MODELADO DE AMENAZAS, UNA TÉCNICA DE ANÁLISIS Y GESTIÓN DE RIESGO ASOCIADO A SOFTWARE Y APLICACIONES.

Barba Olivares, Gabriel Eduardo.
gbarba_olivares@hotmail.com
Universidad Piloto de Colombia

Resumen—El presente documento describe la técnica de análisis y gestión de riesgo modelado de amenazas, la cual se asocia con el entorno de desarrollo, software seguro y aplicaciones.

Dentro de los principales objetivos se encuentra responder a preguntas como ¿qué es el modelado de amenazas y cuál es su finalidad?, además de detallar cada etapa del proceso de implementación, mediante el uso de recursos metodológicos y herramientas de apoyo proporcionadas por las principales entidades impulsoras de la técnica.

Índice de Términos—Amenaza, requerimiento de seguridad, riesgo, vulnerabilidad.

Abstract—This paper describes the risk analysis technique known as threat modeling, a technique associated with the secure software development environment and applications.

In the most important aspects, it is possible to answer the following questions: what is the threat modeling and what is its purpose? In addition, detailed information is specified for each stage on the implementation of this technique, based on the example and backed by methodological resources and tools Support provided by leading entities promoting this technique.

I. INTRODUCCIÓN

En la actualidad, el software inseguro es uno de los retos más grandes en lo que refiere a la seguridad, algunas estadísticas muestran que el 55 % de los sitios web presentan problemas de vulnerabilidades graves con tendencia al aumento.

Se estima que las vulnerabilidades han aumentado un 9% en los últimos años, debido a que las empresas exigen ciclos de lanzamiento de aplicaciones más rápidos dejando a veces a un lado lo referente a comprobaciones de seguridad.

Por ésta razón, cada día se hace más importante el pensamiento de un desarrollo de software seguro; en este sentido, la comprobación o pruebas de seguridad ha demostrado ser un elemento clave para cualquier organización que necesite confiar en el software que produce o usa.

En la actualidad, se pueden encontrar diferentes iniciativas cuyo objetivo se centra en la búsqueda y lucha contra las causas de software inseguro, dentro de las cuales se destacan: Microsoft SDL, BSIMM (Building Security In Maturity Model), SSE CMM (System Security Engineering Capability Matury Model), Owasp (Open Web Application Security Project); cada entidad proporciona diferentes herramientas, documentación y estándares.

II. SISTEMAS SEGUROS

Normalmente el software hace parte de un elemento más grande de un sistema, es por ello que la ingeniería de los sistemas ha evolucionado elaborando publicaciones y técnicas orientadas a la construcción de sistemas seguros.

Todo esto 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 presencia de ataques, errores del sistema o errores humanos.

Las etapas de ingeniería involucradas en la construcción de sistemas seguros, son:

- El desarrollo de requerimientos de seguridad.
- El desarrollo de diseños de sistemas seguros.
- La integración de componentes.
- Las pruebas de sistemas y subsistemas.

A. Desarrollo de requerimientos de software seguro

En el desarrollo de software, se conoce que la gran mayoría de los problemas de seguridad provienen de la especificación inadecuada o inexacta de los requerimientos o en una mala interpretación de los mismos.

Comúnmente los requerimientos para la funcionalidad de seguridad son confundidos con los requerimientos para el software seguro.

Los primeros incluyen las funciones que ponen en práctica las políticas de seguridad en las aplicaciones, por ejemplo: Funciones de control de acceso, identificación, autenticación, autorización, funciones que realizan el cifrado y el descifrado, gestión de claves etc. Se conocen también como "requerimientos de servicios de seguridad".

Por otro lado, lo segundo afecta directamente a la probabilidad de que el software en sí mismo es seguro, busca asegurar que el sistema permanezca funcionando correctamente incluso cuando se encuentre bajo amenaza; de igual manera, está orientado a la reducción o eliminación de vulnerabilidades. Se conocen como "objetivos de seguridad".

Durante la obtención de requerimientos se debe recolectar información relativa a la seguridad como lo son:

- Determinar y valorar los activos, definiendo qué se va a proteger (activos tangibles o intangibles) y determinando su valor (económico y no económico).
- Determinar los actores, dueños de los datos.
- Determinar los casos de uso y roles de cada actor.
- Determinar los requerimientos legales y de negocios (restricciones legales y de negocios, requerimientos de auditoría y no repudio, políticas de seguridad).
- Determinar las amenazas sobre los activos identificados, su probabilidad de ocurrencia y la política de manejo.
- Definir la arquitectura del sistema, entendiendo cómo se integran los mecanismos de seguridad con el sistema, cómo se maneja la confianza, y cómo se comportará el sistema frente a un ataque.

B. Desarrollo de diseños de sistemas seguros

En el diseño de sistemas seguros, se deben incorporar varios elementos claves de diseño para orientar la identificación de vulnerabilidades típicas de sistemas. Por ejemplo, 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 son 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 la puesta en marcha de buenas prácticas.

C. Desarrollo de pruebas de sistemas seguros

Las pruebas para descubrir defectos de diseño normalmente son difíciles de desarrollar; éstas deberían estar orientadas a la construcción de casos de prueba basados en la simulación de un atacante y el conocimiento de software y hardware típico, además de otros tipos de defectos de sistemas. En éstas pruebas se podría llegar a encontrar la siguiente información:

- Casos de mal uso y casos de abuso.
- Reportes del análisis de árbol de amenaza.
- Modelos de amenaza.
- Modelos de política de seguridad.
- Objetivos de seguridad.
- Requerimientos de seguridad del sistema.

