UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 3

MODELO DE SEGURIDAD, BASADO EN LA METODOLOGÍA RUP, PARA EL PROCESO DE DESARROLLO DE SOFTWARE SEGURO EN LA UCI.

Propuesta de aplicación en el módulo Registro Mercantil del sistema SAREN.

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO DE
INGENIERO INFORMÁTICO

Autor(s):

Dusniel Horta Centeno.

Rayner William Ulloa Ramos.

Tutor:

Ing. Ernesto Medina Delgado.

Caracas.

Mayo de 2007.

DECLARACIÓN DE AUTORÍA.

Declaramos que somos los únicos autores de este trabajo y autorizamos a la dirección de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los _____ días del mes de _____ del año _____

_____

Dusniel Horta Centeno.

Autor.

_____

Rayner William Ulloa Ramos.

Autor.

_____

Ing, Ernesto Medina Delgado.

Tutor.

AGRADECIMIENTOS

A la Universidad de las Ciencias Informáticas, por darnos la oportunidad de formarnos en ella durante estos últimos cinco años y hacer de nuestros sueños una realidad.

A nuestro tutor, el Ing. Ernesto Medina Delgado, por haber compartido sus conocimientos y experiencia durante el desarrollo de este trabajo, a pesar de su juventud. Sabes que eres amigo y hermano también.

A los profesores que han compartido sus experiencias durante estos años. Siempre estaremos en deuda con ustedes.

A Yane por estar siempre dispuesta ayudar en lo más mínimo.

A nuestros padres y hermanos por haber confiado siempre en nosotros. Esto es solo el comienzo.

A todos los que en diferentes momentos, nos han brindado su ayuda, durante la realización del trabajo y otros momentos difíciles: amigos de R&N, compañeros de aula, en fin gracias a todos.

Con mucho cariño y admiración, gracias a todos.

Autores.

DEDICATORIA

A ti mami, por hacer de mi lo que soy. No hay una sola línea de este trabajo donde tú no estés presente. Nunca me alcanzará el tiempo para hacer por ti lo que haz hecho por mi. Siempre haz confiado ciegamente en todo lo que he hecho y hacer que te sientas orgullosa ha sido el camino que he tomado desde que tuve conciencia de ello. Eres la mejor madre del mundo y sabes que nunca te fallaré. Vive muchos años más que te necesito.

A mi hemanita y también mamá yadi, por estar siempre tan cerca de mi. Más que hermana siempre te he considerado una segunda madre. En ti valoro mucho la forma de desenvolverte en la vida. Creo que los golpes que recibiste cuando niño por mi culpa te hicieron quererme más, y creo que eso también hizo que yo te quiera como lo hago hoy (jejejeje). Te agradezco mucho lo que me haz ayudado y junto a la vieja son las dos personas que más amo en la vida.

A yane por soportar mis majaderías. Princesa, sabes que en esto hay mucho de ti también, no te pongas celosa porque revelé arriba los dos amores de mi vida, tu estás al mismo nivel pero sabes que te quiero de otra forma. Gracias por soportar pacientemente algunos de mis momentos de molestias cuando algo no me salía como deseaba, o cuando era muy tarde en la noche y querías que no siguiera programando (jejejeje). Sabes que te aprecio mucho como mujer, por ser tan comprensiva como eres. Te quero mucho princesa.

A mis otros hemanos. Bueno ustedes tampoco se pongan celosos por las cosas que le dije a yadi. Saben que los quiero de igual manera y en mi tienen un apoyo que tampoco les fallará. Cualquier cosa saben que pueden contar conmigo. Frank, aunque eres tímido sabes que te quiero con el alma. Yunior, a pesar de que vivíamos peleando cuando niños, sabes que tienes en mí tu mejor amigo. Nandi, a pesar de que tienes el carácter fuerte, se que nos adoras a todos. Kenia, bueno aunque hemos vivido juntos muy poco tiempo quiero decirte que te quiero igual y que también aprecio mucho la forma en que luchas las cosas.

Dusniel.

A mis padres, demás familiares y amigos por apoyarme siempre en todo y en especial a mi mamá que siempre ha estado presente en todo momento de preocupación, porque sé que tanto para mí como para ella éste es uno de sus mayores sueños. Los quiero mucho y siempre confíen en mí, que haré todo a mi alcance para no fallarles. También agradecerle a la revolución por darme la oportunidad de estudiar y graduarme en la UCI y en especial a nuestro Comandante en Jefe, Fidel Castro Ruz por hacer realidad nuestros sueños.

Rayner.

RESUMEN

El desarrollo de software en la Universidad de Ciencias Informáticas (UCI) ha estado creciendo continuamente, desde la fundación misma del centro. Los clientes exigen en sus contratos el desarrollo y liberación de productos que garanticen no solo capacidad funcional sino la confidencialidad, integridad, autenticidad y confiabilidad de los procesos de negocios una vez llevados a un entorno automatizado. Para lograr esto se requiere gestionar y planificar la seguridad de forma paralela al proceso de Ingeniería de Software, constituyendo este uno de los retos más urgentes a que se enfrenta la industria actualmente. Para guiar y controlar las actividades de desarrollo de estos sistemas la Universidad ha adoptado al Proceso Unificado de Desarrollo de Software (RUP)[1], en la generalidad de los proyectos productivos, como proceso a seguir.

El objetivo fundamental de este trabajo es desarrollar un modelo de seguridad que se integre al proceso de desarrollo de software en la UCI, el cual, aplicado paralelamente a la Ingeniería de Software, garantice la gestión efectiva y continua de la seguridad de los activos lógicos críticos en cada uno de los flujos de trabajo propuestos por el Proceso Unificado de Desarrollo de Software, desde el inicio hasta las pruebas del sistema.

El modelo se divide en 4 flujos de trabajo fundamentales: Inicio y captura de requisitos de seguridad, Análisis y diseño seguro, Implementación y codificación segura, Pruebas de seguridad.

Para cada flujo de trabajo se especifican qué artefactos y actividades, en materia de seguridad, se deben elaborar y seguir, respectivamente, así como los pasos necesarios en la generación de los mismos de una manera continua y controlada. Se especifican, además, los trabajadores propuestos por RUP y los artefactos que estos generan y que constituyen entradas de información imprescindibles a los especialistas de seguridad durante el proceso de gestión. La aplicación de este modelo, logrará la elaboración de productos de software más confiables que cumplan con los requerimientos de seguridad exigidos.

El modelo tiene una aplicación práctica inmediata en la planificación de la seguridad del módulo Registro Mercantil perteneciente al Sistema Automatizado de Registros y Notarías (SAREN).

[1] Rational Software Corporation es una empresa que posee más de tres décadas de experiencia en el desarrollo de modelos que guíen el desarrollo de software. Pero no es hasta 1995 cuando compró la empresa Objectory AB y concreta la tarea de unificar principios básicos en los procesos de desarrollo existentes.

ÍNDICE

INTRODUCCIÓN.

La integración de la seguridad, como parte indispensable del proceso de desarrollo de software, es un tema que hasta hace muy poco tiempo no preocupaba a los responsables y desarrolladores de sistemas para el mercado. Es frecuente ver noticias relacionadas con sistemas informáticos donde aparecen serios problemas de seguridad detectados, que han conllevado al colapso de los procesos manejados de forma electrónica causando grandes pérdidas a los consumidores y reconocidas compañías que hacen uso de los mismos. En la actualidad, el aumento creciente de la informática y tecnologías de la información en los procesos de negocios de la mayoría de las empresas del mundo, ha provocado un incremento en las exigencias de los clientes por adquirir productos de software más seguros que garanticen la confidencialidad, autenticidad, disponibilidad y confiabilidad de la información que se manipula. Es un hecho cierto que aquellas compañías que se caracterizan por desarrollar sistemas con altos índices de seguridad pues han tenido mayores éxitos, se han ganado la confianza de los clientes y se han establecido como vanguardias en la industria de la informática (Lipner and Howard 2005). Esto se traduce en un incremento del interés y la preocupación de las compañías de desarrollar productos y servicios informáticos de altas prestaciones que garanticen la seguridad tanto del código como de los activos[2] manejados. El surgimiento de metodologías dedicadas a la modelación de riesgos en sistemas informáticos constituye hoy una necesidad creciente para los proveedores de software en el mundo. La incorporación de estas marca un futuro más confiable en el desarrollo de sistemas de altas prestaciones.

La Universidad de las Ciencias Informáticas, se encuentra en estos momentos inmersa en el desarrollo de una gama de sistemas de software muy importantes a partir de convenios que han sido firmados con diferentes empresas, tanto del ámbito nacional como internacional. Muchos de estos productos que se implementan, manejarán un volumen considerable de información de clientes y de manera general dentro de los procesos de negocios propios de la empresa en que serán instalados finalmente. Para guiar y controlar las actividades de desarrollo de estos sistemas la Universidad ha

[2] Recurso tangible o intangible que tiene un valor importante para el correcto funcionamiento de una empresa o sistema informático, el cual en caso de resultar afectado por alguna razón provoca un impacto sobre otros activos.

adoptado al Proceso Unificado de Desarrollo de Software, en la generalidad de los proyectos productivos, como proceso a seguir.

Los clientes exigen en sus contratos el desarrollo y liberación de un producto de software que garantice la confidencialidad, integridad, autenticidad y confiabilidad de la información sensible almacenada en formato electrónico. Sin embargo, la **situación problémica** que se presenta en los proyectos productivos de la UCI en cuanto al análisis de riesgos y amenazas que afectan al software que se elabora, está dada en que actualmente la Universidad no tiene diseñado y establecido un modelo de gestión de la seguridad adecuado que guíe y especifique por cada uno de los flujos de trabajo propuestos por RUP, qué artefactos, diagramas y actividades, en materia de seguridad, se deben elaborar y seguir para realizar un modelado de amenazas y análisis de riesgos efectivos, así como los pasos necesarios en la generación de los mismos de una manera continua y controlada desde el inicio hasta las pruebas del sistema. Esta situación conlleva a que los proyectos productivos en su mayoría no adopten un guía común que les señale la forma correcta de identificar y modelar amenazas partiendo de las características específicas de cada software desarrollado. Esto se traduce en que los ingenieros y personal ligados al propio desarrollo vean generalmente el tema de la seguridad como una tecnología a introducir y no como parte del proceso y deciden frecuentemente iniciar el desarrollo sin la conformación de un Equipo de Seguridad (ES) que adopte un modelo coherente de planificación y gestión de la misma.

A partir de esta situación problémica surge el siguiente **problema científico:** ¿Cómo integrar la seguridad de forma efectiva al Proceso de desarrollo de software en la UCI mediante el diseño y aplicación de un modelo que contemple los principales artefactos de seguridad que deben generarse en cada flujo de trabajo en el cual se defina el formato para representarlos, documentarlos y gestionarlos así como los especialistas, destinados a ello, y las actividades en que estos deben participar desde el inicio hasta las pruebas?

El **objeto de estudio** en que se basa este trabajo de diploma es el proceso de desarrollo de software de la Universidad de las Ciencias Informáticas, cuyo **campo de acción** se centra fundamentalmente en la gestión de la seguridad de los activos lógicos críticos. A partir del campo de acción anteriormente descrito se ha definido como **objetivo general de esta tesis**, desarrollar un modelo para gestionar de forma efectiva y continua la seguridad de los activos lógicos críticos, en cada uno de los flujos de trabajo propuestos por RUP para el desarrollo de software, desde el inicio hasta las pruebas del sistema.

A partir del objetivo general se plantean los siguientes **objetivos específicos**:

1. Garantizar que el modelo desarrollado sea lo suficientemente aceptable y comprensible para que el Equipo de Seguridad designado en cada proyecto productivo sea capaz de definir y realizar la gestión de la seguridad de forma controlada y continua desde el inicio hasta las pruebas.
2. Garantizar los principales artefactos de seguridad que se generan en cada flujo del proceso de desarrollo, así como las actividades necesarias para generarlos y los especialistas que en ellas participan.
3. Elaborar una propuesta, basada en el modelo desarrollado, dirigida a la gestión de la seguridad en el Módulo Registro Mercantil del software SAREN desde la planificación inicial hasta el diseño de las pruebas de seguridad del sistema.

La **hipótesis** que se plantea este trabajo es que si se elabora e integra al proceso de desarrollo de software en la UCI un modelo para el análisis y gestión de la seguridad, de manera controlada y continua desde el inicio hasta las pruebas del sistema, entonces, se logrará la elaboración de productos de software más confiables que cumplan con los requerimientos de seguridad exigidos.

Para dar cumplimiento a los objetivos general y específicos planteados se han definido una serie de **tareas de investigación concretas**:

1. Realizar un estudio previo del estado del arte sobre los principales temas relacionados con la Seguridad Informática en el desarrollo de software y los problemas que afectan el buen desarrollo de esta.
2. Investigar acerca de metodologías de evaluación de riesgos y sus herramientas de soporte actuales así como de estudios y trabajos anteriores que aporten elementos al desarrollo del modelo.
3. Estudiar el Proceso Unificado de Desarrollo de Software, en el cual se apoyará la construcción del modelo, para definir los flujos de trabajo importantes para la construcción del modelo.
4. Identificar los artefactos de seguridad que deben incluirse en el modelo así como las actividades de gestión, los artefactos de RUP y sus trabajadores que en conjunto con el Equipo de Seguridad realizan el proceso de gestión de la seguridad a través de los flujos de trabajo definidos.
5. Elaborar el formato que debe regir la representación de cada artefacto definido.

6. Entrevistar a los especialistas del Módulo Registro Mercantil y revisar la documentación del sistema SAREN para dar cumplimiento al cuarto objetivo específico planteado.

El Modelo de seguridad desarrollado en el trabajo de diploma pudiera ser de un alto **valor práctico** que se reflejaría en la elaboración de planes de Seguridad Informática coherentes en los proyectos productivos de la UCI. Además de que constituiría un **aporte** práctico relevante por el hecho de que por primera vez los proyectos productivos de la UCI pudieran integrar al Proceso de desarrollo de software un modelo que les apoye y guíe en la planificación de la seguridad durante todo el proceso de desarrollo.

El Trabajo de diploma está estructurado en tres capítulos organizados de la siguiente manera:

- **Capítulo 1. Fundamentación teórica del tema**: Se abordan temas relacionados con diferentes enfoques y conceptos en cuanto a la seguridad de software y elementos importantes que miden la confiabilidad de un sistema. Además se analizan aspectos relacionados con la problemática actual de la seguridad y los objetivos fundamentales para obtener un software confiable. Se hace un estudio del estado del arte sobre el surgimiento y evolución de metodologías y modelos elaborados por proveedores de software destinados al análisis y evaluación de riesgos. También se justifica la selección de RUP como marco de apoyo en cuyos flujos se basará la elaboración del modelo.
- **Capítulo 2. Desarrollo del modelo propuesto:** Se construye el modelo propuesto que permite integrar la gestión de la seguridad al Proceso de desarrollo de software en los proyectos productivos de la UCI, desde el inicio hasta las pruebas, apoyándose en los flujos de trabajo fundamentales definidos por RUP quien ha sido establecido como marco de trabajo a seguir en la generalidad de los proyectos de la Universidad, de manera tal que el proceso de gestión de la seguridad sea paralelo al desarrollo.
- **Capítulo 3. Propuesta de Aplicación del modelo en la planificación de la seguridad del Módulo Registro Mercantil:** A partir del Modelo de seguridad propuesto en el Capítulo 2 se desarrolla el proceso de gestión de la seguridad en el Módulo Registro Mercantil, se identifican los principales artefactos de seguridad necesarios para garantizar que el producto final garantice los requisitos mínimos de seguridad de los recursos importantes una vez instalada la aplicación en un entorno cliente. El capítulo termina con la propuesta de los casos de prueba de seguridad para la aplicación Registro Mercantil.

CAPÍTULO.

1. FUNDAMENTACIÓN TEÓRICA DEL TEMA.

El tema de la Seguridad Informática es un requisito básico para los proveedores de software y ha tomado fuerza en los últimos años a partir de la evolución acelerada de las tecnologías de la información y su inclusión en casi todos los procesos de negocios de pequeñas y grandes empresas. Los desarrolladores de software se ven cada vez más comprometidos y presionados por las fuerzas del mercado dada la necesidad de preservar la confiabilidad de los procesos de negocios una vez llevados a una plataforma de software. Uno de los retos más importantes a los que se enfrentan todos los proveedores de software es crear productos con niveles de seguridad cada vez superiores donde las actualizaciones sean menos frecuentes y la administración de seguridad menos onerosa (Lipner and Howard 2005).

En el sector del software, la clave para cumplir la exigencia actual está en implementar procesos reproducibles que proporcionen niveles de seguridad medibles. Por tanto, los proveedores de software deben adoptar un proceso de desarrollo más estricto que incorpore, en mayor medida, aspectos esenciales de seguridad en todo el desarrollo. Este proceso debe diseñarse para minimizar el número de vulnerabilidades de seguridad presentes en el diseño, la programación y la documentación, así como para detectarlas y eliminarlas cuanto antes en el ciclo de vida del desarrollo de software. La necesidad de disponer de este proceso es mayor para productos de software que suelen operar sobre plataformas distribuidas de procesamiento de información procedente de Internet, manipulación de datos de identificación personal y otros entornos de gran importancia que pueden sufrir ataques (Lipner and Howard 2005).

