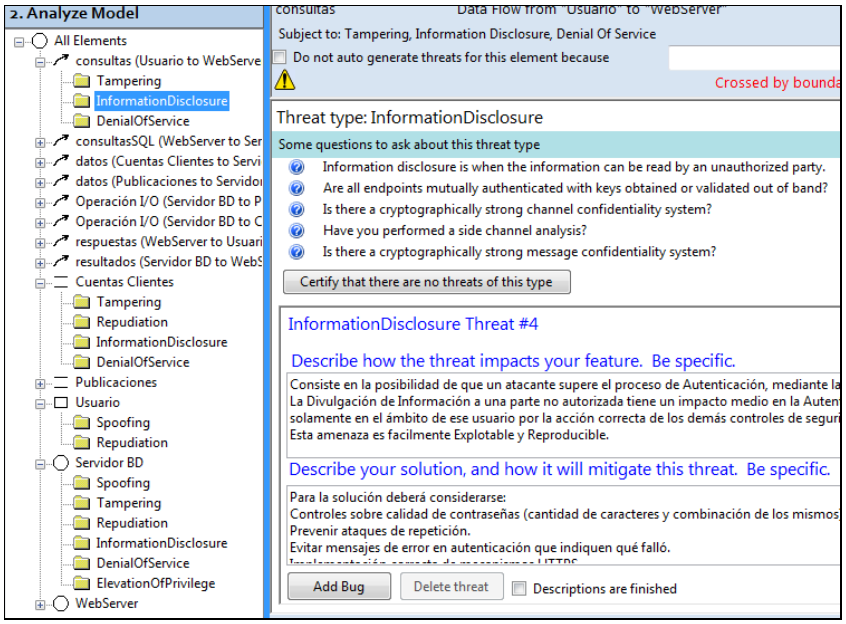

Fig. 8. Amenaza en TM.

6. Conclusiones

El éxito de implementar un modelado de amenazas pasa por realizar sucesivas iteraciones durante la evolución del proyecto y favorecer la reutilización de componentes que puedan servirnos para posteriores mejoras o para su uso en otros proyectos que tengan cierta similitud (en especial aplicaciones web). Las revisiones y mejoras del modelo se deben ir incorporando a medida que se produzcan cambios en el entorno de la aplicación o del sistema, su diseño, implementación, configuración y, por supuesto, cuando se modifiquen o aparezcan nuevos casos de uso. La incorporación de medidas de seguridad en el desarrollo de software puede no parecer rentable a corto plazo, pero las malas prácticas terminan demostrando que a medio-largo plazo esta percepción es errónea. Por ello, el uso de este enfoque viene a desempeñar un papel fundamental en todo ciclo de

vida de desarrollo que aspire a considerarse seguro.
Cuanto antes la comunidad incorpore metodologías y técnicas como las analizadas en este trabajo, mayor será su impacto. Si bien la disciplina de Ingeniería de Software Seguro está en amplia evolución y crece la comunidad de expertos, es necesario que los equipos actuales de diferentes tipos de proyectos comiencen a incorporarlo y la formación de recursos atienda estas cuestiones.

Referencias

1. 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.
2. Microsoft Corporation. "Security Engineering Explained". Remond,WA; Microsoft Corporation Patterns and Practices Developer Center. USA. 2005. http://msdn2.microsoft.com/en-us/library/ms998382.aspx
3. Hope P. and McGraw G. and Anton A. "Misuse and Abuse Cases: Getting Past the Positive". IEEE Security and Privacy. 2004. http://www.cigital.com/papers/download/bsi2-misuse.pdf
4. US CERT, Attack Patterns. US CERT. https://buildsecurityin.us-cert.gov/daisy/bsi/articles/knowledge/attack.html
5. Hoglund G., McGraw G."Exploiting Software: How to Break Code".Addison-Wesley. 2004.
6. Herrmann D. "A Practical Guide to Security Engineering and Information Assurance". Auerbach Publications. 2002.
7.Milano P. "Seguridad en el ciclo de vida del desarrollo de software". Argentina. www.cybsec.com/upload/cybsec_Tendencias2007_Seguridad_SDLC.pdf
8. Shostack A. "Experiences Threat Modeling at Microsoft". 2008. http://blogs.msdn.com/sdl/archive/2008/10/08/
9. Mouratidis H. "Integrating Security and Software Engineering". Idea Group Publishing. Inc. 2007.
10. N. Davis. "Secure Software Development Life Cycle Processes". US-CERT. 2006. https://buildsecurityin.us-cert.gov/daisy/bsi/326.html
11. 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.
12. R. Shores. "Oracle Software Security Assurance". http://www.oracle.com/security/software-security-assurance.html
13. CLASP. http://www.securesoftware.com/CLASP
14. McGraw, G. "Software Security: Building Security In". Addison-Wesley. 2006.
15. Over, J. "TSP for Secure Systems Development". Presentation at CMU SEI. USA. http://www.sei.cmu.edu/tsp/tsp-secure-presentation/
16. Meier J., Mackman A., Dunner M., Vasireddy S., Escamilla R., Murukan A. "Improving Web Application Security: Threats and Countermeasures". Microsoft Corporation. 2003. http://msdn.microsoft.com/en-us/library/aa302419.aspx