III. CATEGORÍAS Y FUENTES DE AMENAZAS

Continuamente se observa que los ataques orientados a explotar vulnerabilidades de software, han aumentado considerablemente a causa del aumento de servicios accesibles a través de internet

o dispositivos como teléfonos móviles. Idealmente se espera que las aplicaciones funcionen continuamente sin interrupción, sin embargo, las amenazas a este funcionamiento son numerosas y vienen dirigidas desde diferentes fuentes.

TABLA I
CATEGORÍAS DE AMENAZAS

Categoría de amenaza	Descripción	Propiedad comprometida
Sabotaje	La ejecución del software se suspende o termina, o su funcionamiento se degrada.	Disponibilidad
Cambio de versión	El software ejecutable se modifica intencionadamente (cambio o corrupción) o se substituye por parte no autorizada.	Integridad
Intercepción	Una entidad no autorizada accede al software.	Control de acceso
Descubrimiento	Se revelan aspectos tecnológicos del software .	Confidencialidad

La anterior tabla referencia las categorías más comunes de amenazas, descripción y propiedad comprometida. Fuente: RAMOZ, Juan. Aplicar el modelo de amenazas para incluir la seguridad en el modelado de sistemas.

Dentro de las principales fuentes de ataques se encuentran:

- Atacantes externos.
- Personas maliciosas que intencionadamente abusan del software.
- Usuarios no maliciosos, que intencionadamente emplean mal uso del software, sea porque encuentran limitaciones para obtener lo que están buscando o simplemente por curiosidad sobre cómo el software podría responder a ciertas entradas o acciones.
- Usuarios benignos que por casualidad (generalmente un malentendido) incluyen una entrada o realizan una acción que emula un patrón de ataque (incidentes accidentales), y si bien son involuntarios, su resultado es el mismo que el de ataques intencionales.

IV. CICLO DE VIDA DE DESARROLLO DE SOFTWARE SEGURO

Existen diferentes definiciones acerca del aseguramiento de software (Software Assurance) establecidos por diferentes entes u organizaciones, pero todas ellas presentan un criterio común "la seguridad se debe presentar durante todo el ciclo de vida de desarrollo de software". Un software seguro es diseñado, implementado, configurado y operado buscando que cumpla algunas características esenciales, tales como que continúe operando en presencia de ataques, aislé o mitigue el daño y que se recupere lo antes posible.

¿Pero exactamente que es el ciclo de vida de desarrollo seguro, o como bien se conoce por sus siglas SDLC?, la fase conceptual de un ciclo común de construcción de software está compuesta por definir, diseñar, organizar, implementar y mantener; el SDLC busca la integración de componentes de seguridad en cada una de éstas fases con el objetivo de lograr asegurar un programa de seguridad rentable y exhaustivo.

V. TÉCNICAS DE COMPROBACIÓN

Durante todo el SDLC existen diferentes técnicas de comprobación que pueden ser implementadas con el fin de validar la seguridad en la construcción de las aplicaciones, cada una de estas contiene una serie de ventajas y desventajas que se presentan a continuación:

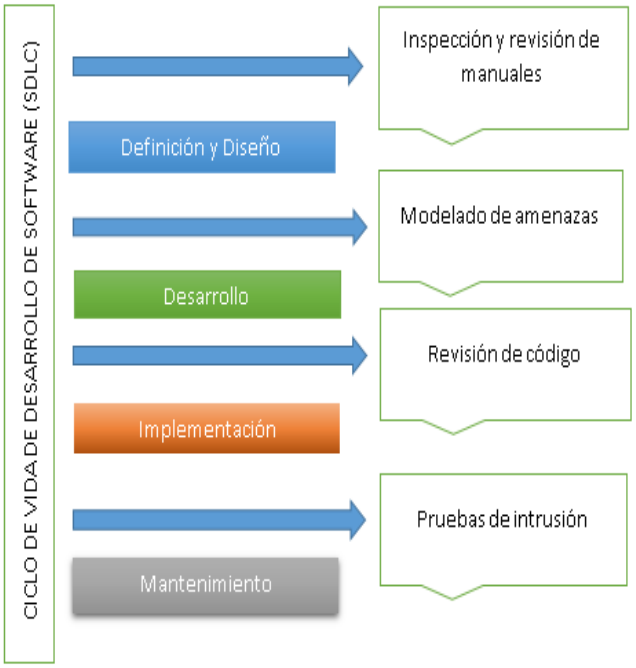


Fig. 1. Técnicas de comprobación utilizadas en las fases del ciclo de vida de desarrollo de software seguro (SDLC). Fuente: El autor.

TABLA II
VENTAJAS Y DESVENTAJAS DE LAS TÉCNICAS DE COMPROBACIÓN

Técnica	Ventajas	Desventajas
Inspección de manuales	No requiere tecnología de apoyo flexible, fomenta el trabajo en grupo, se aplica en una fase temprana del SDLC	Puede consumir mucho tiempo y el material de apoyo no siempre es disponible. Precisa de bastante conocimiento, reflexión y competencia humana.
Modelado de amenazas	Visión práctica del sistema desde el punto de vista de un atacante, flexible, se aplica en fase temprana del SDLC.	Técnica relativamente nueva, unos buenos modelos de amenaza no se traducen en un buen desarrollo.
Revisión de código	Eficacia e integridad, precisión, rapidez (para revisores competentes).	Requiere desarrolladores de seguridad que sean altamente competentes, no puede detectar errores en tiempo de ejecución con facilidad, el código fuente en uso puede ser diferente del que está siendo analizado.
Pruebas de intrusión	Puede ser rápido y por lo tanto económico, requiere un conocimiento relativamente menor que una revisión de código fuente, comprueba el código que está siendo expuesto.	Se ejecuta en una fase tardía del SDLC.

La anterior tabla referencia las ventajas y desventajas que existen con cada técnica de comprobación, aplicadas al ciclo de vida de desarrollo de software seguro (SDLC). Fuente: OWASP. Guía de pruebas OWASP.