En este capítulo se abordan temas relacionados con diferentes enfoques y conceptos en cuanto a la seguridad de software y elementos importantes que miden la confiabilidad de un sistema. El establecimiento y uso de las normas ISO para la seguridad de la información y sus secciones principales. Además como buen punto de partida para la elaboración del modelo en el Capítulo 2, se

clasifican dos tipos básicos de seguridad y algunos factores que llevan a preocuparse por la seguridad del software. También se analizan aspectos relacionados con el modelado de amenazas, ataques a los sistemas informáticos y la gestión de la seguridad, así como la problemática actual de la seguridad y los objetivos fundamentales para obtener un software confiable. Se hace un estudio del estado del arte sobre el surgimiento y evolución de metodologías y modelos elaborados por proveedores de software destinados al análisis y evaluación de riesgos en sus proyectos de desarrollo que dan como resultado una minimización de los principales problemas de seguridad que afectan a los sistemas hoy en día y se demuestran las potencialidades de usar elementos importantes de estas técnicas y metodologías a partir de experiencias y resultados concretos obtenidos por reconocidas compañías de desarrollo de software en el mundo como la Microsoft Corporation[3]. Además se justifica la selección del Proceso Unificado de Desarrollo de Software de Rational como marco a seguir, en cuyos artefactos y flujos se apoyará la elaboración del modelo.

1.1 Seguridad de software.

La definición de seguridad que se abordará en este capítulo está aplicada al desarrollo y uso de software, incorporando una serie de dimensiones que influyen en la clasificación final de software como confiable o no. La seguridad de software es un tema que, junto a muchos otros, se trata como algo borroso y su definición se maneja con cierto grado de incertidumbre, teniendo diferentes definiciones por distintas personas. Según el diccionario de la Real Academia Española de la Lengua (RAE), el término seguridad se refiere a la certeza o garantía de que algo va a cumplirse; mecanismo que previene algún riesgo o asegura el buen funcionamiento de alguna cosa, precaviendo que falle. Si se aplica el concepto al campo de la informática se puede definir que la Seguridad Informática generalmente consiste en asegurar que los recursos tecnológicos y activos tangibles e intangibles de un sistema de información (material informático o programas) de una organización sean utilizados de la manera que se decidió. Esto se traduce en que el sistema se encuentra en un estado que nos indica que está libre de peligro, daño o riesgo. Para la mayoría de los expertos el concepto de seguridad en la informática es utópico porque no existe un sistema 100% seguro por tanto el elemento de riesgo está siempre presente y coinciden más bien en denominar sistemas fiables.

[3] Compañía de desarrollo de software muy expandida en la actualidad, radicada en EE.UU. Microsofot Corporation es creadora y propietaria de la familia de sistemas operativos Windows.

Algunas fuentes la definen como "…una profesión compleja con funciones especializadas"(Manunta 2005). Otras la describen como la interrelación dinámica (competencia) entre el agresor y el protector para obtener (o conservar) el valor tratado, enmarcada por la situación global (A.S.S. Borghello 2001) y que "… es un problema de antagonismo y competencia. Si no existe un competidor-amenaza el problema no es de seguridad". Sin embargo todos coinciden en que las tendencias actuales en el mundo de la seguridad de software establecen que "El objetivo de la Seguridad Informática será mantener la Integridad, Disponibilidad, Privacidad, Control y Autenticidad de la información manejada por la computadora" (Aldegani 1997). Un elemento importante lo aporta Doshi Shreyas (Shreyas 2001), donde incluye el requisito Auditoría definiéndolo como un registro cronológico de los eventos relevantes a la seguridad de un sistema. Este registro puede luego examinarse para reconstruir un escenario en particular.

A partir de los elementos anteriores, puede concluirse que la seguridad de un sistema de software es la combinación de su capacidad para garantizar la integridad, disponibilidad, privacidad, control y autenticidad de la información y demás activos, que dada su importancia para el buen funcionamiento del sistema pueden poner en peligro los procesos habituales del mismo. Por tanto para que un sistema se pueda definir como fiable debe dotarse de estas características (Wikipedia 2007). Así, una falla en proteger cualquiera de esas características implica una violación de seguridad o vulnerabilidad. Por tanto se hace necesario dejar bien claro a qué se refiere cada uno de estos aspectos ya que su correcta comprensión es imprescindible para el trabajo posterior en el Capítulo 2.

1.1.1 Confidencialidad de la información.

La confidencialidad se refiere a que determinada información solo puede ser conocida por individuos autorizados y mediante los mecanismos establecidos. Existen infinidades de posibles ataques contra la privacidad, especialmente en la comunicación de los datos. La transmisión a través de un medio presenta múltiples oportunidades para ser interceptada y copiada. Algunos de los ejemplos más conocidos están relacionados con las líneas intervenidas, la intercepción o recepción electromagnética no autorizada o la simple intrusión directa en los equipos donde la información está físicamente almacenada (A.S.S. Borghello 2001). Es un requisito importante, que los sistemas de software, garanticen el acceso a los recursos, por parte de los usuarios que hacen uso de ellos, de manera controlada. Esto se debe a que muchos procesos de negocios manejan información que puede ser de uso confidencial tales como números de cuentas bancarias, resultados médicos y

científicos y donde un ataque contra la confidencialidad puede desencadenar en ganancias para unos y pérdidas para otros. Los clientes exigen hoy, que una vez montado un negocio sobre una plataforma informática, esta sea capas de garantizar el flujo normal de los procesos sin que se vea comprometida la confidencialidad de los activos. Algunos de los mecanismos implementados en la actualidad, se basan en la criptografía y cifrado de los datos.

1.1.2 Integridad de la información.

La integridad se refiere a la garantía de que la información no ha sido alterada, borrada, reordenada ni copiada por personas no autorizadas, bien durante el proceso de transmisión o en su propio equipo de origen. Es un riesgo común que el atacante al no poder descifrar un paquete de información y, conociendo su relevancia, simplemente lo intercepte y lo borre. Trabajar con información confiable en cuanto a su integridad es de relevancia extrema, sobre todo en aquellos sistemas que dependen de la exactitud de los datos suministrados, por determinadas fuentes, para el correcto funcionamiento de los procesos de negocios que maneja. Existen mecanismos que intentan mitigar ataques de este tipo, algunos están relacionados con la inclusión de firma electrónica en los documentos digitales.

1.1.3 Disponibilidad de la información.

La disponibilidad de la información está relacionada con la seguridad de que la información pueda ser recuperada en el momento que se necesite, esto es, evitar su pérdida o bloqueo, bien sea por ataque, mala operación accidental o situaciones fortuitas de fuerza mayor. Los sistemas de software que son críticos en cuanto a la importancia de mantener disponible determinados servicios de información, deben tener bien identificados aquellos puntos claves, y su manejo debe realizarse cuidadosamente, con el objetivo de reducir la posibilidad de sacar fuera de servicio elementos claves de información. Algunas empresas prestan servicios críticos en cuanto a su disponibilidad, tales como servicios de correo electrónico, mensajería instantánea, resultados actualizados de bolsas de valores, entre otros.

En esta definición de seguridad se involucran entonces diversos factores que llevan a preocuparse por la seguridad que el software presenta (Rodríguez and Grimaldi 2003):

El tamaño del software: A mayor cantidad de líneas de código escrita, mayor es nuestra probabilidad de cometer un error que vulnere la seguridad del mismo.

Importancia del software: Software poco usado o usado en tareas no críticas requiere, y recibe, menos revisión que aquel que es usado con frecuencia o en tareas críticas. Tanto los usuarios como

los atacantes estarán más preocupados de detectar, e informar las fallas que ocurran en este último tipo de software.

Costo de detección de la falla: El proceso de búsqueda de fallas se hace cada vez más tortuoso, a medida que se van arreglando los aspectos encontrados previamente. La preocupación por la fallas del sistema se mantiene en la medida en que el proceso de detección de fallas es menos costoso que el daño que la falla misma produce.

Costo de eliminación de la falla: Si la falla encontrada tiene un costo de reparación tan alto que implica el cambio completo del sistema, eso implica una evaluación inmediata del sistema en el que se la encontró como inseguro.

No existe el software sin bugs[4]: Este ideal sólo se podría cumplir con un tiempo de pruebas infinito.

Estos elementos por supuesto que no son los únicos que influyen directamente en la atención que debe prestarse a un sistema en cuanto a su seguridad, pero sí tienen un gran peso al medirla y gestionarla y todo arquitecto de seguridad debe tenerlas muy presentes durante el proceso de desarrollo.

1.1.4 Clasificaciones básicas.

Básicamente existen dos clasificaciones que son esenciales para ubicar la seguridad, dependiendo de las fuentes de amenazas. Esta puede dividirse en seguridad lógica y seguridad física.

1.1.4.1 Seguridad lógica.

Esta primera clasificación se refiere a la seguridad en el uso de software, la protección de los datos, procesos y programas, así como la del acceso ordenado y autorizado de los usuarios a la información (2006). La seguridad lógica generalmente es la que más se tiene en cuenta, ya que la mayoría de ataques que podamos recibir irán hacia el software y la información manipulada. Para una buena seguridad lógica se deben tener en cuenta muchos factores como es caso de buenas contraseñas, Sistemas de Detección de Intrusos (IDS por sus siglas en inglés), uso de permisos, así como la calidad del código escrito (Margini 2006), lo cual será tratado con más profundidad en el Capítulo 2.

[4] Término usado en las Ciencias de la Computación para referirse a un problema de seguridad presente en un sistema informático.

1.1.4.2 Seguridad física.

Debe aclararse que el tema relacionado con este tipo de seguridad se aleja del objetivo y el alcance de la tesis, debido a que puede constituir todo un tema para un trabajo posterior. Sin embargo se hace necesario, hacer referencia al concepto aunque sea brevemente.

La seguridad física se relaciona con la protección del hardware y los soportes de datos, así como la seguridad de los edificios e instalaciones que los albergan. En este caso se contemplan situaciones como incendios, inundaciones, sabotajes, robos, catástrofes naturales, entre muchos más. Este tipo de seguridad es una de las más olvidadas en el momento de desarrollar un sistema (2006). Sin embargo su importancia en nada es inferior a la seguridad física ya que muchas veces para un atacante resulta más factible destruir un determinada información provocando un daño físico directo a los sistemas que accediendo vía lógica a la información. Por tanto de nada vale obsesionarse con intentar proteger solo los activos lógicos de un sistema de software si con solo destruir el dispositivo donde este se encuentra todos los esfuerzos anteriores serían en vano.

1.1.5 Gestión de la seguridad.

Los dos últimos conceptos son fundamentales en el momento de gestionar la seguridad ya que son los dos elementos sobre los que incidirá básicamente el proceso de gestión que no es más que la realización de las tareas necesarias para garantizar los niveles exigibles y aceptables de seguridad tanto lógica como física en el desarrollo de un sistema de software. En el momento de gestionarla debe tenerse en cuenta varios aspectos importantes tales como: los problemas de seguridad no son únicamente de índole tecnológica; los riesgos no se eliminan... se gestionan; la seguridad no es un producto, es un proceso. Su gestión debe estar basada en la evaluación del riesgo y ser dinámica, debiendo comprender todos los niveles de las actividades de los participantes y todos los aspectos de sus operaciones. Asimismo ha de incluir posibles respuestas anticipadas a riesgos emergentes y considerar la prevención, detección y respuesta a incidentes que afecten a la seguridad, recuperación de sistemas, mantenimiento permanente, revisión y auditoría. Las políticas de seguridad de los sistemas y redes de información, así como las prácticas, medidas y procedimientos deben estar coordinadas e integradas para crear un sistema coherente de seguridad. Las exigencias en materia de gestión de seguridad dependerán de los niveles de participación, del papel que desempeñan los participantes, del riesgo de que se trate y de los requerimientos del sistema.

1.1.5.1 ¿Por qué gestionar?

Son múltiples las razones por las que es necesario gestionar la seguridad, primero porque garantizar la confidencialidad, integridad y disponibilidad de los activos es crítico para cualquier sistema software. Además porque las tecnologías con que desarrollamos los sistemas introducen amenazas y no siempre se pueden eliminar los riesgos. Sin embargo es importante tener claro que en el momento de gestionar la seguridad vale la pena analizar si realmente es imprescindible y cuales son los puntos críticos que lo necesitan; no es solo por el hecho de decir que el sistema tiene implementado mecanismos de seguridad que realmente no necesita, porque hay que recordar que los sistemas donde la seguridad es un factor determinante, son más difíciles de usar, ya que requieren, generalmente, sistemas de autenticación y control de acceso, lo que implica el uso de contraseñas fuertes que muchas veces son difíciles de recordad, restricciones sobre determinados recursos. Además porque al aplicar técnicas de cifrado y criptografía muchas veces el rendimiento de las aplicaciones se ve afectado en gran medida, lo que lleva a plantearse la disyuntiva de usabilidad contra seguridad.

1.1.5.2 Principios de gestión de seguridad.

Existen una serie de principios que no deben olvidarse nunca en el momento de gestionar la seguridad en el ciclo de vida del Proceso de desarrollo de software. A continuación se exponen una serie de estos principios divididos en dos categorías: principios generales y principios de programación (Howard and LeBlanc 2002).

1.1.5.2.1 Principios generales.

- Establecer procesos seguros.
- Aprender de los errores.
- Usar los privilegios imprescindibles.
- Practicar la defensa escalonada o en profundidad.
- Asumir que todos los sistemas externos son inseguros.
- Planificar la respuesta a fallos.

- Nunca implementar seguridad con “oscuridad”[5] .

1.1.5.2.2 Principios para la programación.

- Definir los objetivos y características del producto con respecto a la seguridad.
- Considerar los aspectos de seguridad como características intrínsecas al desarrollo de los productos de software.
- En caso de fallo, quedar con el nivel mínimo de privilegios.
- Establecer configuraciones confiables para el desarrollo así como diseñar las aplicaciones con la menor cantidad de privilegios por defecto con el objetivo de reducir la superficie de ataques.
- Tener presente que las características de seguridad de un producto no determinan que este sea seguro.

1.1.5.3 Garantizar la tolerancia a fallos.

Uno de los problemas más serios que enfrentan los sistemas de software está relacionado con la tolerancia a los fallos, que no es más que la capacidad de un sistema a responder a un suceso inesperado, como puede ser un fallo de programación, excepciones inesperadas no controladas o un fallo de hardware. Los riesgos como se planteó anteriormente no se eliminan, se mitigan, por lo que es muy importante tener en cuenta en el proceso de gestión de la seguridad que lo más relevante no siempre es garantizar que los sistemas no fallen, porque esa garantía es imposible en un 100%, sino garantizar que cuando fallen lo hagan elegantemente y de manera robusta.

No existe una manera estándar de establecer los pasos a tener en cuenta para implementar un buen sistema de gestión de la seguridad en las organizaciones. Sin embargo la Organización Internacional de Estándares (ISO) define una serie de normas, que constituyen un buen punto de referencia, las cuales se verán a continuación de manera general.

[5] Este término se usa cuando se implementa software y se esconden puntos débiles de seguridad en el diseño o la implementación, los cuales pueden desencadenar en vulnerabilidades. Los desarrolladores de este tipo de seguridad creen que estos puntos débiles son poco probables de descubrir por terceros y por tanto difíciles de atacar.

1.1.5.4 Emplear las normas de gestión de la seguridad de la información ISO/IEC 17799 e ISO/IEC 27001.

En toda organización que haga uso de las tecnologías de información se recomienda implementar buenas prácticas de gestión de seguridad de la información. En muchas ocasiones el no seguir un proceso de implementación adecuado como el que establece el ISO 17799 puede generar huecos por la misma complejidad de las organizaciones, en ese sentido, aumenta la posibilidad de riesgos en la información (Sandoval 2005). La serie ISO 27000 comprende un conjunto de normas relacionadas con la seguridad en este aspecto.

El resumen de normas es:

- **ISO 27000.** Conjunto de vocabulario y definiciones (terminología para el resto de estándares de la serie).

- **ISO 27001.** Especificación del sistema de gestión de la seguridad de la información (SGSI). Esta norma será certificable bajo los esquemas nacionales de cada país.

- ISO 27002. Actualmente la ISO 17799:2005, que describe el Código de buenas prácticas para la gestión de la seguridad de la información.

- ISO 27003. Contendrá una guía de implementación.

- ISO 27004. Estándar relacionado con las métricas y medidas en materia de seguridad para evaluar la efectividad del sistema de gestión de la seguridad de la información.

- **ISO 27005**, que proporcionará el estándar base para la gestión del riesgo de la seguridad en sistemas de información (Sandoval 2005; Dutra 2006). A su vez, la norma ISO 17799:2000 sufrió modificaciones, teniendo ahora una nueva versión, la ISO 17799:2005 que proporciona recomendaciones de las mejores prácticas en la gestión de la seguridad de la información a todos los interesados y responsables en iniciar, implantar o mantener sistemas de gestión de la seguridad de la información (Figura 1). La seguridad de la Información se define en el estándar como la preservación de la confidencialidad (asegurando que sólo quienes estén autorizados pueden acceder a la información), integridad (asegurando que la información y sus métodos de proceso son exactos y completos) y disponibilidad (asegurando que los usuarios autorizados tienen acceso a la información y a sus activos asociados cuando lo requieran).

Figura 1. Evolución del estándar ISO 17799.