VI. ¿QUÉ ES MODELADO DE AMENAZAS Y CUÁL ES SU FINALIDAD?

El modelado de amenazas es una técnica de comprobación cuyo objetivo es ayudar a identificar y planificar de forma correcta, la mejor manera de mitigar las amenazas de una aplicación mediante un enfoque moderno de análisis de gestión de riesgos, y la implementación de medidas o controles que contribuyan a mejorar la seguridad.

Está técnica tiene como principales ventajas la aplicación durante una etapa temprana del SDLC, y mas allá de que su implementación es relativamente nueva está despertando un gran interés.

En la actualidad, Microsoft es uno de los principales impulsores de esta técnica, aplicando su visión y proporcionando diferentes tipos de enfoque y herramientas. Otras entidades, como por ejemplo OWASP y expertos en reconocidas empresas de seguridad a nivel internacional, opinan que la iniciativa se encuentra muy bien encaminada.

Básicamente, esta técnica consiste en revisar el diseño y/o la arquitectura siguiendo una metodología que ayude a identificar y corregir los problemas de seguridad actuales o futuros. La idea es que los diferentes actores que participan en la construcción de una aplicación, como los desarrolladores, tester, gerencia, administradores de sistemas, consultores y auditores; intervengan en el proceso de identificación de posibles amenazas, poniéndose en el papel de una atacante y tomando actitud pro activa frente a éstas.

La relación de un modelo de amenazas ayuda a identificar y cumplir los objetivos de seguridad de cada entorno, facilitando la clasificación de tareas con base con el nivel de riesgo resultante.

VII. CONSIDERACIONES A TENER EN CUENTA

Es importante tener en cuenta que un pequeño fallo de seguridad puede llegar a comprometer todo el sistema, por eso el modelo de amenazas debe apuntar a seguir una metodología que esté en constante evolución; gracias a esto, se puede identificar y mitigar el mayor número posible de amenazas y vulnerabilidades en una fase temprana del ciclo de vida de desarrollo de software:

El modelo a implementar, debe ser capaz de:

- Identificar amenazas potenciales, así como las condiciones necesarias para que un ataque se logre llevar a cabo con éxito.
- Facilitar la identificación de las vulnerabilidades que una vez eliminadas o contrarrestadas, puedan llegar a mitigar las posibles amenazas.
- Proporcionar información relevante, sobre cuáles serían los controles más eficaces para contrarrestar un posible ataque y como prevenir la consecución de éstos.
- Transmitir a la gerencia la importancia de los riesgos tecnológicos en términos de impacto de negocio.
- Facilitar la comunicación y promover una mejor concienciación sobre la importancia de la seguridad.
- Simplificar la posterior actualización mediante el uso de componentes reutilizables.

VIII. METODOLOGÍAS PARA MODELADO DE AMENAZAS

En la actualidad existen diversas metodologías para el modelado de amenaza, las cuales intentan ayudar en la medición y mitigación de los riesgos implícitos durante el desarrollo de software; dentro de las más conocidas se encuentran:

- Ace Threat Analysis and Modeling (Microsoft).
- CORAS (Consultative Objective Risk Analysis System).
- Trike.
- PTA (Practical Threat Analysis).

IX. ACE THREAT ANALYSIS AND MODELING

Para el presente documento, se ha escogido la metodología "Ace Threat Analysis and Modeling" establecida por Microsoft, la cual está basada en una combinación de ideas propias, y cuya metodología ha ido evolucionando y recogiendo ideas con diversos enfoques.

En sus primeras versiones, se buscó construir un árbol donde se mapee los principales vectores de ataques y se pueda cruzar con las amenazas para posteriormente clasificarlas, de tal manera que sea posible priorizar las acciones necesarias para mitigar los riesgos.

Desde el punto de vista de un posible atacante se trata de identificar:

- Los puntos de entrada de la aplicación.
- Los activos a proteger.
- Los diferentes niveles de confianza.

Se establece cual es el nivel de seguridad del sistema:

- Mediante casos de uso.
- Conociendo las distintas dependencias.
- Utilizando modelos del sistema.

Se determina cuáles son las amenazas:

- Se identifican las amenazas.
- Se analizan y clasifican.
- Se identifican las vulnerabilidades a las que se ve expuesto el sistema.

X. ETAPAS DEL MODELADO DE AMENAZAS

Para llevar a cabo exitosamente el modelado de amenaza, el modelo de referencia indicado por Microsoft divide el proceso en cinco etapas esenciales:

A. Etapa 1: Conformar un grupo de análisis de riesgo

Se trata de conformar un equipo de trabajo, responsable de llevar adelante el modelado; entre sus actividades principales se encuentran:

Identificar los objetivos de seguridad: Determinar los objetivos ayudará a clarificar y cuantificar el esfuerzo que se debe dedicar en las actividades subsiguientes.

Crear una descripción general de la aplicación, con la cual se trata de identificar los actores y características más importantes que interactúan con la aplicación, esto con el fin de lograr reconocer cuales podrían llegar a ser las amenazas más significativas.

B. Etapa 2: Descomponer la aplicación e identificar componentes claves

Se trata de conocer la arquitectura de la aplicación y descomponerla, con el objetivo de identificar las principales funcionalidades y los módulos susceptibles a provocar un mayor impacto en la seguridad (Vulnerabilidades conocidas).

Lo realmente importante en esta etapa es el DFD (diagrama de flujo de datos), como se señala coloquialmente "una imagen vale más que mil palabras" y en un diagrama se puede obtener suficiente información fundamental.

Un DFD no es un diagrama de flujo normal, es un diagrama que muestra los flujos de datos entre los componentes de la aplicación; para entender de manera más clara las diferencias, se establece un ejemplo que contiene un diseño erróneo de una DFD

para el cajero automático de un banco. Si se observa, éste representa un diagrama de flujo común que reproduce todo el proceso.

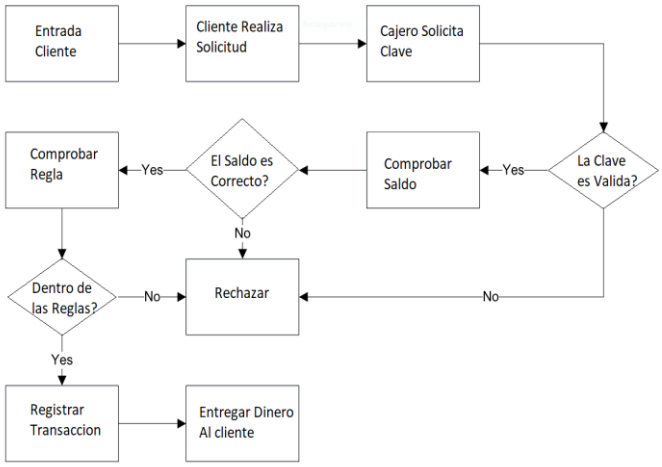


Fig. 2. Diagrama de flujo común, que describe el proceso de un cajero automático para un banco. Fuente: TORR, Peter. Guerrilla Threat Modelling.

Mientras un DFD, comúnmente comienza por un diagrama de contexto acompañado de todos sus componentes y todas las entidades externas con las que se comunica la aplicación; un ejemplo para el cajero automático del banco podría ser el siguiente.

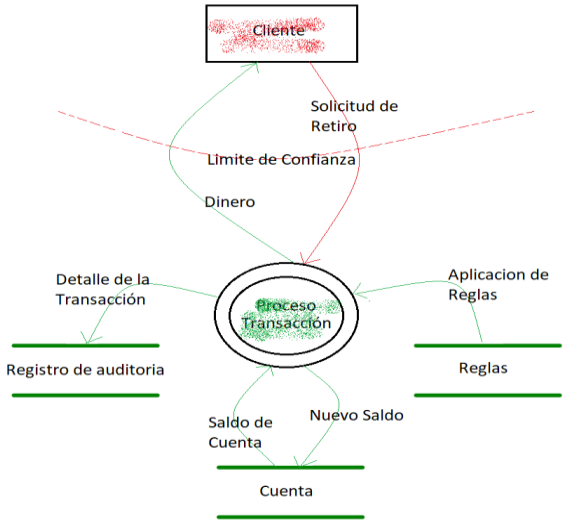


Fig. 3. Diagrama de flujo de datos (DFD) sin descomponer, que describe el proceso de un cajero automático para un banco. Fuente: TORR, Peter. Guerrilla Threat Modelling.

Lo ideal está en descomponer el diagrama en uno o más de N niveles utilizando un mayor detalle, esta descomposición puede ser tan básica o compleja como se desee, aunque siempre se recomienda utilizar algo simple (un diseño con mucho detalle puede llegar a ser muy difícil de interpretar).

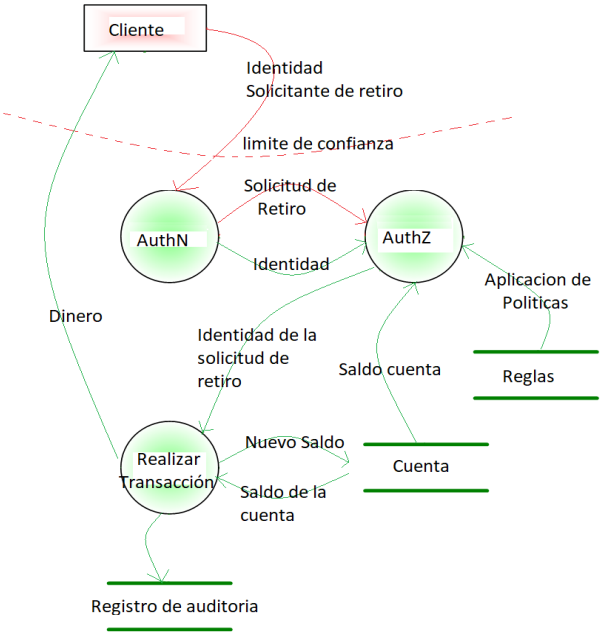


Fig. 4. Diagrama de flujo de datos (DFD) descompuesto, que describe el proceso de un cajero automático para un banco. Fuente: TORR, Peter. Guerrilla Threat Modelling.

En el diseño anterior se puede identificar AuthN, el cual es una manera corta de señalar la autenticación de usuario y AuthZ, que indica autorización de usuario.

Dentro de las sintaxis establecidas se muestran:

- **Los rectángulos:** Representan las entidades externas que aplican al proceso, no se estudian en el modelado de amenazas (Están etiquetados por nombres).
- **Los círculos:** Representan los sub-componentes del componente principal de procesamiento de datos, contienen tanto datos de entrada como salida (Están etiquetados por verbos).
- **Doble circulo:** Representan componentes que se van a convertir en diagramas más detallados.
- **Doble líneas:** Representan almacenes de datos (Físicos o lógicos) a los cuales se accede desde los componentes del modelado de amenazas, estas líneas pueden tener flujos de datos que entran, salen o ambos (Están etiquetados por nombres).
- **Flechas:** Representan los flujos de datos que entran y salen de los componentes y entidades externas (esta flecha tiene un único sentido no se pueden utilizar doble flecha) (están etiquetados por nombres).

- **Líneas con puntos:** Representan los límites de confianza, los límites de la máquina, los límites de procesos entre otros (están etiquetados por nombres).