La versión de 2005 del estándar incluye las siguientes once secciones principales:

a) Políticas de seguridad (Estrada 2006);

b) Seguridad organizacional;

c) Clasificación y control de activos;

d) Seguridad del personal;

e) Seguridad física y de entorno;

f) Comunicaciones y administración de operaciones;

g) Control de acceso;

h) Desarrollo de sistemas y mantenimiento;

i) Continuidad de las operaciones de la organización;

j) Requerimientos legales;

Cada una de las áreas establece una serie de controles que serán seleccionados dependiendo de los resultados obtenidos en el análisis de riesgos, además, existen controles obligatorios para toda organización, como es el de las políticas de seguridad cuyo número dependerá más de la organización que del estándar, el cual no establece este nivel de detalle (Sandoval 2005). Para cada uno de los controles se indica asimismo una guía para su implantación. El número total de controles suma 133 entre todas las secciones aunque cada organización debe considerar previamente cuantos serán realmente los aplicables y según sus propias necesidades.

Con la aprobación de la norma ISO/IEC 27001 en Octubre de 2005 y la reserva de la numeración 27000 para la seguridad de la información, se espera que ISO/IEC 17799:2005 pase a ser

renombrado como ISO/IEC 27002 en la revisión y actualización de sus contenidos en el 2007 (2005; 2005).

La correcta selección de los controles es una tarea que requiere del apoyo de especialistas en Seguridad Informática, con experiencia en la implementación del ISO 17799, ya que cuando éstos se establecen de forma inadecuada pueden generar un marco de trabajo demasiado estricto y poco adecuado para las operaciones de la organización (Sandoval 2005).

Un aspecto importante que se debe tener en cuenta es que el estándar no elimina el cien por ciento de los problemas de seguridad pero como todo estándar, el ISO 17799 proporciona un marco ordenado de trabajo al cual deben sujetarse todos los integrantes de la organización, donde establece una valoración de los riesgos a los que se enfrenta una organización en materia de seguridad de la información. Dicha valoración permite administrar los riesgos en función de los recursos tecnológicos y humanos con los que cuenta la organización; adicionalmente, establece un entorno que identifica los problemas de seguridad en tiempos razonables, situación que no es posible, la mayoría de las veces, si no se cuenta con controles de seguridad como los establecidos en el ISO 17799, es decir, la aplicación del estándar garantiza que se podrán detectar las violaciones a la seguridad de la información, situación que no necesariamente ocurre en caso de no aplicarse el estándar (Estrada 2006). El estudio del estándar aportará en el desarrollo del modelo elementos claves relacionados con la clasificación y control de activos, control de acceso entre otros elementos.

1.1.5.5 Gestión de amenazas y ataques a los sistemas informáticos.

En el proceso de gestión de la seguridad ocupa un lugar importante el análisis de amenazas y los ataques que pueden tener efecto sobre un sistema de software. Teniendo en cuenta que uno de los puntos claves dentro del desarrollo del modelo es precisamente el artefacto *Modelo de Amenazas* es determinante analizar algunos elementos relacionados con este.

Las amenazas constituyen una condición del entorno del sistema de información (persona, máquina, suceso o idea) que, dada una oportunidad, podría dar lugar a que se produjese una violación de la seguridad (confidencialidad, integridad, disponibilidad o uso legítimo). Las políticas de seguridad y el análisis de riesgos habrán identificado las amenazas que han de ser contrarrestadas, dependiendo del diseñador del sistema de seguridad, especificar los servicios y mecanismos de seguridad necesarios (Marañón 2005).

Tipos de amenazas.

Existen 3 tipos básicos de amenazas, aquellas que se realizan contra la red, las que se efectúan contra los servidores, equipos de cómputo, locales y finalmente las que se realizan contra las aplicaciones. Las defensas para amenazas contra la red y contra los servidores son casi en todos los casos administrativas, las defensas de las amenazas contra el aplicativo normalmente consisten en escribir código seguro. Sin embargo en este capítulo no se entrará en muchos detalles acerca de cada uno de estos tipos específicos ya que la construcción del modelo tratará con más profundidad estos aspectos sobre todo las amenazas concernientes a las aplicaciones.

El modelado de amenazas.

El modelado de amenazas es una técnica de ingeniería cuyo objetivo es ayudar a identificar de forma correcta y estructurada aquellos eventos que valiéndose de un problema de característica de seguridad en un sistema pueden desencadenar acciones mal intencionadas. Ayuda a colocar prioridad en el uso de los recursos de manera que se logra un resultado más seguro de acuerdo al desarrollo que se está estableciendo. Constituye una parte esencial del proceso de desarrollo, tanto como, el análisis y diseño, la codificación y las pruebas. Para aprovechar mejor los beneficios que proporciona su uso, en un escenario ideal, debería iniciarse desde las primeras etapas del desarrollo y planificación de cualquier aplicación o sistema (P.F 2006). La verdadera ventaja de esta técnica reside en el hecho de que al aplicarse como un proceso durante todo el ciclo de vida de desarrollo del software, supone un enfoque diferente al tradicional análisis de riesgos y la posterior aplicación de medidas que contribuyan a mejorar la seguridad. Es por esto que el análisis y modelado de amenazas, aunque es un campo relativamente nuevo, está despertando un gran interés.

El análisis y modelado de amenazas, trata por lo tanto, de un proceso que va a facilitar la comprensión de las diferentes amenazas de seguridad a las que va a estar expuesto un sistema o una aplicación, y cuya finalidad es preparar las defensas adecuadas durante las fases de diseño, implementación y también durante su posterior revisión y pruebas. Se trata de identificar las amenazas y definir una estrategia adecuada que nos permita mitigar el riesgo a unos niveles aceptables, de una forma efectiva y en base a unos costos razonables. En los epígrafes 1.3 y epígrafe 1.4 se abordará con más detalle el tema del modelado de amenazas, cuando se analicen algunas metodologías que sobre este asunto han evolucionado y difundido en el mundo y las

herramientas asociadas a estas que permiten humanizar el trabajo de gestión a través de la automatización de los procesos establecidos por cada metodología.

El proceso de modelado de amenazas consiste básicamente en seguir los siguientes pasos: identificar los recursos, documentar la arquitectura, descomponer la aplicación, identificar las amenazas y documentarlas, clasificar estas amenazas y finalmente elaborar el plan de medidas que permitirá establecer entonces, mecanismos de protección, lo más tempranamente posible en el desarrollo del sistema. Al evaluar amenazas que afectan los sistemas informáticos deben tenerse en cuenta tres momentos importantes:

- **Antes del ataque:** Debemos tener en cuenta los mecanismos que aumentan la seguridad del sistema en cuestión durante su funcionamiento habitual.
- **Durante el ataque:** Implementar mecanismos orientados a revelar violaciones a la seguridad de los sistemas.
- **Después del ataque:** Disponer de mecanismos y técnicas que se aplican luego que la ejecución de un ataque se ha detectado (Marañón).

Los ataques.

El ataque no sería otra cosa que la ejecución de una amenaza (Marañón). Estos pueden dividirse en dos categorías fundamentales:

Ataques pasivos.

En los ataques pasivos el atacante no altera la comunicación, sino que únicamente la escucha o monitoriza, para obtener información que está siendo transmitida. Sus objetivos son la intercepción de datos y el análisis de tráfico, una técnica más sutil para obtener información de la comunicación, que puede consistir en:

- Obtención del origen y destinatario de la comunicación, leyendo las cabeceras de los paquetes monitorizados.
- Control del volumen de tráfico intercambiado entre las entidades monitorizadas, obteniendo así información acerca de actividad o inactividad inusuales.
- Control de las horas habituales de intercambio de datos entre las entidades de la comunicación, para extraer información acerca de los períodos de actividad.

Los ataques pasivos son muy difíciles de detectar, ya que no provocan ninguna alteración de los datos. Sin embargo, es posible evitar su éxito, en gran medida, mediante el cifrado de la información y otros mecanismos que se verán más adelante.

Ataques activos.

Estos ataques implican algún tipo de modificación del flujo de datos transmitido o la creación de un falso flujo de datos, pudiendo subdividirse en cuatro categorías:

- *Suplantación de identidad:* el intruso se hace pasar por una entidad diferente. Normalmente incluye alguna de las otras formas de ataque activo. Por ejemplo, secuencias de autenticación pueden ser capturadas y repetidas, permitiendo a una entidad no autorizada acceder a una serie de recursos privilegiados suplantando a la entidad que posee esos privilegios, como al robar la contraseña de acceso a una cuenta.
- *Reactuación:* uno o varios mensajes legítimos son capturados y repetidos para producir un efecto no deseado, como por ejemplo ingresar dinero repetidas veces en una cuenta dada.
- *Modificación de mensajes:* una porción del mensaje legítimo es alterada, o los mensajes son retardados o reordenados, para producir un efecto no autorizado.
- *Degradación fraudulenta del servicio:* impide o inhibe el uso normal o la gestión de recursos informáticos y de comunicaciones. Por ejemplo, el intruso podría suprimir todos los mensajes dirigidos a una determinada entidad o se podría interrumpir el servicio de una red inundándola con mensajes espurios. Entre estos ataques se encuentran los de denegación de servicio, consistentes en paralizar temporalmente el servicio de un servidor de correo, Web, FTP, entre otros.

1.2 Problemática actual de la seguridad en el Proceso de desarrollo de software.

La Ingeniería de Software enfrenta hoy en día básicamente dos puntos débiles relacionados con la problemática actual de la seguridad en el proceso de implementación de soluciones de software. Estos puntos pueden ser clasificados en dos grandes categorías: Fallas para implementar software seguro y Fallas para implementar seguridad en el software, las cuales se especificarán a continuación (Asteasuain and Schmidt).

1.2.1 Fallas para implementar software seguro.

Lamentablemente, la mayoría de las herramientas que tiene disponible un desarrollador de software sufren de fallas propias de seguridad. Lo cual es una realidad y que debe tenerse en cuenta constantemente. Una de las debilidades más trascendentes al momento de implementar software seguro surge del estado de los lenguajes de programación desde el punto de vista de la seguridad. Son escasos los lenguajes que proveen primitivas “seguras” que ayuden al programador a escribir un mejor código. Ejemplos de estos lenguajes son C y C++. El desarrollador ingenuamente confía casi siempre en las librerías estándar que traen implementadas el propio lenguaje y pasan inadvertidos elementos de seguridad que luego se traducen en huecos y vulnerabilidades difíciles de corregir (Viega, Bloch et al. 2000). Este tema será ampliado en el Capítulo 2 cuando se profundice en el uso de guías de código como artefacto a incluir en el modelo desarrollado para la gestión de la seguridad. Sin embargo lenguajes más recientes con arquitecturas de seguridad mucho más complejas, como Java, aún no satisfacen todas las necesidades. Un alto porcentaje de aplicaciones tiene problemas de seguridad que están presenten desde la fase de diseño y que permanecen hasta la implementación, independientemente del lenguaje de programación usado (Viega, Bloch et al. 2001).

Las razones por las cuales estas fallas “internas” permanecen en la actualidad son varias: mal entendimiento de los protocolos de seguridad, una visión ingenua respecto a lo que un sistema debiera considerar como seguro, aproximaciones poco serias a la seguridad como “corregir luego” o “no se van a dar cuenta”, o directamente desconocimiento, ya que lamentablemente estas fallas no son conocidas universalmente, y existen pocas fuentes de información para escribir código seguro. Otro tipo de falla en esta categoría, nace del hecho de que la seguridad es un tema complejo y requiere un entendimiento completo sobre lo que puede ir mal y qué es lo que puede ser explotado por un posible atacante (Win, Vanhaute et al. 2001). Un programador promedio no cuenta con la experiencia suficiente como para poder determinar los requerimientos de seguridad que necesite su aplicación. Esto resulta en la subestimación de pequeños detalles que luego pueden llegar a introducir grandes fallas de seguridad.

1.2.2 Fallas para implementar seguridad en el software.

Es una realidad que la Ingeniería de Software recién está aprendiendo sobre la seguridad. Esto se debe a que las metodologías de desarrollo de software actuales en su mayoría, entre las que puede

mencionarse a RUP consideran la seguridad como un requerimiento no funcional. Luego, debido a los problemas de planificación y presupuesto, la seguridad sólo es tenida en cuenta una vez que los requerimientos funcionales son obtenidos. Esto conduce a que la seguridad sea considerada como un concepto "afterthougth" [6] [10], y que se incorpora tardíamente al sistema. Esto lleva a una implementación pobre, ineficiente, e inadecuada de la seguridad. También, la mayoría de las metodologías de diseño y herramientas dedicadas a la seguridad trabajan de esta forma, como herramientas "afterthougth". Por lo tanto, es necesario que los conceptos de seguridad formen parte integral en todo el ciclo de vida de desarrollo de software [30].

Una de las aproximaciones más ampliamente utilizada para la seguridad es la aproximación "ataque-parche", donde la seguridad es tratada a medida que las fallas se van revelando. Esto es, se desarrolla el sistema con mínimas consideraciones con respecto a la seguridad. Luego, una vez que el sistema está funcionando se detectarán los ataques al mismo, y se buscará, en ese momento, la manera de corregirlos. Bajo esta aproximación, claramente, es imposible implementar la seguridad de una manera adecuada.

1.2.2.1 Limitación de los estándares de codificación.

Dentro de las fallas que con más frecuencia se encuentran dentro del proceso de desarrollo para implementar seguridad en el software está la relacionada con el uso de guías de codificación por parte del equipo de desarrolladores que contribuyan a escribir código de calidad. Por eso se hace necesario dedicar un segmento específico dentro de este epígrafe para tratar el tema.

Los estándares de codificación son reglas específicas a un lenguaje que reducen perceptiblemente el riesgo de que los desarrolladores introduzcan errores. Estos no destapan problemas existentes, evitan más bien que los errores ocurran. Los bugs frecuentes en programas pueden ser detectados con anterioridad o pueden incluso ser evitados totalmente. Durante el desarrollo, los estándares de codificación ayudan a los ingenieros a producir un código de alta calidad así como entender y utilizar el código de sus compañeros. Pero también realzan considerablemente la capacidad de mantenimiento y rehúso a largo plazo del producto final. Tal práctica del control de bugs en el proceso de desarrollo mejora la calidad mientras que reduce el tiempo de desarrollo, el coste, y el esfuerzo . En la actualidad el uso de estándares de codificación se ha convertido en uno de los

[6] Término usado para referirse a algo aplicado tardíamente.

principios básicos para los diseñadores de software a nivel mundial por las ventajas que aporta su aplicación en el proceso de desarrollo de sistemas informáticos . Sin embargo las guías existentes reflejan en su mayoría una pobreza en cuanto a la inclusión de elementos de seguridad en su contenido. Independientemente de que se persiga una mayor uniformidad en el código, facilidad de lectura y mantenimiento futuro; las guías diseñadas y adoptadas en el desarrollo de software deben contener elementos capaces de garantizar: la reducción de la probabilidad de introducir errores; precisar algunos errores ocultados o inesperados; evitar que los bugs se propaguen en su software; así como aumentar la confiabilidad.

Aunque el propósito principal para llevar a cabo revisiones del código a lo largo de todo el desarrollo es localizar defectos en el mismo, las revisiones también pueden afianzar los estándares de codificación de manera uniforme. La adopción de un estándar de codificación sólo es viable si se sigue desde el principio hasta el final del proyecto de software. No es práctico, ni prudente, imponer un estándar de codificación una vez iniciado el trabajo (Fernández 2005). En el Capítulo 2 se abordará con más profundidad este tema.

1.2.3 Objetivos para un software confiable.

Seguridad desde el comienzo: la seguridad debe ser considerada como un requerimiento desde el inicio del diseño.

Uniformidad: la seguridad debe aplicarse de manera correcta y consistente a través de toda la aplicación y del proceso que desarrolla la misma.

Ambiente seguro: se debe partir de un entorno confiable. Es decir, las herramientas de desarrollo y lenguajes de programación deben contener un nivel bajo de agujeros de seguridad.

1.3 Evolución de las metodologías para el modelado y análisis de riesgos.

El surgimiento de metodologías destinadas al modelado y análisis de riesgos cada vez se hacen más frecuentes, fundamentalmente a partir del año 2000 donde según las investigaciones realizadas se ha hecho más evidente el tema. Sin embargo estas metodologías muchas veces son desarrolladas para entornos de desarrollo específicos y de manera interna por algunas empresas para su autogestión. Dentro de las metodologías que intentan ayudar en la medición y mitigación del riesgo inherente al desarrollo de software, algunas de las más conocidas son:

- Modelación y Análisis de Amenazas (TAM por sus siglas en inglés).

- CORAS.
- Trike.
- Análisis Práctico de Amenazas (PTA)[7].

1.3.1 Modelación y Análisis de Amenazas.

Precisamente es Microsoft una de las compañías que más ha avanzado en el tema de la seguridad para garantizar la calidad de los productos de software que produce con vistas al mercado. Microsoft ha desarrollado una metodología de análisis y modelado de amenazas, basada en la combinación de ideas propias y de @stake[8] que recientemente ha incorporado a sus filas. Esta metodología ha ido evolucionando y recogiendo ideas de diversos enfoques. En las primeras versiones, principalmente se apoyaba en el uso de árboles de ataques para luego realizar una clasificación de las amenazas y establecer una puntuación con el fin de priorizar las actuaciones necesarias para mitigar el riesgo (P.F 2006). Sin embargo ya en la segunda versión de esta metodología, han establecido de forma más acabada los conceptos de amenaza, ataque y vulnerabilidad, perfeccionando la herramienta de modelado y cambiando sustancialmente los aspectos originales (P.F 2006).

La metodología propuesta por Microsoft sigue una serie de pasos y puntos de vista muy bien definidos y delimitados a partir de los casos de uso del sistema. 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. Luego se establece cual es el nivel de seguridad del sistema ya sea mediante casos de uso, conociendo las distintas dependencias o utilizando modelos del sistema. Se determina cuales son las amenazas: se identifican las amenazas, se analizan y clasifican, se identifican las vulnerabilidades a las que se ve expuesto el sistema (P.F 2006).

El uso de algunos métodos durante el desarrollo del modelado, como son los árboles de ataques y los diagramas de flujo de datos, permiten conocer cuales son los activos de mayor interés, los puntos de entrada y los niveles de confianza. Lo que conlleva a la identificación de riesgos y amenazas que a simple vista pueden pasar desapercibidos y desencadenar serios bugs de seguridad más adelante. Realmente el uso de estos métodos constituye una buena manera de comprender la lógica de la

[7] Las siglas responden a su nombre en inglés: Practical Threat Analysis.

[8] Compañía de seguridad informática radicada en Cambridge, Massachusetts, EE.UU.

aplicación y saber cómo puede afectar el tratamiento de los datos a la integridad de los activos (Howard and LeBlanc 2002).

Los árboles de ataques, ayudan a identificar las amenazas y analizar cuales serían las formas de mitigarlas. El objetivo del atacante se sitúa en el nodo principal del árbol. Los nodos hijo representan las diferentes formas que tiene el atacante de conseguir sus objetivos. Estos nodos (subobjetivos) pueden representar métodos mutuamente exclusivos de conseguir el objetivo padre, o bien nodos que representen todas las acciones a efectuar necesarias para conseguir un objetivo. Una vez efectuados los pasos iniciales del proceso, se procede a realizar una clasificación y puntuación de las amenazas, de modo que el equipo de trabajo pueda determinar las distintas prioridades.

Microsoft define dos esquemas principales de clasificación y puntuación: STRIDE[9] y DREAD[10].

El esquema STRIDE es básicamente un sistema de clasificación, mientras que DREAD es un modelo diseñado fundamentalmente para establecer la puntuación de una amenaza. En entornos de desarrollo reales, para la evaluación y análisis de riesgos y amenazas lo que se hace principalmente es complementar ambos métodos, proporcionando así excelentes resultados partiendo siempre de la situación específica de cada negocio.

1.3.1.1 Método de clasificación STRIDE.

El método establece la descomposición del sistema en componentes significativos y cruciales, tras analizar cada componente para determinar si es importante en cuanto a la posibilidad de sufrir amenazas, se proponen acciones que traten de mitigarlas. A continuación, se repite el proceso hasta llegar a una situación aceptable con las amenazas restantes(Howard and LeBlanc 2002). Como se utiliza un modelo de información basado en patrones, se podrán identificar los patrones de soluciones y problemas repetibles y organizarlos en categorías. Utilizando estas categorías para descomponer la aplicación para un mayor análisis, e identificando las vulnerabilidades de la aplicación relacionadas con cada categoría. De esta manera, se promueve la reutilización de la información y una comunicación más eficaz (P.F 2006).

---

[9] Acrónimo de (Spoofing identity, Tampering with data, Repudiation, Information disclosure, Denial of service, Elevation of privilege), por sus siglas en inglés.

[10] Acrónimo de (Damage potential, Reproducibililty, Exploitability, Affected users, Discoverability), por sus siglas en inglés.

Seleccionar cada activo por grado de importancia y analizar si es susceptible a los siguientes elementos.

- Suplantación de Identidad: Los usuarios no deberían ser capaces de hacerse pasar por otros usuarios. Es necesario disponer de una gestión apropiada de los procesos de autenticación. En particular, es aconsejable prestar atención a aquellas situaciones en las que pueda ser posible realizar un robo de sesiones o en aquellas en las que se deba valorar una inapropiada implementación de los sistemas de autenticación que pueda suponer un riesgo excesivo para la seguridad global del sistema.
- Manipulación de datos: Durante el desarrollo de software, por desgracia es habitual sacrificar la implementación de medidas de seguridad en beneficio de unos menores tiempos de desarrollo. Una de las primeras medidas preventivas que se suelen sacrificar es el filtrado y la validación de los datos enviados a y recibidos de los usuarios de la aplicación. Confiar en exceso en la buena fe del usuario y las diferentes entradas es un error que puede costar muy caro. Tanto la validación como la integridad de los datos son cuestiones de vital importancia.
- Repudio: Establecer un nivel adecuado del seguimiento de las acciones realizadas por los usuarios de la aplicación puede evitar la aparición de situaciones no deseadas. Se debe intentar garantizar el no repudio de los usuarios.
- Revelación de información: Cualquier información que pueda facilitar la tarea a un atacante proporcionando detalles del funcionamiento de la propia aplicación o del usuario que favorezcan una posterior explotación, o información cuya obtención y/o divulgación suponga una pérdida de confianza y reputación de cara a posibles o ya existentes usuarios y clientes.
- Denegación de servicio: Situaciones que puedan devengar en la no disponibilidad de algún servicio ofrecido por la aplicación y sus dependencias (Howard and LeBlanc 2002).
- 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 conlleven el uso de privilegios deben ser filtradas a través de un mecanismo adecuado de autorización. Este método de validación de los privilegios deberá ser lo suficientemente robusto para impedir una posible escalada de privilegios.

1.3.1.2 Método de puntuación DREAD.

A partir de una lista de amenazas identificadas, se procede a establecer su puntuación de acuerdo al riesgo que suponen. Esto permitirá priorizar las medidas a tomar para mitigar el riesgo, ya que muchas veces las medidas más fuertes implican gastos bastante costosos.

Existe un método clásico de evaluación del riesgo que supone una amenaza y consiste en multiplicar la probabilidad de que la amenaza se produzca, por el daño potencial de esta (P.F 2006).

En forma de ecuación matemática el riesgo se calcula:

Riesgo = Probabilidad * Daño potencial.

Este método hace uso de 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 daño potencial, 1 indicaría un daño mínimo y 10 un daño máximo. Este enfoque, bastante sencillo, permite en un primer momento clasificar las amenazas en una escala entre 1-100 que se puede dividir en tres partes según su riesgo: Alto, Medio, Bajo.

Ejemplo 1: Probabilidad 10, Daño potencial 7, el riesgo será: Riesgo = 10 * 7= 70

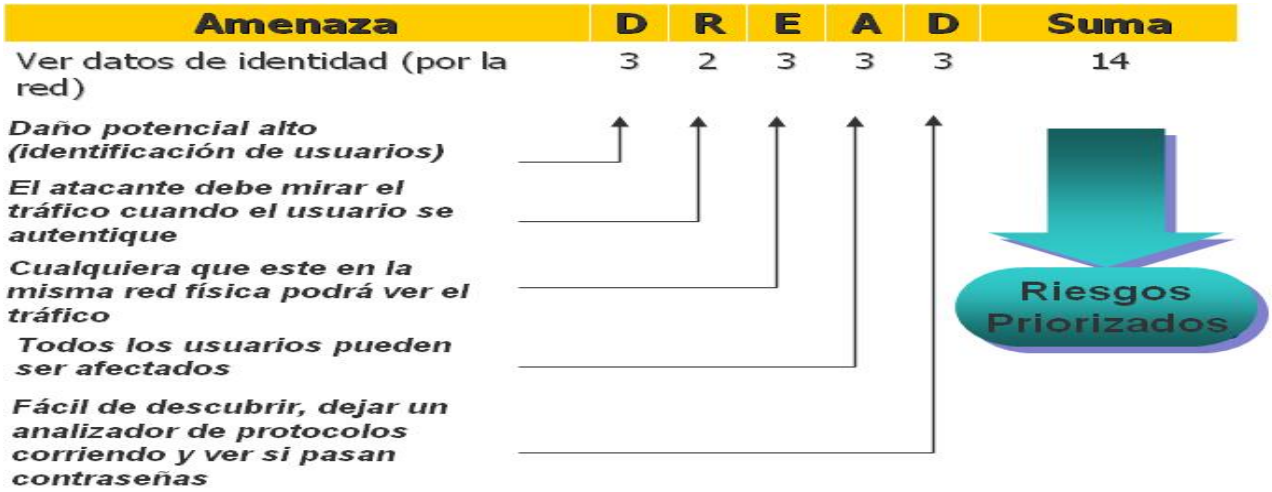
Una amenaza con un riesgo alto, debería ser paliada de forma rápida. Una amenaza con un riesgo medio, aunque importante, es de menor urgencia, y por último las de riesgo bajo, se podrían llegar a ignorar dependiendo del coste y del esfuerzo necesarios (Shreyas 2001).

Sin embargo el problema de aplicar este sencillo método, radica en la dificultad de valorar de igual forma el riesgo cuando esta valoración se efectúa entre varias personas (Howard and LeBlanc 2002), ya que los criterios suelen en muchos casos ser divergentes.

Para contrarrestar esta dificultad Microsoft desarrolló un método igual de sencillo pero más efectivo: el método DREAD, el cual trata de facilitar el uso de un criterio común respondiendo a las siguientes cuestiones:

- Daño potencial: ¿Cual es el daño que puede originar la vulnerabilidad si llega a ser explotada?
- Reproducibilidad: ¿Es fácil reproducir las condiciones que propicien el ataque?
- Explotabilidad: ¿Es sencillo llevar a cabo el ataque?
- Usuarios afectados: ¿Cuantos usuarios se verían afectados?
- Descubrimiento: ¿Es fácil encontrar la vulnerabilidad?

A cada criterio se le da un valor entre uno y tres y se suman todos, el valor de dicha suma es el riesgo que implica la amenaza. Para saber si implica un riesgo alto se procede a establecer rangos: una media en el rango 12-15 sería considerada un riesgo alto, entre 8-11 medio, y entre 5-7 bajo.



1.3.2 Metodología CORAS.

CORAS es un proyecto creado por la unión europea con el objetivo de proporcionar un marco de trabajo orientado a sistemas donde la seguridad es crítica, facilitando el descubrimiento de vulnerabilidades de seguridad, inconsistencias, y redundancias. CORAS proporciona un método basado en modelos, para realizar análisis de riesgos, y se basa en el uso de tres componentes (Group 2004):

- Un lenguaje de modelado de riesgos basado en el UML.
- La metodología CORAS, una descripción paso por paso del proceso de análisis con una directriz para construir los diagramas CORAS.
- Una herramienta para documentar, mantener y crear los informes del análisis.

La metodología CORAS, puede contribuir a la reducción de riesgos y la adopción de contramedidas haciendo un uso intensivo de los diagramas existiendo 5 tipos diferentes (P.F 2006):

- **Diagrama superficial de activos:** Muestra una visión general de los activos y cómo el daño sobre un activo puede afectar al resto.
- **Diagrama de amenazas:** Muestra una visión completa de la secuencia de eventos iniciados por las amenazas y las consecuencias que tienen éstas sobre los activos. Sus componentes

básicos son: amenazas deliberadas, amenazas accidentales, amenazas no-humanas, vulnerabilidades, escenarios de amenazas, incidentes no deseados y activos.

- **Diagrama superficial de riesgo:** Es un resumen del diagrama de amenazas, mostrando los riesgos. Tiene 5 componentes básicos: amenazas deliberadas, accidentales y no-humanas, riesgos y activos. A cada riesgo se le asigna un valor.
- **Diagrama de tratamiento:** Ofrece una visión completa de las contramedidas propuestas. Se basa en el diagrama de amenazas, sustituyendo las consecuencias del impacto sobre los activos con los riesgos procedentes del diagrama superficial de riesgo, y añadiendo los escenarios de contramedidas propuestos.
- **Diagrama superficial de tratamiento:** Es un resumen de las contramedidas, añadiendo los distintos escenarios posibles y mostrando las relaciones entre los distintos elementos propuestos para tratar el riesgo(P.F 2006).

La metodología CORAS mediante el uso del numeroso grupo de diagramas vistos anteriormente se convierte en una metodología muy práctica si se tiene en cuenta que el proceso de gestión de la seguridad implica muchos de estos elementos, que ofrecen ideas y representaciones claras de los diferentes artefactos durante el modelado de amenazas.

1.3.3 Metodología Trike.

Es una metodología conceptual, acompañada por una herramienta que intenta facilitar el proceso de modelado. La metodología, está diseñada con el propósito de permitir al Analista de Sistema describir de forma completa y precisa las características de seguridad de un sistema, desde los detalles de alto nivel de la arquitectura, hasta la implementación. O al menos en teoría. Según sus autores, Trike está aún en desarrollo, y aunque supuestamente debería proporcionar un nivel suficiente de detalle para permitir su uso práctico en un entorno real, realmente hay muchas dudas. La última versión disponible de la herramienta, data del 7 de Febrero del año 2006, y el borrador de la metodología es de 2005. Por lo que no parece que estén muy pendientes de su actualización (Trike 2005).

1.3.4 Metodología PTA.

La empresa PTA Technologies ha desarrollado su propia metodología que trata de solventar las limitaciones que según ellos tiene la versión 1 de la metodología propuesta por Microsoft (P.F 2006). PTA CTMM (Calculative Threat Modeling Methodology por su nombre en inglés. Antes de comenzar

el proceso de modelado, el Analista de Sistema debe familiarizarse con la aplicación /sistema (P.F 2006). Resultando particularmente útil recopilar la siguiente documentación con el fin de que sirva de ayuda en el momento de decidir si se aplican o no los distintos escenarios del sistema que se modelará:

- Descripción funcional del sistema donde se incluyan casos de uso típicos.
- Diagrama de la arquitectura del sistema y documentación de los distintos módulos.
- Un diccionario de términos, donde se expliquen los distintos vocablos utilizados en los restantes documentos.

La metodología propone varias etapas a través de las cuales:

- Se identifican los activos y sus vulnerabilidades, dependiendo de la arquitectura, funcionalidad y la lógica del negocio.
- Se definen las contramedidas en función de las vulnerabilidades y los costos que suponen.
- Se crean los escenarios de amenazas y planes de mitigación.

Esta parece ser una buena metodología al menos por las características que describen su funcionamiento, aunque durante la investigación no se encontraron resultados que puedan dar una posición al respecto.

Si bien el surgimiento y uso de estas metodologías, aún no garantizan la identificación y mitigación de todos los riesgos y amenazas que pueden estar presentes durante le Proceso de desarrollo de software, sí hay que reconocer que constituyen un punto de apoyo determinante hoy en día para los proveedores de software que intentan ganarse un puesto líder en el mercado. En el epígrafe 1.6 se da una muestra de cuan efectiva puede resultar la aplicación de las metodologías de evaluación de riesgos en el desarrollo de software, a partir de resultados concretos obtenidos. Es válido aclarar que ningún método de evaluación y clasificación de riesgos es mejor que otro, ni existe una forma única de hacerlo, una buena opción puede ser suplir las deficiencias de un método con las virtualidades del otro y viceversa. Aunque siempre deben analizarse aspectos tales como el daño potencial que puede acarrear el riesgo así como la probabilidad de que se explote una vulnerabilidad y se ejecute un ataque. Los métodos propuestos por Microsoft en el epígrafe 1.3.1 proporcionan una manera relativamente fácil de clasificar y dar puntuación a los mismos una vez identificados. Ninguna de las metodologías analizadas es peor que la otra, simplemente fueron concebidas en ambientes diferentes e incluso algunas no han sido explotadas en todo su potencial o su diseño aún no se ha terminado.

La clave está en apoyarse en aquellos aspectos individuales de cada una que creemos pueden ayudar y a partir de ahí elaborar nuestro propio modelo.

Sin embargo por las investigaciones realizadas y los resultados obtenidos parece ser que la metodología propuesta por Microsoft ha dado resultados concretos. Al final del capítulo se dan algunas muestras de la efectividad de aplicar dicha metodología en los procesos de desarrollo de software de la compañía.

El modelo propuesto usará como procedimientos para clasificación de las amenazas y evaluación de riesgos, los métodos STRIDE y DREAD, respectivamente descritos cuando se analizó la metodología TAM (Ver Anexos 1 y 2).

1.4 Herramientas de modelado de amenazas y análisis de riesgos.

Desarrollar el Modelo de amenazas y el análisis de riesgos es una tarea que requiere constante actualización. Esto se debe al hecho de que, a medida que se avanza en los flujos que guían el desarrollo, aparecen nuevos elementos que forman parte de este proceso; ya sean activos, roles, riesgos, amenazas; lo que hace necesario revisar el modelado para incluirlos, relacionarlos y analizar el impacto que tienen sobre otros elementos ya identificados. Además se generan nuevos diagramas y flujos que normalmente requieren cuidado y tiempo. Realizar estas tareas de forma manual, mediante dibujos y otros medios puede resultar una labor tediosa y casi imposible de culminar, lo que puede convertir el modelado en algo sumamente inconsistente y confuso. El problema se vuelve aún más crítico cuando se trata de grandes sistemas de software en desarrollo, con una cantidad de procesos considerables y complejos.

En el epígrafe 1.3 se abordaron algunas de las metodologías que existen hoy en día para el análisis de riesgos y el modelado de amenazas. Los creadores de estas metodologías comprenden la necesidad de crear herramientas de software que permitan, de manera automatizada y siguiendo la línea de la metodología, facilitar el análisis y modelado de amenazas y así evitar los problemas planteados anteriormente. Ya es difícil encontrar una metodología en el mundo del software que prescinda de una herramienta automatizada para gestionar los procesos que le dan vida. Además porque se convierte en algo poco práctico de usar. A continuación se abordan algunas herramientas desarrolladas, para dar soporte práctico a las metodologías anteriormente tratadas, así como varias de sus características.