Los círculos deben tener datos que entran y datos que salen ya que su propósito es procesar una entrada y producir una salida; si en el diseño se tiene círculos que únicamente aceptan entradas o repartan datos, lo mejor es modelar éste como un almacén de datos o como componente externo.

Los componentes dentro de DFD también se pueden clasificar por colores:

- **Rojo:** Componentes que no son de confianza. Únicamente los rectángulos, líneas o líneas dobles pueden utilizar este color, los círculos no pueden ser de este color.
- **Verde:** Componentes de confianza, cualquier tipo de objeto dentro del diagrama puede tener este color.
- **Azul:** Componentes que están siendo modelados, únicamente los círculos pueden ser azules.
- **Naranja:** Componente con confianza parcial, aplica únicamente para rectángulos y líneas.

C. Etapa 3: Determinar las amenazas de cada componente de la aplicación

En esta etapa se trata de adoptar el punto de vista del atacante, basándose no solo en las características propias de un componente de la aplicación, sino también en la interacción con otros componentes. Para esta etapa (de acuerdo con el modelo propuesto por Microsoft), se encuentra el método de clasificación STRIDE.

Método de clasificación STRIDE.

Este término es el acrónimo de “Spoofing identity, Tampering with data, Repudiation, Information disclosure, Denial of service, Elevation of privilege".

Para seguir el método STRIDE, se descompone el sistema en diferentes componentes, seguido de analizar cada uno, con el objetivo de comprobar si es susceptible de sufrir amenazas; después se proponen acciones que traten de mitigarlas y se repite el proceso hasta llegar a una situación cómoda con las amenazas restantes.

- **Spoofing Identity (Suplantación de identidad):**

Esta categoría de amenazas indica básicamente que los usuarios no deberían ser capaces de hacerse pasar por otros usuarios. Es importante enfocarse en un adecuado sistema de autenticación y poner mucha atención en los temas de uso de sesión con el fin de evitar robos por medio del uso de éstas.

- **Tampering with Data (Manipulación de datos):**

Por la llegada de metodologías de desarrollo de software ágiles, es habitual que con el fin de cumplir con los tiempos se sacrifique la implementación de las medidas de seguridad.

Una de las primeras medidas preventivas que se suelen sacrificar, es el filtrado y la validación de los datos enviados y recibidos de los usuarios de la aplicación.

En el caso específico de las aplicaciones desarrolladas para un entorno web, se puede dar a manera de ejemplo, las vulnerabilidades conocidas, tales como la inyección de sentencias SQL o el XSS; en general, en aquellos casos en los que las aplicaciones proporcionan al usuario, datos que no son obtenidos directamente de la propia aplicación, se realiza algún tipo de cálculo con ellos y posteriormente se almacenan, teniendo en cuenta que estos datos pueden ser susceptibles de una manipulación efectuada por un usuario malintencionado que disponga de las herramientas adecuadas.

- **Repudiation (Repudio):**

En esta categoría, se trata de establecer un nivel adecuado del seguimiento de las acciones realizadas por los usuarios de la aplicación; con el fin de evitar que aparezcan situaciones no deseadas se debe intentar garantizar él no repudio de los usuarios.

Cada aplicación sobre la base de su funcionalidad, características y entorno, presenta diferentes riesgos y por lo tanto, las medidas de registro de las

actividades que son efectuadas sobre los usuarios serán también diferentes.

Se debe resaltar, que no serán las mismas medidas de seguimiento y registro en una aplicación que efectúa operaciones de compra y venta, que en otra que maneje un simple gestor de contenidos.

Por ejemplo, una aplicación que permite a sus usuarios efectuar operaciones de compra y venta de acciones en bolsa, deberá ser muy estricto en el manejo y almacenamiento de los registros, aunado a ello, no se puede dar el lujo de perder el rastro de ninguna operación realizada; de no ser así, y de producirse un error durante el transcurso de las operaciones, cualquiera de las partes involucradas podrían salir perjudicada.

Si se imagina que se produce un error en una transacción (por ejemplo, en la venta de las acciones del usuario x) y está fue realmente causada por error del mismo usuario en el momento de efectuar una acción de manera errónea, aquel posiblemente trataría de imputar el inconveniente, indicando que fue originado por un error en el sistema; si la entidad responsable de la aplicación le es imposible verificar de cuál de las dos situaciones se trata, es poco probable pero posible que deba terminar asumiendo la pérdida económica.

- **Information Disclosure (Revelación de información):**

La existencia de vulnerabilidades en una aplicación que permitan extraer información sensible, es un factor claro de riesgo que en ocasiones puede derivar en pérdidas económicas, así como también, en la disminución de la confianza y reputación de cara a posibles o ya existentes usuarios, clientes, proveedores o inversores.

Por ejemplo, el uso inseguro de memoria USB compartida que facilite la obtención de credenciales, obtención de un listado de usuarios o correos electrónicos válidos (envío de spam), obtención de información que facilite la identificación del entorno (sistemas, aplicaciones) y en general, cualquier información que pueda facilitarle la tarea a un atacante proporcionando detalles del funcionamiento de la propia aplicación, sistema o del usuario que favorezcan a su explotación.

- **Denial of Service (Denegación de servicio):**

A la hora de diseñar una aplicación, o en el momento de añadir una nueva funcionalidad, es conveniente evitar aquellas situaciones que puedan devengar en la consecución de un ataque de denegación de servicio.

Por esta razón es conveniente proporcionar un uso racional de los recursos con los que interactúa la aplicación, ya sea evitando cálculos complejos, búsquedas intensivas o el acceso a inserción de archivos de gran tamaño a usuarios no permitidos.