1.4.1 TAM v2.1: Herramienta de modelado de Microsoft.

En el año 2006, Microsoft liberó y puso a disposición del público de manera gratuita la versión 2.0 de una herramienta que facilita el análisis y modelado de amenazas. Esta nueva versión, ya difiere sustancialmente de la original, favoreciendo el uso de la nueva metodología y los cambios que ha sufrido. En diciembre del propio año se actualizado a la versión 2.1, incorporando algunas mejoras sobre todo en el asistente que guía el modelado. Es una herramienta con un grado de funcionalidad profundo, con una interfaz elegante, práctica y al mismo tiempo sencilla. Prueba de ello es la facilidad con que se puede hacer un esqueleto básico de nuestro primer modelado siguiendo el asistente, para completarlo con posterioridad (P.F 2006).

Entre las características destacables de esta última versión merece la pena citar las siguientes (Corporation 2006):

- Creación de un modelo básico mediante un asistente.
- Biblioteca por defecto de ataques con una guía descriptiva de contramedidas.
- Generación automática de amenazas y casos de uso.
- Flujo de llamadas, ámbito de ataque, árbol de amenazas (exportables a Microsoft Visio[11]).
- Generación de informes y estadísticas exportables a HTML[12].
- Manuales en video que, aunque son muy básicos, resultan de mucha ayuda por lo menos al dar los primero pasos en el uso de la herramienta.

Otra funcionalidad digna de mención es la posibilidad que nos proporciona la herramienta para hacer uso de diferentes técnicas de medición del riesgo que se pueden incorporar al programa. 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:

Matrices de acceso a datos.

- Casos de uso.
- Diagramas de flujos de datos.
- Superficie de ataque.

---

[11] Herramienta de modelado que forma parte del Paquete Microsoft Office.

[12] Acrónimo en inglés de HyperText Markup Language (lenguaje de marcas hipertextuales), lenguaje de marcación diseñado para estructurar textos y presentarlos en forma de hipertexto, es el formato estándar de las páginas web en Internet.

- Informes.

Desde la misma herramienta 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.
La herramienta ofrecida por Microsoft cumple su función con creces, de forma efectiva, simple y bastante intuitiva. Es sobre todo gratuita y la descarga de la misma se puede realizar desde el sitio oficial de Microsoft (Corporation 2006).

1.4.2 Herramienta de modelado CORAS.

La metodología CORAS está acompañada de dos herramientas. La primera es un editor de diagramas, y la segunda la aplicación cliente-servidor. Ambas están basadas en Java. La aplicación principal permite crear nuevos proyectos de análisis, documentación, editar resultados de los análisis, generar informes así como reutilizar información procedente de otros análisis. El cliente proporciona una interfaz gráfica más o menos eficaz (P.F 2006). Aunque es lamentable que la ejecución de ambas aplicaciones juntas afecten tanto al rendimiento de un equipo medio.
La herramienta, utiliza dos bases de datos a modo de repositorios[13]. El repositorio de evaluación almacena todos los resultados procedentes del análisis, mientras que el repositorio de experiencia, contiene resultados reutilizables procedentes de anteriores análisis como modelos UML, procedimientos, o listas de comprobación.
Además de los modelos UML, la herramienta también permite incluir tablas, diagramas de árboles de eventos, registros de detección de intrusiones y por supuesto descripciones en texto para acompañar el modelado. Se puede integrar con otras herramientas (Group 2004).

1.4.3 Herramienta de modelado de la metodología PTA.

La herramienta de modelado que PTA pone a disposición del público es gratuita para estudiantes, investigadores, desarrolladores de software y consultores de seguridad independientes. Esta aplicación está basada en el motor de Microsoft Access[14], y según la opinión de algunas fuentes consultadas(P.F 2006), la aplicación presenta errores frecuentes durante las pruebas que se han realizado con ella. Resulta curioso también el hecho de que entre los requerimientos se advierta la

[13] Sinónimo de almacén, donde se guardan datos y gestionan sus cambios.

[14] Herramienta de administración de bases de datos, incluida en el paquete de Microsoft Office.

necesidad de contar con permisos de administrador del sistema para ejecutarla, porque si se trata de una herramienta gratuita para ser usada por cualquier persona en el proceso de gestión de la seguridad y no maneja información sensible, solo el modelado de amenazas, pienso que no es necesario tal restricción de acceso, aún así, parece una herramienta interesante.

Como puntos más destacables se pueden citar los siguientes:

- El esquema de la base de datos se puede personalizar para incluir más tipos de entidades como pasos de auditoría, topología, etc.
- Permite la importación automática de datos de entidades y sus parámetros desde fuentes externas como escáneres de vulnerabilidades (Nessus[15]).
- Los distintos métodos de cálculo se pueden adaptar a diferentes situaciones. Por ejemplo los valores financieros que se asignan a los distintos activos se pueden calcular utilizando diferentes fórmulas.
- Utiliza librerías de entidades. Permite elaborar listas de chequeo en conformidad con diferentes estándares de seguridad como ISO 17799 y BS7799, abordados anteriormente.
- Permite la elaboración de un variado número de informes.
- Permite la definición de activos, describirlos y asignarles valores o ver las amenazas que los afectan.
- Algo muy interesante que posee la herramienta es que genera informes mostrando el ranking de amenazas generado por el modelado de acuerdo a su nivel de riesgo. Además de fijar un coste estimado para las contramedidas y llevar un seguimiento de su grado de implementación en cada fase.

PTA es una herramienta que está pensada para facilitarle el trabajo al Analista de Sistema y permitirle utilizar hasta cierto punto su propia metodología, debido al hecho de que se pueden cargar librerías externas de los distintos elementos que componen el modelado, permitiendo un cierto grado de flexibilidad.

En su mayoría, las herramientas de análisis de riesgos y modelado de amenazas tienen varios elementos en común, tales como la representación de árboles de ataques, activos, y por supuesto las

[15] Programa informático utilizado para encontrar vulnerabilidades en la red.

amenazas y riesgos que afectan a los sistemas de software desde su proceso de desarrollo, independientemente de las diferentes perspectivas en que puede enfocarse el modelado de amenazas. El uso de este tipo de herramientas tiende a agilizar el modelado y a disminuir tanto el tiempo de actualización como el esfuerzo que se requiere.

El modelo propuesto en el Capítulo 2 genera un número de artefactos que requieren representación mediante diagramas y tablas, por lo que el uso de una herramienta de modelado que facilite el trabajo de gestión de la seguridad es vital. Sin embargo queda a disposición del personal de seguridad seleccionar aquella que más pueda ayudar a representar los diferentes artefactos del modelo propuesto.

La generalidad de los proveedores de software a nivel mundial utilizan una metodología de Ingeniería de Software (IS) para guiar el proceso de desarrollo a través de diferentes fases y flujos de trabajo de manera tal que exista una organización y sobre todo planificación de todos los aspectos involucrados en el desarrollo tales como recursos, viabilidad, plataformas, personal, entre otros muchos. Esto conlleva a que en el momento de integrar la gestión de la seguridad como elemento paralelo al proceso de IS se tomen en cuenta muchos artefactos que si bien son generados por la metodología de IS seleccionada, constituyen puntos de entrada de mucha utilidad para comprender e identificar en gran medida aquellos entornos donde la seguridad es crítica, así como para la generación de artefactos de seguridad que permitan la gestión de la misma, de manera aceptable, por lo que se aborda este tema a continuación con el objetivo de justificar la selección de RUP como marco de apoyo para elaborar el modelo propuesto.

1.5 RUP como marco de apoyo.

La Ingeniería de Software es un área que aunque en los últimos años ha tenido un auge relativamente acelerado, aún tiene mucho que aprender (A.S.S. Borghello 2001). Algunos de estos elementos están relacionados con la incorporación de características y criterios de seguridad fuertes en el desarrollo de sistemas. La integración de la seguridad al Proceso de desarrollo de software no puede ir separada de la metodología de IS seleccionada, ya que muchos de los artefactos y diagramas generados por esta última constituirán una entrada importante para elaborar artefactos de seguridad. A continuación se analizan algunos aspectos importantes del RUP que permitan comprender su estructura, flujos y artefactos como apoyo para la construcción del modelo propuesto de gestión de la seguridad en el Proceso de desarrollo de software.

La UCI ha adoptado en la generalidad de sus proyectos productivos a RUP como metodología de desarrollo a seguir. RUP es un marco de desarrollo de software iterativo e incremental. No constituyendo simplemente un proceso, sino un marco de trabajo extensible que puede ser adaptado a organizaciones o proyectos específicos (Jacobson, Booch et al. 2001).

1.5.1 Fases definidas por la metodología.

Existen cuatro fases fundamentales y bien definidas sobre las cuales se sustenta, denominadas: Inicio, Elaboración, Construcción y Transición. La figura 2 muestra un diagrama donde se representan estas cuatro fases y el peso que en cada una tiene cada flujo de trabajo (Jacobson, Booch et al. 2001).

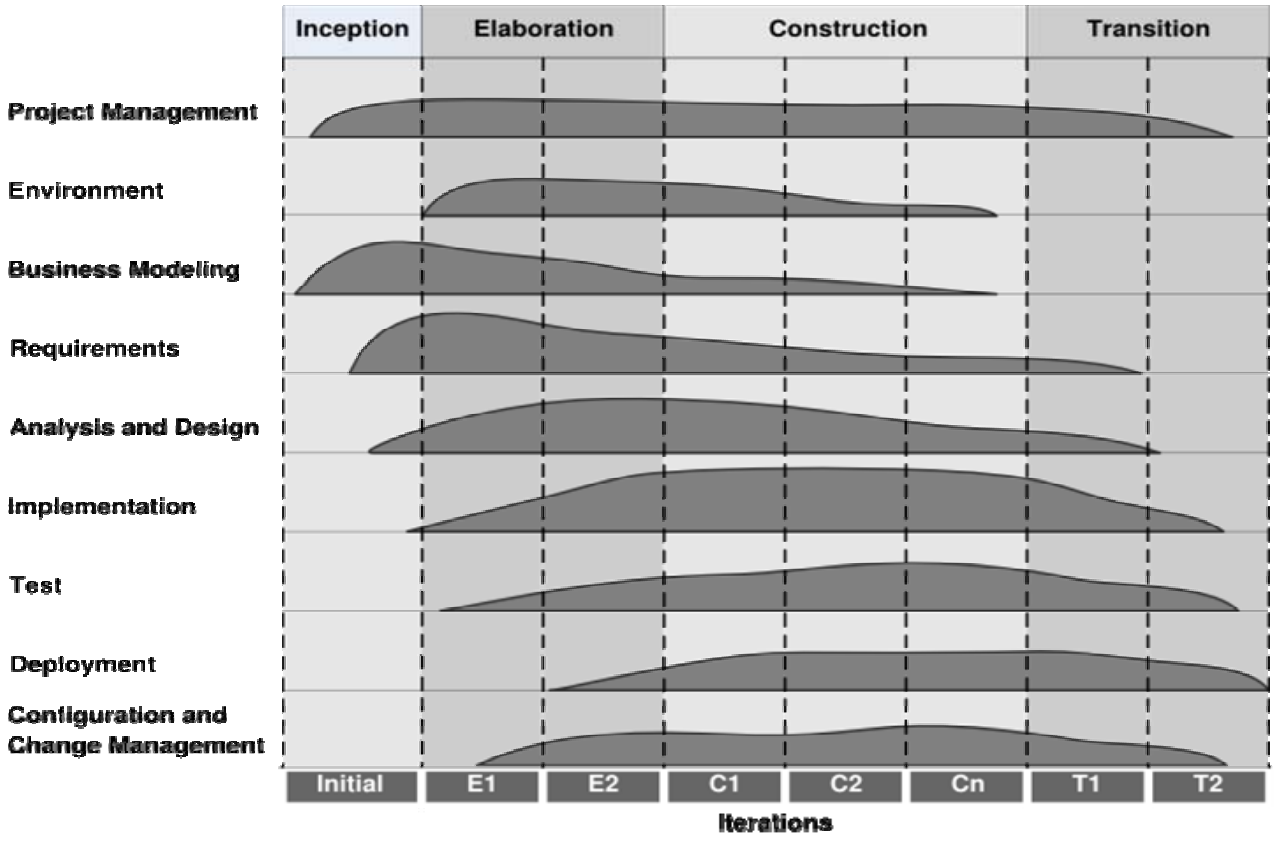


Figura 2. Diagrama ilustrando como el énfasis relativo en las distintas disciplinas cambia a lo largo del proyecto.

Como puede verse, la imagen muestra básicamente el peso que en cada fase tienen los flujos, tanto al inicio de la fase como al final. Estos flujos guiarán el proceso de gestión de la seguridad durante todo el desarrollo y determinarán las actividades y artefactos de seguridad importantes, según los objetivos del flujo en que se esté y los elementos que este aporte.

A continuación se abordarán las cuatro características fundamentales que sustentan RUP:

Iterativo e incremental.

Las fases se dividen en una serie de iteraciones (la de inicio sólo consta de varias iteraciones en proyectos grandes) (Molpeceres 2002). Estas iteraciones ofrecen como resultado un incremento del producto desarrollado que añade o mejora las funcionalidades del sistema en desarrollo. Cada una de estas iteraciones se divide a su vez en una serie de disciplinas que recuerdan a las definidas en el ciclo de vida clásico o en cascada: Análisis de requisitos, Diseño, Implementación y Prueba. Aunque todas las iteraciones suelen incluir trabajo en casi todas las disciplinas, el grado de esfuerzo dentro de cada una de ellas varía a lo largo del proyecto (Molpeceres 2002).

Dirigido por los casos de uso.

Los casos de uso se utilizan para capturar los requisitos funcionales y para definir los contenidos de las iteraciones. La idea es que cada iteración seleccione un conjunto de casos de uso o escenarios y desarrolle todo el camino a través de los distintos flujos de trabajo (Molpeceres 2002).

Centrado en la arquitectura.

Se asume que no existe un modelo único que cubra todos los aspectos del sistema. Por dicho motivo existen múltiples modelos y vistas que definen la arquitectura software de un sistema.

Enfocado en los riesgos.

El Proceso Unificado requiere que el equipo del proyecto se centre en identificar los riesgos críticos en una etapa temprana del ciclo de vida. Los resultados de cada iteración, en especial los de la fase de Elaboración, deben ser seleccionados en un orden que asegure que los riesgos principales son considerados primero.

1.5.2 Artefactos de seguridad.

RUP guía el ciclo de vida del Proceso de desarrollo de software a través de la generación de una serie de diagramas y modelos denominados artefactos (Jacobson, Booch et al. 2000) tales como diagramas UML, Casos de uso, Diagrama de actividades, ente muchos otros, los cuales son creados por los diferentes trabajadores definidos por el Proceso. El Modelo de seguridad a desarrollar se

centrará fundamentalmente en la definición y propuesta de los diferentes diagramas, documentos, y criterios, agrupados bajo el termino de artefactos que desde el punto de vista de la seguridad deben generarse siguiendo básicamente los flujos de trabajo propuestos por RUP, de ahí la importancia de dejar claro este concepto que se aplicará exhaustivamente durante la elaboración del modelo.

1.5.3 Justificación de la selección de la metodología.

La elaboración del Modelo de seguridad propuesto pretende integrar la gestión de la misma durante todo el ciclo del desarrollo de software en la UCI, que abarca desde la captura de requerimientos hasta las pruebas, por lo que hará un uso extensivo de las fases y flujos de trabajo principales por los que transita el desarrollo de software en la Universidad y por tanto es necesario tomar una metodología que apoye la generación de los artefactos de seguridad en cada uno de los flujos de trabajo. RUP genera entregas basadas en artefactos después de cada fase, y es por otra parte una metodología bastante completa que genera gran cantidad de documentación, de ahí su clasificación como “pesada”, aunque define un proceso de desarrollo genérico adaptable a las más diversas características, incluso, debido a su carácter general algunos autores consideran todos los demás procesos de desarrollo como casos particulares de este (Molpeceres 2002). A través de RUP es posible conseguir una mejor estructura y disciplina del proceso de desarrollo (Molpeceres 2002). Estos elementos unidos al hecho de que el modelo que se desarrollará tiene como su centro fundamental de aplicación el Proceso de desarrollo de software de la UCI y esta última principalmente desarrolla proyectos largos en su planificación y ha adoptado a RUP como metodología de desarrollo principal, se considera el uso de RUP como la alternativa más viable para el desarrollo de este modelo, Además de ser una metodología estudiada en profundidad en nuestra universidad. No obstante los artefactos de seguridad que propone la elaboración del modelo serán los mismos independientemente de la metodología de desarrollo que se adopte, solo que los criterios de entrada que aporte la metodología seleccionada tendrán una interpretación diferente. Por ahora los elementos proporcionados permiten tener una idea de RUP y su funcionamiento, sin embargo no se profundiza más, ya que en el Capítulo 2 se abordarán, por cada flujo, aspectos más específicos.

1.6 Algunos resultados obtenidos por Microsoft en el desarrollo de software.

Las investigaciones realizadas, con el objetivo de encontrar resultados concretos que demuestren la efectividad de la aplicación de modelos de seguridad desde el inicio del proceso por parte de proveedores de software reconocidos en el mundo que comprenden la necesidad de crear productos de software seguros, han arrojado muestras claras de cuanto puede avanzarse en garantizar software seguros y confiables.

Ciertamente Microsoft es una de las compañías que más ha avanzado en este tema en los últimos años, desarrollando sus propias metodologías y herramientas integradas al Proceso de desarrollo de software para garantizar una efectiva modelación de riesgos desde fases tempranas del desarrollo (Lipner and Howard 2005). Windows Server 2003 fue el primer lanzamiento de sistema operativo de Microsoft en el que se implementaron partes importantes del Ciclo de Vida de Desarrollo Seguro (SDL) desarrollado por ellos (Lipner and Howard 2005). El primer gráfico del Anexo 3 muestra que el número de boletines de seguridad publicados en un período de un año tras el lanzamiento de Windows Server 2003 fue de 24 boletines no siendo así para Windows 2000, antes de aplicar el SDL cuyo número de boletines fue de 62. Ambos son los sistemas operativos de servidor de Microsoft más recientes (Cuando se lanzó Windows 2000, Microsoft no disponía de un sistema formal de calificación de la gravedad de los boletines de seguridad. Microsoft ha evaluado todos los boletines de seguridad que se aplican a Windows 2000 respecto al sistema de calificación de la gravedad actual.) Windows Server 2003 se desarrolló con la mayoría de los procesos del SDL (aunque no con todos), mientras que Windows 2000 no los incluyó (Rodríguez and Grimaldi 2003).

Otros lanzamientos de software de Microsoft también han aplicado elementos del SDL. Los equipos de producto de SQL Server y Exchange Server también realizaron una campaña de seguridad (incluido el modelado de amenazas, las revisiones del código y las pruebas de seguridad) antes de lanzar un Paquete de Servicio [16], un lanzamiento de software que acumula soluciones para vulnerabilidades de seguridad y de otro tipo. Los resultados de la campaña de seguridad de SQL Server se incorporaron en SQL Server 2000 Service Pack 3 y los resultados de la campaña de

[16] Conjunto de actualizaciones de software que se instalan a los sistemas ya sea para mitigar vulnerabilidades existentes o simplemente para adicionar nuevas funcionalidades. Service Pack por sus siglas en inglés.

seguridad de Exchange Server se incorporaron en Exchange 2000 Server Service Pack 3. Los gráficos 2 y 3 del Anexo 3 muestran el número de boletines de seguridad publicados en los mismos períodos antes y después de los respectivos Service Pack (24 meses para SQL Server 2000 y 18 meses para Exchange 2000 Server) fueron de 16 boletines antes del SDL y solamente 3 luego de su establecimiento.

Una prueba importante de que el camino que ha tomado Microsoft en cuanto a la integración de la seguridad en el ciclo de desarrollo de sus productos parece no ser equivocado se desprende de un reciente artículo de David Litchfield (NGSSoftware) (Litchfield 2006), reconocido consultor y descubridor de un elevado número de vulnerabilidades. En este documento, el autor efectúa una comparativa entre la seguridad de dos conocidos gestores de bases de datos: Oracle y SQL Server. En él, se puede apreciar tanto la nefasta respuesta por parte de Oracle respecto a la política de corrección de vulnerabilidades, como el elevadísimo número de estas que han sido descubiertas en los últimos 5 años en comparación con las que se han detectado en las distintas versiones de SQL Server.

Otro resultado obtenido por Microsoft es el Internet Information Server de Windows Server 2003 (IIS 6). En un periodo de más de tres años que han transcurrido desde su lanzamiento, Microsoft ha publicado un boletín de seguridad que afecta al servidor Web, consistente en un componente (WebDAV) que no se instalaba de manera predeterminada.

Aunque las muestras de las vulnerabilidades de seguridad son reducidas y los períodos de tiempo son limitados, estos resultados demuestran que el SDL es eficaz. Microsoft actualmente continua supervisando los índices de vulnerabilidades de Windows Server 2003 y los Service Packs de Exchange Server y SQL Server para ver cómo evoluciona esta tendencia. También analiza más software a medida que se lancen nuevas versiones tras una implementación completa del SDL para determinar si el número y el nivel de gravedad de las vulnerabilidades de seguridad siguen descendiendo.

Conclusiones del capítulo.

Realizar la gestión de la seguridad de manera efectiva en el Proceso de desarrollo de software no es un asunto fácil, antes que todo debe comprenderse una serie de conceptos importantes relacionados con esta y su gestión, amenazas, ataques, etc. Esto se debe a que la seguridad de software y de sistemas informáticos de manera general es manejada muchas veces como un concepto borroso y

con cierto grado de incertidumbre. Durante el capítulo se han definido algunos de estos conceptos sin embargo muchos otros serán tratados con más detenimiento en el próximo.

Garantizar aspectos esenciales de seguridad tales como: Integridad, Disponibilidad, Privacidad, Control y Autenticidad de la información manipulada por el software es una tarea que requiere gestión y seguimiento desde el inicio mismo del desarrollo. Requiere educar al personal involucrado, en un ambiente de desarrollo seguro, crearle una cultura de escribir código de calidad y más que eso inculcarle la necesidad de ver la seguridad como parte imprescindible de su trabajo y no como una imposición.

Gestionar la seguridad en el proceso de desarrollo está muy ligado a la metodología de Ingeniería de Software adoptada ya que los flujos definidos por esta última determinarán en gran medida los pasos a seguir por el ES y los artefactos a generar en cada momento. Es por eso que para el modelo que se propone desarrollar en el Capítulo 2 se ha concluido seleccionar al RUP como metodología de desarrollo fundamental de apoyo para la realización del modelo propuesto a partir del uso de sus flujos de trabajo para generar los artefactos de seguridad imprescindibles durante la planificación, de manera tal que el avance sea paralelo y se puedan aprovechar importantes artefactos generados por RUP para la gestión de la seguridad, tales como los diagramas de secuencias, de actividades, interacción, etc.

Las metodologías de análisis y evaluación de riesgos que han surgido si bien no resuelven todos los problemas de seguridad en el software, sí constituyen un elemento clave para que los proveedores desarrollen sistemas más confiables que garanticen los niveles de seguridad básicos de la información sensible y del sistema de manera general. A pesar de que el análisis realizado en el epígrafe 1.3 permite llegar a la conclusión de que generalmente estas metodologías son implementadas por empresas específicas para gestionar su seguridad, estas constituyen un punto de referencia importante en el momento de elaborar nuestro propio modelo, sobre todo en aquellos casos en que los resultados de su aplicación han sido alentadores, como se analizó en el epígrafe 1.6.

La construcción del Modelo de seguridad propuesto usará como procedimientos de clasificación de amenazas y evaluación de riesgos a los métodos STRIDE y DREAD propuestos por la metodología TAM desarrollada por Microsoft Corporation y analizados anteriormente (Ver Anexos 1 y 2).

2
CAPÍTULO.

2. DESARROLLO DEL MODELO PROPUESTO.

El capítulo tiene como objetivo fundamental, elaborar el Modelo de seguridad propuesto que abarca este trabajo. La implementación del modelo se apoya en los flujos de trabajo fundamentales definidos por RUP, que ha sido establecido como proceso de trabajo a seguir en la generalidad de los proyectos de la Universidad; garantizando que la gestión de la seguridad forme parte inherente del Proceso de desarrollo de software hasta finalizar las pruebas del sistema.

El capítulo se estructurará en cuatro flujos de trabajo principales (Flujo de trabajo de Planificación inicial y Captura de requisitos de seguridad, Flujo de trabajo de Análisis y Diseño seguro, Flujo de trabajo de implementación y Codificación segura, Flujo de trabajo de Pruebas de seguridad). En cada flujo de trabajo se definirán y detallarán los artefactos de seguridad que deben generarse a partir de las actividades de gestión en que participan los especialistas en seguridad durante el avance de cada flujo. También se especifican los trabajadores de RUP y los artefactos generados por estos que constituyen entradas de información valiosas para garantizar que el proceso de gestión de la seguridad sea controlado, continuo y paralelo a la Ingeniería de Software. Cada artefacto de seguridad se representa en el documento de gestión a través de diagramas y tablas según corresponda, con el fin de alcanzar resultados más organizados que permitan una comprensión más sencilla. Los artefactos obtenidos en materia de seguridad en los flujos anteriores, juegan un papel importante ya que constituyen las entradas a partir de los cuales se generan otros a medida que se avanza en la gestión, o para actualizar e incorporar nuevos elementos a los ya existentes. La verdadera ventaja del desarrollo del modelo reside en el hecho de que al aplicarse como un proceso desde el inicio del proyecto hasta las pruebas, supone un enfoque diferente al tradicional análisis de riesgos y la posterior aplicación de medidas que contribuyan a mejorar la seguridad.

2.1 Flujo de trabajo de Planificación inicial y Captura de requisitos de seguridad.

La necesidad de considerar la seguridad "de abajo a arriba" es uno de los principios fundamentales del desarrollo de sistemas seguros. El inicio de un proyecto de desarrollo de software es un punto complejo y a la vez determinante para el paso hacia los flujos siguientes. Las decisiones y actividades deficientes y no controladas de esta etapa continuarán deficientes en adelante, incluso es probable que empeoren el resto del proceso (Devanbu and Stubblebine 2000). Esta constituye una de las múltiples razones que conllevan a definir que, el planeamiento inicial, a pesar de ser el primer acercamiento a lo que más tarde se transformará en todo un proceso que llevará meses de esfuerzos, dedicación y sobre todo responsabilidad por parte de un número de especialistas para tomar decisiones determinantes, constituye una etapa clave que marcará el buen ritmo, o los problemas posteriores, del proceso de desarrollo. Algunas razones que sustentan tal afirmación estarán dadas por el hecho de que los desarrolladores no serán los usuarios finales del sistema y no conocen en profundidad el funcionamiento de los procesos de negocios del cliente (Booch, Jacobson et al. 2001). Estos a su vez en la mayoría de los casos solo tienen la idea de que necesitan un sistema pero son incapaces de definir las características que debe tener, por lo que el trabajo del Analista de Sistema en la captura de requerimientos se debe desarrollar generalmente en un ambiente hostil y confuso que requiere pericia, claridad y experiencia. A continuación se tratará más a fondo y en términos de seguridad, los aspectos fundamentales que deben reflejarse en esta etapa.

La siguiente figura muestra las actividades, trabajadores y artefactos de seguridad durante el Flujo de trabajo de Planificación inicial y Captura de requisitos de seguridad.

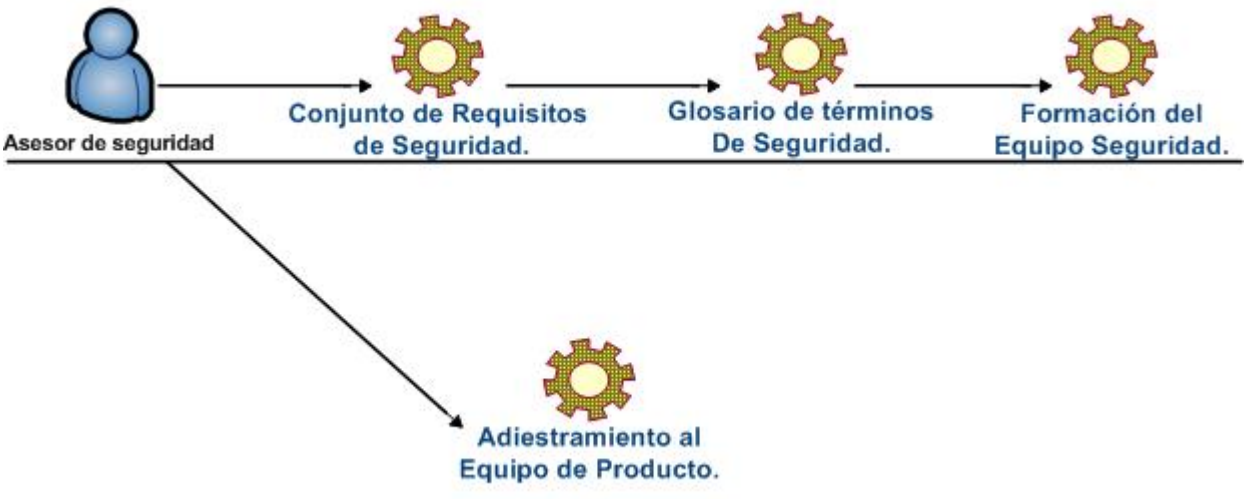


Figura 3. Trabajadores, artefactos y actividades fundamentales en el Planificación inicial y Captura de requisitos de seguridad.

2.1.1 Actividad: Solicitud del Asesor de Seguridad.

El planeamiento inicial, es el momento adecuado para integrar la gestión de la seguridad como parte del proceso de desarrollo (Shreyas 2001). Esto se debe a múltiples razones y la primera de ellas es que este flujo constituye un momento crítico en cuanto a la captura de requisitos que más tarde se convierten en funcionalidades materializadas del nuevo sistema. Lo cual implica que si se quieren tener resultados realmente concretos en cuanto a seguridad al final del proceso, entonces la identificación de requerimientos como elemento fundamental de este flujo debe incluir aspectos relevantes de seguridad que permitan tener un control de todos los puntos críticos y comprometedores del futuro sistema. En un primer momento el Equipo de Producto (EP)[17], debe solicitar a su organización la asignación de un especialista experimentado en temas de seguridad como asesor principal, el cual en conjunto con el EP realizará las primeras planificaciones y la captura de requisitos de seguridad. El asesor realiza los cálculos pertinentes a partir de los aspectos encontrados y planificados y determinará en un primer momento el número de miembros que

[17] En adelante se tratará como Equipo de Producto o EP al personal involucrado de alguna forma con el desarrollo del software, esto no incluye al personal encargado de la gestión de la seguridad, ya que se establecerá otro término con el fin de separar las tareas de cada uno.

conformarán el Equipo de Seguridad[18]. El asesor actuará, en fin, como punto de contacto, recurso y guía a través de los procedimientos de planeamiento. En conjunto, el EP y el asesor, revisan los planes, aportando recomendaciones y asegurándose de planificar los recursos necesarios de acuerdo con el programa de fechas del EP. Se advierten sobre los puntos básicos de seguridad y los criterios de salida que serán necesarios en función del tamaño, la complejidad y los riesgos del proyecto. El asesor rendirá cuentas a la dirección del EP acerca del correcto avance de la gestión de la seguridad del proyecto para evitar resultados inesperados e imprevistos en etapas posteriores relacionados con la seguridad.

2.1.1.1 El documento de gestión de seguridad.

El Asesor de Seguridad necesita llevar como parte esencial de su trabajo un documento donde se registren todos los aspectos y artefactos que durante su gestión se van generando. A medida que el proceso de desarrollo avanza, este documento se nutre y actualiza con nuevos elementos. El documento forma parte de la documentación que se genera durante el ciclo de vida del desarrollo y debe ser actualizado constantemente. A continuación se indican algunos aspectos del contenido del documento.

Estructura del Documento de Gestión de Seguridad (DGS).

1. El formato del documento debe ajustarse a la plantilla establecida, por la institución o proyecto de desarrollo de software donde labora el ES.
2. Nombre del especialista líder del Equipo de Seguridad.
3. Fecha de inicio del proceso de gestión de la seguridad.
4. Organización por flujos:
    - Flujo de trabajo de Planificación inicial y Captura de requisitos.
    - Flujo de trabajo del Análisis y Diseño seguro.
    - Flujo de trabajo de Implementación y Codificación segura.
    - Flujo de trabajo de Pruebas de seguridad.
5. La estructura de cada flujo debe contener al menos:

[18] En adelante se tratará indistintamente como Equipo de Seguridad o ES al personal encargado de la gestión de la seguridad en el proceso de desarrollo de software.

- Listado de artefactos de seguridad que se generan durante el flujo.
- Trabajadores de RUP y miembros del ES que participan en la realización del artefacto de seguridad.
- Artefactos de RUP y artefactos de seguridad ya existentes que se usan en la generación de un artefacto de seguridad determinado.
- Fecha de generación de cada artefacto de seguridad.

6. Glosario de Términos de Seguridad.

2.1.2 Artefacto: Conjunto de Requisitos de Seguridad.

La captura de requisitos es el proceso de averiguar y descubrir, normalmente en circunstancias difíciles, lo que debe hacer el sistema (Booch, Jacobson et al. 2001). Teniendo en cuenta que muchos proyectos de desarrollo de software generan la versión a partir de la anterior, la captura de requisitos y el planeamiento inicial ofrece una oportunidad estupenda para crear software más confiable. Los requisitos suelen ser asignatura pendiente en muchos proyectos de software. Obtenerlos, analizarlos y especificarlos sin ambigüedades ni omisiones, de forma que resulten verificables y cuantificablemente medibles, no es fácil. Y si no resulta así con los requisitos funcionales, cuando se trata de requerimientos de seguridad, el terreno se vuelve aún más difícil (Howard and LeBlanc 2002).

2.1.2.1 Trabajadores que participan.

El Analista de Sistema es el rol establecido por RUP como responsable de la captura de requisitos y su representación mediante Casos de uso en un Modelo de Negocio y posteriormente en un Modelo de Sistema, el cual refleja las relaciones entre los casos de uso y los actores, delimitando el sistema. El Asesor de Seguridad designado por el EP labora en conjunto, y de manera muy estrecha, con el Analista de Sistema como pieza fundamental de este flujo. Teniendo en cuenta que para hacer su trabajo de identificación de requisitos de seguridad requiere, como entrada, algunos artefactos generados por este último, haciéndose necesario especificar aquellos que tienen un peso importante en términos de seguridad.

2.1.2.2 Artefactos de entrada de RUP.

Ningún proyecto de software es igual otro. Existen diferencias en el tipo de sistema, la tecnología, incluso el cliente y sus negocios y la estructura de desarrollo establecida. Muchas veces los puntos de partida en la captura de requisitos tienen diferentes orígenes y enfoques (Booch, Jacobson et al. 2001). La captura de requerimientos es una actividad que puede tener diferentes puntos de partida. En algunas ocasiones puede comenzarse realizando un Modelo de Negocio o continuar con uno ya iniciado por otra empresa. En otros casos puede tenerse como entrada un Modelo de objetos sencillo, como un Modelo de Dominio**,** debido a que los procesos de negocios no tienen una definición clara (Booch, Jacobson et al. 2001). O en el mejor de los casos una especificación de requisitos detallada, ya realizada con anterioridad por otros, a partir de la cual se negocian los cambios. Sin embargo muchas veces hay que enfrentarse a la dura realidad de que no existe nada para comenzar, solo una idea del cliente.