- **Elevation of Privilege (Elevación de privilegios):**

En el caso de que la aplicación o el sistema proporcionen diferentes niveles de privilegio en función de los distintos tipos de usuarios, todas las acciones que lleven al uso de privilegios deben ser filtradas a través de un mecanismo adecuado de autorización. Este método de validación de los privilegios deberá ser suficientemente robusto para impedir un posible escalamiento de privilegios.

Para almacenar los datos que se van recolectando, se puede utilizar como ejemplo la siguiente tabla:

TABLA III
EJEMPLO CLASIFICACIÓN STRIDE

Categoría	Manipulación de datos.
Descripción	Inyección de comandos SQL.
Objetivo de la amenaza	Componente de acceso a base de datos.
Nivel de riesgo	¿?????
Técnicas de ataque	El atacante introduce comandos SQL en el campo usuario utilizado para formar una sentencia SQL.
Contramedidas	Filtrar el contenido del campo usuario, utilizar un procedimiento almacenado que utilice parámetros para acceder a la base de datos.

La anterior tabla referencia un ejemplo del método de clasificación STRIDE. Fuente: P.F, Daniel. Análisis y Modelado de Amenazas.

Lo ideal es aplicar cada una de las categorías del STRIDE para cada componente del DFD de manera repetitiva, en búsqueda de posibles amenazas.

D. Etapa 4: Asignar valor a cada amenaza

Se trata de calcular el riesgo asociado a cada amenaza, esto nos permite identificar qué respuesta dar para cada amenaza y priorizarlas según el riesgo que representa para la aplicación. Para esta etapa (de acuerdo con el modelo propuesto por Microsoft) se encuentra el método de puntuación DREAD.

Método de puntuación DREAD.

Una vez que se tiene identificada la lista de amenazas, el siguiente paso consiste en puntuarlas de acuerdo con el riesgo que suponen. Esto permite priorizar las actuaciones a efectuar para mitigar el riesgo. Comúnmente un riesgo se puede cuantificar, como el resultado de multiplicar la probabilidad de que la amenaza se produzca por el impacto.

Riesgo = Probabilidad * impacto.

Se emplea una escala del 1 al 10 para valorar la probabilidad de ocurrencia, donde el 1 representaría una amenaza que es poco probable que ocurra, y 10 sería una amenaza muy probable. De igual forma, con el impacto, 1 indicaría un impacto mínimo y 10 un impacto máximo.

Este enfoque tan simplista, nos permite en un primer momento clasificar las amenazas en una escala entre 1-10 que podemos dividir en tres partes según su riesgo: Alto, Medio, Bajo.

Ej. Probabilidad 10, impacto 7, el riesgo será:
Riesgo = 10 * 7= 70

Una amenaza con un riesgo alto, debería ser tratada de forma rápida, una amenaza con riesgo medio, aunque importante es de menor urgencia, y por último la de riesgo bajo se podría llegar a ignorar dependiendo del costo y del esfuerzo que se requiera.

El problema de aplicar este método, está dada por la dificultad de valorar de igual forma el riesgo cuando esta valoración es efectuada por diferentes personas. El método DREAD, trata de facilitar el uso de un criterio común respondiendo a las siguientes preguntas:

- **Damage potential (Daño potencial):** ¿Cuál es el daño que puede originar la vulnerabilidad si llega a ser explotada?
- **Reproducibility (Reproducibilidad):** ¿Es fácil reproducir las condiciones que propicien el ataque?
- **Exploitability (Explotabilidad):** ¿Es sencillo llevar a cabo el ataque?
- **Affected users (Usuarios afectados):** ¿Cuantos usuarios se verían afectados?
- **Discoverability (Descubrimiento**): ¿Es fácil encontrar la vulnerabilidad?

De modo que, por ejemplo, se podría establecer un sistema de puntuación de la siguiente manera, de 12-15 sería considerado un riesgo alto, entre 8-11 medio, y entre 5-7 bajo. A continuación, se muestra lo que podría ser una tabla típica de puntuación para priorizar las amenazas:

TABLA IV
TABLA DE PUNTUACIÓN DREAD.

Puntuación	Alto (3)	Medio (2)	Bajo (1)
Damage potential (Daño potencial)	El atacante podría Ejecutar aplicaciones con permiso de administrador.	Divulgación de información sensible.	Divulgación de información trivial.
Reproducibility (Reproducibilidad)	El ataque es fácilmente reproducible.	El ataque se podría reproducir, pero únicamente en condiciones muy concretas.	Ataque difícil de reproducir.
Exploitability (Explotabilidad)	Un programador novato podría implementar el ataque en poco tiempo.	Un programador experimentado podría implementar el ataque.	Se requieren cierta habilidad y conocimiento.
Affected users (Usuarios afectados)	Todos los usuarios.	Algunos Usuarios.	Pocos usuarios afectados.
Discoverability (Descubrimiento)	Existe información pública que explica el ataque.	Vulnerabilidad afecta a una parte de la aplicación que casi no se utiliza.	El fallo no es trivial, no es muy probable utilizarlo para causar un daño.

La anterior tabla referencia el método de puntuación DREAD suministrado por Microsoft. Fuente: P.F, Daniel. Análisis y Modelado de Amenazas.

Si se toma como ejemplo, el caso mencionado anteriormente donde existía la posibilidad de inyectar comandos SQL en la aplicación, de acuerdo con la clasificación DREAD se obtendría.

TABLA V
EJEMPLO PUNTUACIÓN DREAD

Amenaza	D		R	E	A	D	Total	puntuación
Inyección de comandos SQL	3		3	3	2	3	14	Alto

La anterior tabla referencia ejemplo del método de puntuación DREAD. Fuente: P.F, Daniel. Análisis y Modelado de Amenazas.

Ahora que ya se conoce cuál es el riesgo, es posible actualizar la información referente a esta amenaza que se obtuvo durante la clasificación construida con el método STRIDE.

TABLA VI
RESULTADO NIVEL DE RIESGO.

Categoría	Manipulación de datos.
Descripción	Inyección de comandos SQL.
Objetivo de la Amenaza	Componente de acceso a base de datos.
Nivel de riesgo	Alto.
Técnicas de ataque	El atacante introduce comandos SQL en el campo usuario utilizado para formar una sentencia SQL.
Contramedidas	Filtrar el contenido del campo usuario, utilizar un procedimiento almacenado que utilice parámetros para acceder a la base de datos.

La anterior tabla referencia el nivel de riesgo, resultado del modelo de puntuación DREAD y clasificación STRIDE. Fuente: P.F, Daniel. Análisis y Modelado de Amenazas.

E. Etapa 5: Definir cómo responder a las amenazas

Frente a las amenazas se pueden dar diferentes respuestas, basándose en el valor del riesgo asociado a cada una, por esta razón, finalmente después de haber puntuado el riesgo de la aplicación, se tendrá una lista priorizada de amenazas para mitigar; como regla general se deberá trabajar en aquellas amenazas que supongan un mayor riesgo primero. Mitigar los riesgos de menor importancia no ayudará a reducir el riesgo global, incluso si son fáciles o baratos de arreglar. Es importante siempre tener en cuenta el costo de los controles, con el objetivo de validar si finalmente la inversión es justificable, por ejemplo, si la implementación de un control tiene un valor de $ 1’000.000 y su objetivo es contener un fraude de $20.000, llevaría unos 50 años recuperar la inversión para compensar la pérdida.

XI. HERRAMIENTAS PARA MODELADO DE AMENAZA

En la actualidad existen algunas herramientas de apoyo (proporcionadas por las principales entidades) que ayudarán a construir un modelado de amenaza de una manera sencilla. Dentro de las principales se encuentran:

- SDL threat modeling tool (SDL).
- Threat Analysis and Modeling tool (TAM).
- Consultative Objective Risk Analysis System (CORAS).

Estas herramientas proporcionan bibliotecas de ataques predefinidos y recomiendan cuales deberían ser las mitigaciones eficaces para los diferentes tipos de ataque asociado a cada amenaza.

A. Escenario considerado.

Con el objetivo de mostrar la funcionalidad y el enfoque que proporciona estas herramientas en el análisis y diseño del modelado de amenazas, se mostrará una aplicación web típica exponiendo una parte de ésta, para mostrar como ejemplo.

El escenario a mostrar consiste en un sistema proveedor de servicios de distribución de contenidos académicos o biblioteca virtual, el cual suministra información como textos, artículos de revistas científicas, investigaciones, entre otros. Esta información se encuentra en formato digital con la opción de impresión.

El sistema está basado en una arquitectura de 3 capas:

- Navegador web.
- La aplicación está expuesta en internet.
- Base de datos (Cuentas de clientes y publicaciones).

En el diagrama de componentes se pueden identificar 2 servidores:

- Servidor de bases de datos.
- Servidor web.

Para el análisis, se presentan las siguientes fases:

- **Fase de requerimientos:** Identificando requerimientos funcionales principales y requerimientos de seguridad.
- **Fase de diseño y modelado de amenazas:** Mostrado específicamente a la sección de autenticación de usuario.

Se presenta el resultado utilizando la aplicación SDL, como ayuda para documentar los requerimientos de la aplicación y su arquitectura.

B. Implementación del escenario considerado, aplicando herramienta SDL.

La herramienta SDL funciona mediante el uso de representaciones gráficas, la cual agiliza la construcción de modelado de amenazas.

La herramienta da soporte con adaptaciones construidas de tal forma, que los componentes se representan como burbujas de procesos, los roles de usuarios dependencias externas como entidades, y los datos de forma directa.

Con el fin de determinar cuáles son las principales amenazas, es necesario conocer:

- Cuáles son los puntos de entrada.
- Los niveles de confianza.
- Los activos de mayor interés.

Para el caso mencionado anteriormente se obtiene el siguiente diagrama:

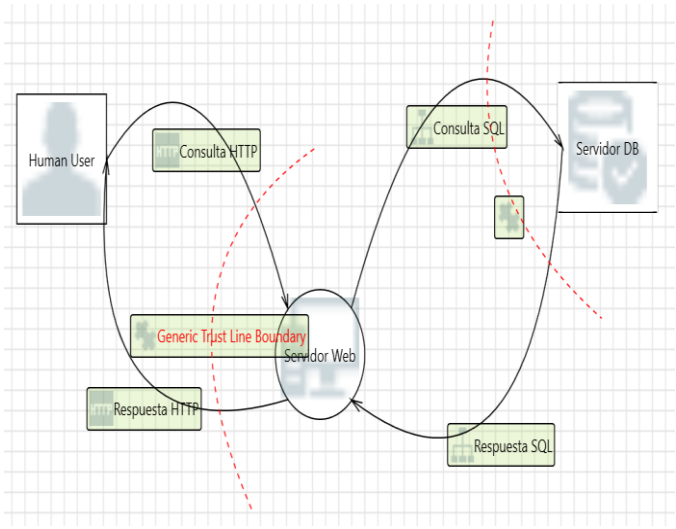


Fig. 5. Diagrama del escenario considerado, construido en la herramienta Threat modeling tool. Fuente: El autor.