Tanto el Analista de Sistema como el Asesor de Seguridad, a partir de las diferentes entradas anteriores, obtienen una comprensión global del entorno sobre el cual se mueve la empresa y sus gestiones. El Modelo de Casos de uso, como parte del Modelo de Sistema es el artefacto generado por el Analista de Sistema que más elementos aportará al Asesor de Seguridad, esto se detallará más adelante en este mismo flujo. Sin embargo, si bien el analista mediante la generación de estos artefactos, obtiene elementos claves que luego le ayudan a identificar clases en el análisis y diseño del sistema, así como las relaciones que estos pueden tener posteriormente; los principales beneficios para el Asesor de Seguridad están dados en términos de:

1. Identificación de futuros usuarios y sus roles: El Modelo del Dominio captura los conceptos más importantes en el contexto del sistema. Estos conceptos representan ”cosas” que se manipulan en el negocio y objetos del mundo real, entre otros (Booch, Jacobson et al. 2001). Muchos de estos conceptos contenidos en el Modelo de Objetos, o en el Modelo del Negocio, como son los actores y trabajadores representarán los usuarios que más tarde harán uso del sistema y sobre los cuales se identifican requisitos de seguridad. Estas son informaciones valiosas, que permitirán durante el modelado de amenazas identificar y establecer los roles de los usuarios para hacer uso del sistema, así como las responsabilidades y derechos que estos tendrán sobre determinados recursos. La Matriz de Control de Acceso es el artefacto de seguridad que se nutre de estos resultados.

2. Comprensión del contexto del sistema: El Modelado del Dominio y el Modelado del Negocio permitirán identificar y delimitar los procesos y el contexto en que se enmarcará el futuro sistema. La información que aportan estos artefactos permitirán, durante el análisis y diseño del sistema, la creación del Diagrama de Contexto, como artefacto necesario para elaborar el Modelo de Amenazas. Muchos requisitos de seguridad, sobre todo los relacionados con el entorno en que se implantará el sistema, aparecerán a partir del análisis que se realiza de estos artefactos.
3. Identificación de recursos importantes que requiere ser protegida de acceso no autorizado: Los recursos que representan un valor crítico para el correcto funcionamiento del sistema (activos críticos) tienen una fuente de entrada importante en las entidades del negocio identificadas a partir de la Especificación de casos de usos, Modelo de objetos de negocio, Diagramas de Actividades. Sobre las entidades identificadas se definen muchos requisitos de seguridad, relacionados con su confidencialidad y protección.

Existen muchos más elementos que a simple vista resulta difícil de identificar, sin embargo un buen análisis de estos artefactos ayudarán a encontrarlos.

2.1.2.3 Actividad: Capturar requerimientos de seguridad.

El principal artefacto de seguridad que se genera en este flujo por parte del Asesor de Seguridad, es el *Conjunto de Requisitos de Seguridad (CRS)*. La forma tradicional de captura de requisitos de seguridad que persiste en la generalidad de los proyectos de desarrollo de software, consiste en establecer como una plantilla tres elementos de gestión de seguridad:

- El sistema debe garantizar la **Integridad** de los recursos y activos sensibles.
- El sistema debe garantizar la **Confidencialidad** de la información y los datos.
- El sistema debe garantizar la **Disponibilidad** de la información y del sistema.

Es válido aclarar que esto no está mal cuando se plantea, sin embargo la pregunta es: ¿Realmente los sistemas cumplen, al finalizar su desarrollo y establecerse en un entorno hostil, con estos requisitos? Lamentablemente la respuesta es que no cumplen con estos requisitos, de lo contrario no existieran tantos productos de software fracasados en el mercado. Aunque muchos requisitos de características de seguridad aparecerán a partir del Modelo de Amenazas, es probable que sean los requisitos de los usuarios los que dictaminen la inclusión de características de seguridad como

respuesta a las demandas de los clientes (Lipner and Howard 2005). En esta etapa es determinante el trabajo que deben realizar en conjunto el Asesor de Seguridad y el Analista de Sistema.

A continuación se exponen una serie de pasos que si bien no son una plantilla que debe seguirse de manera esquemática, si constituyen una guía importante que se propone para dar un orden a la actividad de encontrar elementos de seguridad claves en el proceso de captura de requisitos. Estos pasos muestran la actividad a realizar y los participantes en ella:

1. Antes que el Analista de Sistema realice su primera entrevista con los clientes para capturar requisitos, es necesario que tenga un encuentro con el Asesor de Seguridad designado en ese momento, el cual, además de tratar algunos temas importantes de seguridad, elabora una serie de preguntas y cuestiones generales y específicas que el Analista de Sistema debe preguntar o tener presente durante la entrevista con los clientes, de manera tal que aporten elementos de seguridad fuertes y permitan, ya desde el inicio mismo, tener algunos elementos claves y delimitados dentro del contexto del negocio. Hay muchos elementos importantes para el Analista de Sistema que también lo son para el asesor, sin embargo es válido que ambos comenten sobre estos. Algunas de las preocupaciones y preguntas pueden formularse en términos mucho más acotados que realmente lleven a resultados valiosos y no a planteamientos tan globales que al final dejan mucho que desear:
    - ¿Quiénes son las entidades involucradas en el negocio, en términos de personas, dependencias y cuáles son sus roles?
    - ¿Cuál es el ambiente en que se mueve el negocio, para conocer el entorno (Internet, desconectado, etc.) a que estará expuesto el sistema una vez liberado?
    - ¿Cuáles entidades y procesos del negocio son críticas en cuanto a su manipulación y funcionamiento?
    - ¿La empresa necesita realizar auditorías frecuentemente y cuáles son esas áreas?
    - ¿Qué datos o activo crítico podría no estar disponible y por cuánto tiempo?

Infinitas preguntas pueden surgir. Sin embargo la experiencia anterior que posee el Asesor de Seguridad en el desarrollo de otros sistemas es un elemento clave que ayuda a identificar requisitos importantes que a simple vista son difíciles de ver.

2. Un segundo encuentro, luego de la captura de requerimientos que más tarde se representan en Casos de Uso, permite, a partir de la visión general del negocio que adquiere el Analista de

Sistema, como resultado de las entrevistas, identificar e informar al asesor, sobre algunos **objetivos claves**, además de los requerimientos de seguridad. Cuando se dice objetivos claves, no se refiere a aspectos tan generales como los que se plantean al inicio del epígrafe, sino garantizar las características de seguridad en aquellos lugares donde realmente es crítico el tema de la seguridad.

3. A partir de los diferentes artefactos de los que se vale el Analista de Sistema para estructurar los requisitos y darle forma, ya sean Modelo de Dominio y Modelo del Negocio, entre otros, se inicia un proceso determinante, donde el asesor en conjunto con el Analista de Sistema, obtiene de este los actores que interactúan con el sistema y los documenta, pudiendo delimitar el entorno. Este detalle es un punto muy importante, ya que estas interacciones de los actores con el sistema permitirán definir roles futuros de los usuarios y controlar el acceso a determinadas funcionalidades en dependencia del rol.

Una decisión importante puede ser agrupar los requisitos de seguridad en dependencia del aspecto de seguridad sobre el que influyen directamente, así como asociarlos a casos de uso específicos. A Continuación se presenta una propuesta de cómo deben quedar agrupados y documentados los requisitos de seguridad en el documento de gestión.

GRUPO	REQUISITO	CU ASOCIADO	ACTOR
Confidencialidad	•	•	•
Integridad	•	•	•
Disponibilidad	•	•	•
Autenticidad	•	•	•
Otros	•	•	•

Tabla 1. Representación y agrupamiento de los requisitos de seguridad identificados.

GRUPO: Representa las características de seguridad que se desea preservar con el requisito en cuestión. Todos los requerimientos que respondan a esta característica estarán en este grupo.

REQUISITO: Representa cada uno de los requisitos de seguridad capturados hasta este momento, como resultado del trabajo entre el asesor y el Analista de Sistema.

CU ASOCIADO: El caso de uso sobre el que incide directamente un determinado requisito debe ser especificado en esta columna. Esto permite conocer aquellos flujos del proceso de manera acotada donde la seguridad es crítica y cuál es la(s) característica(s) a considerar con mayor fuerza.

ACTOR: Se especifica el actor que inicia el flujo de sucesos dentro del caso de uso.

Existen una serie de requisitos que no pueden asociarse a Casos de uso específicos, debido a su impacto en varios Casos de uso, en ese caso solo se registran los demás datos.

Es importante recordar que muchos requisitos de seguridad aparecerán en otras etapas sobre todo durante el flujo de trabajo de análisis y diseño seguro, cuando se definan las tecnologías y plataformas de desarrollo. Justo en ese flujo aparecerán la mayoría de los requisitos de seguridad para el proceso de escritura del código fuente.

2.1.3 Artefacto: Glosario de Términos de Seguridad.

Todo proceso de desarrollo debe contar con un glosario de términos, el ES tendrá uno que le permita comprender determinados conceptos de seguridad que como parte del desarrollo son imprescindibles y deben tenerse claros. El glosario irá nutriéndose a medida que aparezcan nuevos elementos. Los términos reflejados deben poseer una definición clara, que evite confusiones en su comprensión, y su representación debe estar estructurada preferiblemente en orden alfabético para una mejor localización de los términos si es demasiado grande el glosario. El asesor es el responsable, en el momento de su designación y a medida que avanza el proceso, de establecer la estructura e incluirlo al final del documento de gestión.

2.1.4 Actividad: Formación del Equipo de Seguridad.

Una vez que el Asesor de Seguridad posee elementos suficientes, sobre todo el CRS en una primera aproximación, procede a evaluar la formación inmediatamente del ES del cual posiblemente será su máximo representante, en cuyo caso cambia su rol a Líder del ES. La cantidad de personas que debe formar parte del ES no tiene un límite exacto. Sin embargo debe estar en correspondencia con el alcance del proyecto y la cantidad de áreas que realmente necesitan ser gestionadas en cuanto a seguridad, así como la complejidad que implica cada acción concreta. Muchas veces la organización proveedora de software posee un número significativo de personas capacitadas en temas de seguridad a partir de desarrollos y experiencias adquiridas en otros proyectos, y esta es la cantera a partir de la cual se nutre el ES para el proyecto. Sin embargo la selección de los miembros del equipo

no debe ser un acto al azar, deben realizarse entrevistas a personas que tengan una experiencia en temas de seguridad y sobre todo que realmente estén en la mayor disposición de apoyar el proceso para el que se requieren. Puede suceder, y de hecho ocurre con frecuencia, que la empresa proveedora del software al no tener experiencias en temas de seguridad, prefiere contratar personal externo a la organización. Esta idea es válida y puede resultar la mejor opción sobre todo cuando realmente el producto de software es crítico en cuanto a la garantía de sus activos. Sin embargo hay que tener claro que contratar personal externo implica costos adicionales. Y muchas veces no existe el presupuesto necesario para ejecutar esta acción en cada proyecto de software que se implemente, por lo que es recomendable capacitar y formar un equipo propio de la empresa, dentro del cual deben existir responsabilidades específicas para cada uno de sus miembros, cuya intervención tiene mayor peso en determinados flujos. Algunos de estas responsabilidades y habilidades son:

- **Arquitectura de seguridad:** Habilidades para el trabajo con arquitectura de redes, análisis de riesgos, criptografía, división de la aplicación en capas seguras, protocolos.
- **Diseñadores:** Diseñar casos de prueba de seguridad, análisis de riesgos, protocolos.
- **Desarrolladores:** Codificación segura, herramientas automatizadas de análisis de código.
- **Probadores:** Pruebas de seguridad en condiciones reales, pruebas de penetración, errores comunes, herramientas de prueba.

Otras responsabilidades pueden definirse en dependencia de la magnitud del proyecto y las áreas de seguridad implicadas

2.1.5 Actividad: Primer adiestramiento al Equipo de Producto en temas de seguridad.

Ya con el CRS y la conformación del ES, una de las actividades básicas que realiza en este punto el asesor en conjunto con el ES, es planificar e impartir un curso de capacitación, en temas generales de seguridad, al Equipo de Producto. Además de informar sobre las futuras acciones que se desarrollarán durante el proceso de gestión de la seguridad en los flujos siguientes y algunas medidas importantes que deben tomarse. El adiestramiento debe involucrar a todo el personal, de manera absoluta. El hecho de implicar a los diferentes entes (desarrolladores, gerentes, administradores de sistemas, probadores) en cualquier proceso que ataña al término seguridad ya es de por sí todo un logro, muy positivo.

2.1.6 Resumen del Flujo de trabajo de Planificación inicial y Captura de requisitos de seguridad.

El Flujo de trabajo de Planificación inicial y Captura de requisitos es el momento idóneo para integrar la gestión de la seguridad al Proceso de desarrollo de software. El Analista de Sistema es el principal trabajador de RUP que, en esta etapa, trabaja en conjunto con el asesor para encontrar requerimientos de seguridad provenientes de los resultados obtenidos en las entrevistas con los clientes y de la visión que se alcanza acerca del entorno que rodean los procesos de negocios que desean automatizarse. En algunos casos a partir del Modelo de Negocio y del Dominio, y en otros, a partir del Modelo de Casos de Uso, como artefacto principal generado por RUP durante la Captura de requisitos. Se han especificado las razones por las que es importante la presencia de un asesor con experiencia en temas de seguridad en esta primera parte, así como las actividades en las que este participa y los artefactos principales que genera, entre los cuales se definieron tanto el Conjunto de Requisitos de Seguridad, como el Glosario de Términos de Seguridad. La formación del Equipo de Seguridad es otro resultado importante de este flujo, ya que constituye junto al Conjunto de Requisitos de Seguridad un elemento de entrada importante para el siguiente flujo de análisis; ya que la labor de gestión de la seguridad se intensifica a partir de entonces, y la planificación en adelante se centrará en garantizar los requisitos hasta el momento encontrados y los nuevos que surgen en futuras iteraciones previstas en la metodología RUP. Para lograr esto se asignan responsabilidades a los miembros del ES para diferentes actividades, por las cuales responde e informa al Líder del ES. En el siguiente flujo de trabajo se estarán abordando cada una de estas actividades específicas.

2.2 Flujo de trabajo de Análisis y Diseño seguro.

El Flujo de trabajo de Análisis y Diseño seguro desde la perspectiva de la seguridad, recibe como entrada del Flujo de Planificación inicial y Captura de requisitos de seguridad, el artefacto Conjunto de Requisitos Seguridad y el ES ya conformado. Además del Modelo de Casos de uso, como artefacto fundamental de RUP generado en dicho flujo por el trabajador Analista de Sistema. Debe recordarse que el Modelo de Casos de Uso, está compuesto por una descripción general, un conjunto de diagramas, y una descripción detallada de cada caso de uso, los cuales reflejan la interacción de los actores con los procesos de negocios (Booch, Jacobson et al. 2001). Estos elementos de entrada son utilizados por RUP durante el actual flujo para analizar, estructurar y refinar los requisitos con mayor profundidad, de manera tal que se comience a razonar sobre los aspectos internos del sistema. Se

define cómo llevar a cabo la funcionalidad dentro del sistema, a partir del uso de una plataforma de programación y un lenguaje específico y funge como una antesala inmediata del Flujo de trabajo de Implementación y Codificación segura (Booch, Jacobson et al. 2001). Las principales actividades en que interviene el ES, durante la primera parte de este flujo correspondiente al análisis del sistema, están relacionadas con la identificación de paquetes de análisis y paquetes de servicios reutilizables y seguros, por lo que se tratará con más énfasis la parte correspondiente al diseño ya que estos paquetes del análisis tienen influencia directa en los subsistemas de diseño y en los subsistemas de servicio, respectivamente.

Sin embargo, ¿Cuáles son los artefactos de RUP que se generan en este flujo y que constituyen elementos imprescindibles en los cuales se apoyan el ES para darle seguimiento al proceso de gestión de la seguridad? ¿Cuáles son los trabajadores de RUP que desempeñan un papel determinante en la labor conjunta con el ES? ¿Cuáles son los artefactos de seguridad que se generan en este flujo? Estas y otras preguntas tendrán respuesta durante el transcurso del flujo.

2.2.1 Objetivos del análisis y diseño para la gestión de la seguridad.

El Conjunto de Requisitos de Seguridad tiene un peso importante durante el análisis y diseño del sistema ya que todas las actividades en que labora el ES a partir de entonces, tendrán como premisa fundamental garantizar la mejor manera de dar cumplimiento a dicho artefacto. El flujo de trabajo actual es un punto complejo en la fabricación de un producto de software, constituye la etapa en que se toman decisiones determinantes para entrar en el Flujo de trabajo de Implementación y Codificación segura ya que impone una estructura del sistema que debe conservarse lo más fielmente posible a medida que se le da forma (Booch, Jacobson et al. 2001). Los propósitos del diseño, en términos de seguridad, de manera concreta son:

- Adquirir una comprensión en profundidad de los aspectos relacionados con los requisitos de seguridad y restricciones de los lenguajes de programación, componentes reutilizables, sistemas operativos, tecnologías de distribución, seguridad del código para la implementación, entre otros elementos.
- Definir la arquitectura de seguridad del software y las directrices de diseño: Definir la estructura global del software desde el punto de vista de la seguridad e identificar los componentes cuyo correcto funcionamiento es esencial para la seguridad. La identificación de técnicas de diseño, como el uso de capas, la aplicación de privilegios mínimos y la

minimización de la superficie de ataque, que se aplican al software de manera global. Los detalles específicos de cada uno de los elementos de la arquitectura se indican en las especificaciones de diseño individuales, pero la arquitectura de seguridad corresponde a una perspectiva global sobre el diseño de seguridad.