A partir de la construcción del diagrama, la herramienta genera de manera automática un análisis por cada componente de la lista definida en el diseño, con fundamento en el método STRIDE y con base en la siguiente matriz:

TABLA VII
MATRIZ STRIDE THREAT MODELING TOOL

Elemento	Entidad Externa	Proceso	Almacén de datos	Flujo de datos
Tampering with data		x	x	x
Repudiation	x	x	x	
Information disclosure		x	x	x
Denial of service		x	x	x
Elevation of privilege		x		

La anterior tabla referencia la matriz utilizada por la herramienta threat modeling tool, para generar el análisis de forma automática. Fuente: RAMOZ, Juan. Aplicar el modelo de amenazas para incluir la seguridad en el modelado de sistemas.

De acuerdo con el diagrama anterior se obtiene el siguiente resultado:

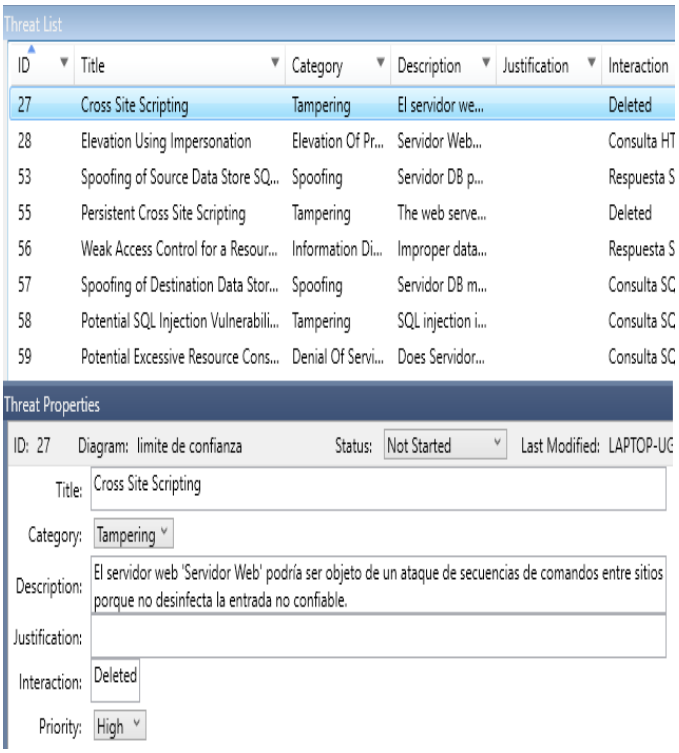
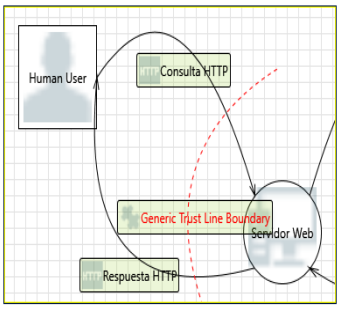


Fig. 6. Resultado análisis de riesgo generado automática por la herramienta threat modeling tool. Fuente: El autor.

También es posible generar el reporte en formato HTML, el cual se muestra de la siguiente manera:

Interaction: Consulta HTTP



1. Elevation by Changing the Execution Flow in Servidor Web **[State: Not Started] [Priority: High]**

Category: Elevation Of Privilege
Description: An attacker may pass data into Servidor Web in order to change the flow of program execution
Justification: <no mitigation provided>

Fig. 7. Informe en formato HTML, generado desde la herramienta threat modeling tool. Fuente: El autor.

CONCLUSIONES

1) El modelado de amenaza es un proceso abierto, lo cual se traduce en que cada persona puede llegar a tener diferentes interpretaciones del mismo, sin embargo, el seguimiento de una metodología y un enfoque de mejoramiento se presenta como un factor clave de cara a mejorar la seguridad de una aplicación o un sistema.

2) La técnica permite la reutilización de componentes, lo cual es un factor importarte debido a que puede servir posteriormente en otros proyectos que tengan cierta similitud, particularmente en el caso de las aplicaciones web, teniendo en cuenta que en muchas ocasiones, la propia naturaleza de estas, hace que se encuentren funcionalidades similares en aplicaciones diferentes.

3) El éxito de implementación de un modelado de amenazas, está en aplicarlo en múltiples ocasiones durante todo el desarrollo y evolución del proyecto.

4) Las repetidas revisiones y mejoras del modelado de amenazas, se deben ir incorporando a medida que se produzcan cambios en el entorno de la aplicación o sistema, diseño, implementación, configuración, se modifiquen o aparezcan nuevos casos de uso.

5) La incorporación de la técnica de comprobación o medidas de seguridad durante el ciclo de vida de desarrollo de software, puede no parecer rentable a corto plazo; sin embargo, la historia termina demostrando que utilizar malas prácticas que se logren materializar en amenazas, traen como consecuencia, que el impacto y perdidas económicas para las organizaciones superen el costo de implementación de sus controles.

REFERENCIAS

[1] TORR, Peter. Guerrilla Threat Modelling [En línea]. Febrero, 2005. Disponible en Internet: https://blogs.msdn.microsoft.com/ptorr/2005/02/22/guerrilla-threat-modelling-or-threat-modeling-if-youre-american.

[2] RETANA, Pilar. Análisis y Modelado de Amenazas [En línea]. Septiembre, 2013. Disponible en Internet: https://prezi.com/uky4fywbiexu/analisis-y-modelado-de-amenazas/

[3] OWASP, Foundation, Guía de Prueba OWASP. 2008. 372p.

[4] P.F, Daniel. Análisis y Modelado de amenazas. España. 2006. 42p.

[5] CASTELLARO, Martha y RAMON, Juan. Aplicar el Modelo de Amenazas para incluir la seguridad en el Modelado de Sistemas. Argentina. 2016. 16p.

Autor
Gabriel Eduardo Barba Olivares.
Ingeniero de Telecomunicación de la Universidad Santo Tomás de Aquino, desde el 2008 ha desempeñado labores en diferentes ramas del sector de TI, como es el desarrollo de software, infraestructura y seguridad informática.