- Identificar y documentar los elementos de la superficie de ataques del software: Teniendo en cuenta que este no logrará una seguridad perfecta, es importante que únicamente se expongan de manera predeterminada las características que utilicen la mayoría de los usuarios y que dichas características se instalen con el mínimo nivel de privilegios posible. La medición de los elementos de la superficie de ataque ofrece al Equipo de Producto un indicador continuo de la seguridad predeterminada y les permite detectar las instancias en las que el software es más susceptible de recibir ataques. Aunque algunas partes con mayor superficie de ataque pueden estar justificadas por una mayor facilidad de uso o unas mejores funciones del producto, es importante detectar y considerar cada una de estas instancias durante el diseño y la implementación para lanzar el software con la configuración predeterminada más segura posible.
- Realizar el modelado de amenazas: El Equipo de Seguridad debe realizar un Modelado de Amenazas por componentes y activos, en conjunto con el Arquitecto, el Ingeniero de Casos de Uso y el Ingeniero de Componentes. Más adelante se abordará en detalles esta parte del diseño como artefacto decisivo en la calidad y confiabilidad final del software.
- Elaborar una guía de codificación segura que permita reducir los errores de seguridad que los desarrolladores introducen u olvidan durante sus rutinas de escritura del código fuente.
- Preparar al Equipo de Producto para entrar al Flujo de implementación de manera segura, con los elementos claves y claros de cómo garantizar un proceso de codificación exitoso.

En el resto del epígrafe se presentará cómo conseguir estos objetivos.

2.2.2 Trabajador: Líder del ES.

El flujo de trabajo actual implica una carga muy marcada, en cuanto a la cantidad de actividades en que se ve involucrado el ES. Sus labores deben estar centradas en abarcar el mayor número de áreas importantes para la seguridad con el objetivo de evitar que prevalezcan puntos escondidos o al menos sin identificar. El Líder del ES tiene, entonces, una tarea de mucho peso en el inicio del flujo, distribuir las tareas y responsabilidades de cada uno de los miembros en diferentes áreas del proceso

de gestión. Este es el momento en que las responsabilidades se hagan valer, aunque pueden surgir otras e incluso aumentar el número de especialistas designados en un primer momento para una tarea específica, en dependencia de la complejidad de la misma. Cada uno de los miembros debe participar en la generación de artefactos de seguridad o actividades durante el proceso de gestión de la seguridad que comprende este flujo.

La capacitación al Equipo de Producto es también responsabilidad del Líder del Equipo de Seguridad durante el flujo actual. Para desarrollar esta actividad puede valerse de los miembros del ES para que en determinado momento ofrezcan detalles sobre la actividad específica que desarrolló cada cual durante el flujo y los resultados que obtuvo. Más adelante se especifica con más claridad los aspectos que se deben tener en cuenta para desarrollar tal actividad.

Por último el Líder del ES tiene como responsabilidad elaborar el artefacto Informe final de resultados el cual también se detalla más adelante.

2.2.3 Trabajador: Arquitecto de seguridad.

El Arquitecto de seguridad es el trabajador designado por el Líder del ES para encargarse de los artefactos Modelo de Amenazas y Medida de mitigación de riesgos para cada una de las vulnerabilidades encontradas y modeladas. Las actividades que permiten generar estos artefactos se desarrollan en conjunto con el trabajador Arquitecto, los Ingenieros de casos de uso y los Ingenieros de componentes, establecidos por RUP para este flujo.

2.2.4 Trabajador: Redactor.

El trabajador Redactor es el especialista, designado por el Líder del ES, encargado de elaborar el artefacto Guía de diseño de codificación segura para el lenguaje y plataformas de desarrollo definidos por el Equipo de Producto. Para esto se requiere conocimientos de programación y es necesario llevar un proceso investigativo donde se obtenga documentación relacionada con el lenguaje que describa buenas prácticas para el desarrollo seguro. Esta actividad se realiza en conjunto con el Arquitecto del sistema. Más adelante se detalla con detenimiento los aspectos a tener en cuenta en la guía.

2.2.5 Trabajador: Diseñador de Procedimiento de mitigación.

El especialista Diseñador de Procedimiento de mitigación tiene la responsabilidad, dentro del flujo de análisis y diseño, de generar y especificar cada uno de los pasos que son necesarios para una vulnerabilidad. A medida que el Arquitecto de seguridad va identificando las medidas de mitigación el Diseñador de Procedimiento de mitigación elabora los pasos detallados a seguir para implementar la medida. Más adelante se detalla el artefacto y las ventajas que ofrece su generación para desarrollos futuros.

Para terminar esta parte correspondiente a los especialistas de seguridad que participan en el flujo actual, es válido concluir que las actividades en que cada rol del ES participa tienen como objetivo fundamental dar como salida un artefacto de seguridad detallado y completo. Los trabajadores del Equipo de Producto, normalmente serán los mismos definidos por RUP y que a partir de los artefactos de los que son responsables durante el diseño pues introducen elementos importantes al especialista del ES para generar sus artefactos, de manera conjunta.
A continuación se presenta la estructura básica del flujo.

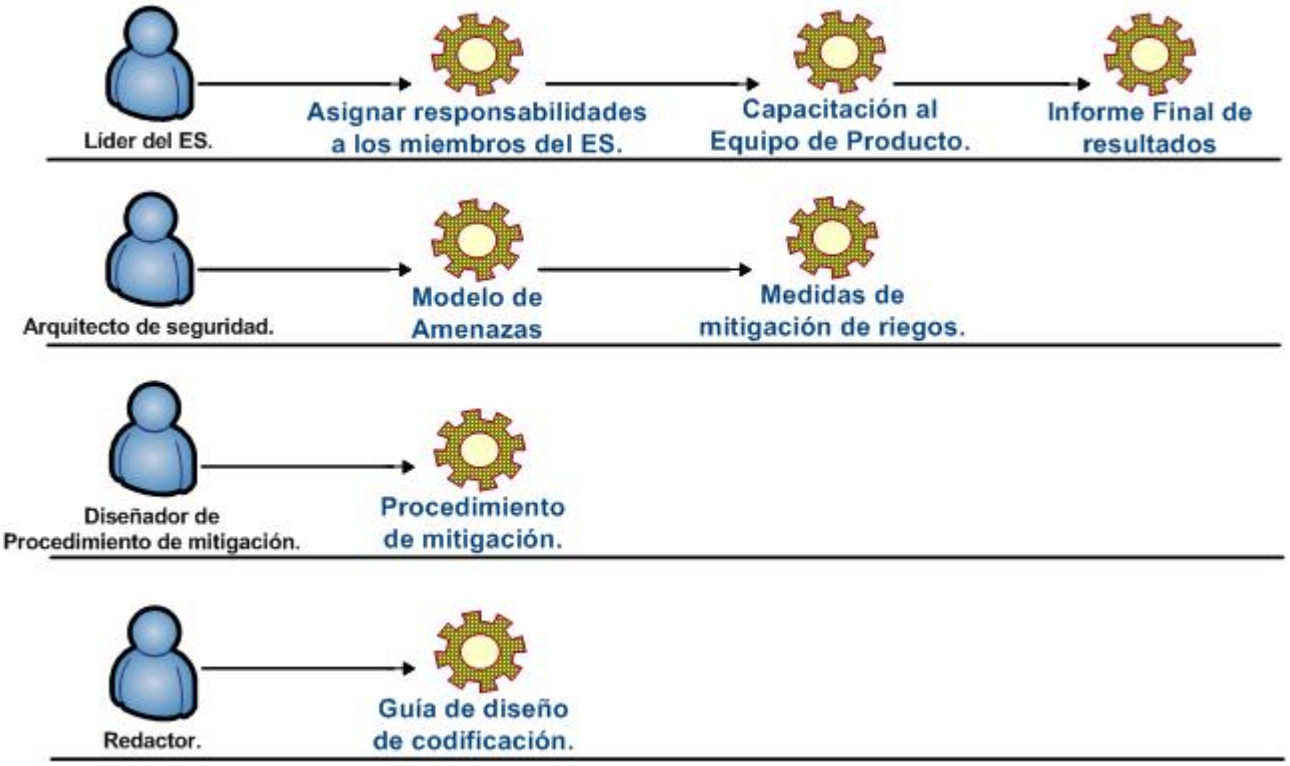


Figura 4. Trabajadores y artefactos de seguridad involucrados en el Análisis y Diseño del sistema.

El Líder del ES es el responsable de guiar el proceso de gestión de seguridad durante todo diseño, chequea el avance de cada uno de los artefactos, su articulación e integración con los demás, sin embargo las especificaciones particulares de cada una de las actividades y artefactos generados es responsabilidad de cada trabajador responsable de la tarea. Incluso dentro de la realización de un determinado artefacto de seguridad, pueden existir tareas específicas que requieran más de un especialista, tal es el caso del Modelo de Amenazas, el cual puede llegar a ser muy complejo y es recomendable asignar más de un especialista para que desarrollen determinados actividades dentro del modelado, los cuales quedarían bajo la dirección del Arquitecto de Seguridad como máximo responsable del artefacto Modelo de Amenazas.

2.2.6 Artefacto: Modelo de Amenazas.

El modelado de amenazas es una técnica usada para identificar, comprender y planificar de forma correcta la mejor manera de mitigar las amenazas que asechan a un sistema informático, ayuda a

encontrar los puntos vulnerables de la aplicación y a asignar prioridades (Scheibelhofer 2005; P.F 2006). La finalidad del modelado de amenazas es preparar las defensas adecuadas durante los flujos de Diseño, Implementación y también durante su posterior Revisión y Pruebas. Se trata de identificar las amenazas y definir las contramedidas adecuadas que nos permita mitigar el riesgo a unos niveles aceptables, de una forma efectiva y en base a unos costes razonables (P.F 2006). Puede resumirse que, para lograr tales propósitos, se identifican los activos que deben administrar el software y las interfaces que permitirán el acceso a dichos activos. El proceso de modelado de amenazas identifica las amenazas que pueden dañar a estos activos y la probabilidad de que se inflija dicho daño (estimación del riesgo). A continuación, se identifican las contramedidas que pueden mitigar el riesgo, ya sea mediante características de seguridad (por ejemplo, el cifrado) o mediante un funcionamiento correcto del software que proteja a los activos del daño. Por tanto, el modelado de amenazas ayuda a:

- Identificar donde una aplicación es más vulnerable.
- Identificar qué amenazas requieren mitigación.
- Reducir los riesgos a niveles aceptables.

El proceso debe realizarse con una herramienta capaz de capturar modelos de amenazas en un formato que pueda leer el equipo, para almacenarlo y actualizarlo.

Es importante resaltar que el modelado de amenazas no es tarea única del ES. La idea que persigue esta técnica es que los diferentes entes involucrados (desarrolladores, probadores, gerencia, administradores de sistemas, consultores, y demás) participen en el proceso de identificación de las posibles amenazas. Tomando una actitud defensiva y también poniéndose en ocasiones, en el papel de un posible atacante. Siguiendo este enfoque, se fuerza a todos ellos a explorar las debilidades que puedan surgir o que ya estén presentes, y se determina de este modo si existen las oportunas contramedidas, o en caso contrario, se documentan y se asumen riesgos. El proceso de modelado de las amenazas, requiere de la intervención de varios trabajadores del EP, entre ellos están el Arquitecto, el Ingeniero de casos de uso y el Ingeniero de componentes, más adelante e indistintamente se especificarán sus aportes.

2.2.6.1 Consideraciones previas al proceso de modelado.

La explotación con éxito de un sólo fallo de seguridad podría llegar a comprometer todo el sistema. Por ello, y para conseguir que nuestro modelado de amenazas resulte efectivo, es importante tener

en cuenta que se debe realizar como un proceso sistemático, en continua actualización. De este modo conseguiremos identificar y mitigar el mayor número posible de amenazas y vulnerabilidades. Nuestro modelado de amenazas debería 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 condiciones o aquellas vulnerabilidades que, una vez eliminadas o contrarrestadas, afectan a la existencia de múltiples amenazas.
- Proporcionar información relevante sobre cuales serían las contramedidas más eficaces para contrarrestar un posible ataque y/o mitigar los efectos de la presencia de una vulnerabilidad en nuestro sistema/aplicación.
- Proveer información sobre cómo las medidas actuales previenen la consecución de ataques.
- Transmitir a la gerencia la importancia de los riesgos tecnológicos en términos de impacto de negocio.
- Proporcionar una estrategia sólida para evitar posibles brechas de seguridad.
- Facilitar la comunicación y promover una mejor concienciación sobre la importancia de la seguridad.
- Simplificar la posterior actualización mediante el uso de componentes reutilizables.

Teniendo esto presente, el ES procede a realizar las labores de modelado de forma inmediata.

2.2.6.1 Determinar el alcance del sistema.

Los pasos que a continuación se analizarán constituyen las primeras entradas que guiarán el proceso de modelado de amenazas. El objetivo de estos radica en identificar y establecer los límites y el alcance del sistema, para garantizar que las amenazas encontradas sean las correctas y las que realmente están dentro de nuestro control.

Paso 1. Definir los objetivos del sistema.

Todos los sistemas de software que se desarrollan son concebidos con objetivos específicos. En la mayoría de los casos, surgen de la necesidad de humanizar los procesos de negocios que tradicionalmente resultan engorrosos de realizar debido a su realización manual o en condiciones adversas. Por tanto los objetivos del negocio, son los objetivos del sistema. El primer elemento que debe tener como entrada el modelado de amenazas es la delimitación del alcance del sistema, a

partir de los objetivos reales de su concepción. El Líder del ES, a estas alturas del proceso, posee ya una visión clara, proveniente de la captura de requisitos y sus contactos frecuentes con el Analista de Sistema en dicho flujo. Por tanto, esta visión debe quedar documentada y su conocimiento debe ser obligatorio para el resto del ES y el EP. No se concibe iniciar el modelado sin que los involucrados conozcan los objetivos del sistema de forma clara. Si embargo, no es suficiente solo con la visión adquirida por el asesor, sino que debe, ponerse en contacto con la gerencia del producto de software y, en conjunto, dejar de manera concisa y exacta tales objetivos.

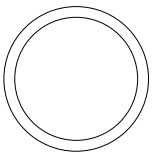
Paso 2. Artefacto: Diagrama de Contexto de alto nivel del sistema.

Una vez definidos y documentados los objetivos del sistema se procede a comprender el contexto en que esta funcionará. Para esto se realiza un Diagrama de contexto, de manera que se reflejen los principales **procesos** y **entidades** que interactúan con el sistema.

El Diagrama de contexto debe desarrollarse a un alto nivel, por lo que no debe caer en especificidades que entorpezcan y confundan el objetivo que se persigue con su representación, ya que las cuestiones particulares son reflejadas por los Diagramas de Flujo de Datos (DFDs), que posteriormente desempeñan un papel fundamental en el modelado. El Diagrama de contexto, al igual que los DFDs, es recomendable que se realicen usando una herramienta UML o mediante la herramienta de modelado de amenazas y análisis de riesgos seleccionada.

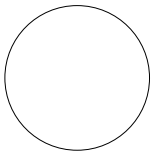
En la construcción del diagrama participan el trabajador Arquitecto de sistema y el Arquitecto de Seguridad, encargado del Modelo de Amenazas.

A continuación se muestran los estereotipos para representar los entes que intervienen en la realización del Diagrama de contexto y los DFDs. El Diagrama de contexto es un supraconjunto del DFDs, porque de él se desprenden los DFDs específicos, pero solo contiene un proceso y no se representan almacenes de datos. Fundamentalmente durante su construcción, se usan los estereotipos de **Proceso múltiple**, **Entidad, Conector de flujo** e **Interfaz confiable.**



Proceso múltiple: Los procesos múltiples son aquellos que manipulan o transforman datos pero que pueden ser divididos en procesos más simples. Un proceso puede ser físicamente un centro de procesamiento, un procedimiento, o una combinación de actividades manuales y

automatizadas. El Diagrama de contexto estará representado principalmente por este tipo de procesos, ya que los simples serán tratados en los DFDs fundamentalmente. Algunos ejemplos de Procesos múltiples son: Servicios Web, Servidores de Base de datos, Servicios externos, Servidor de aplicación, incluso la aplicación de software misma (Williams 2005).



Proceso simple: Representa un proceso que se deriva de un proceso múltiple, pero que no puede descomponerse en más procesos.

Entidad: Representa las entidades a más alto nivel que interactúan con los procesos y que muchas veces proporcionan las entradas a la aplicación. Son generalmente clases lógicas de cosas o de personas, las cuales representan una fuente o destino de transacciones, como por ejemplo clientes, empleados, proveedores, con las que el sistema se comunica. También pueden ser una fuente o destino específico, como por ejemplo un Departamento Contable, en caso que el sistema bajo análisis reciba datos de otro sistema o bien se los provee, en este caso sería una Entidad Externa.

Mediante la designación de alguna cosa o de algún sistema como Entidad Externa estamos estableciendo implícitamente que se encuentra fuera de los límites del sistema que estamos considerando por lo cual no nos interesa la transformación o proceso que se realiza dentro de ellos. Son sólo proveedores o requeridores de datos del sistema bajo consideración.

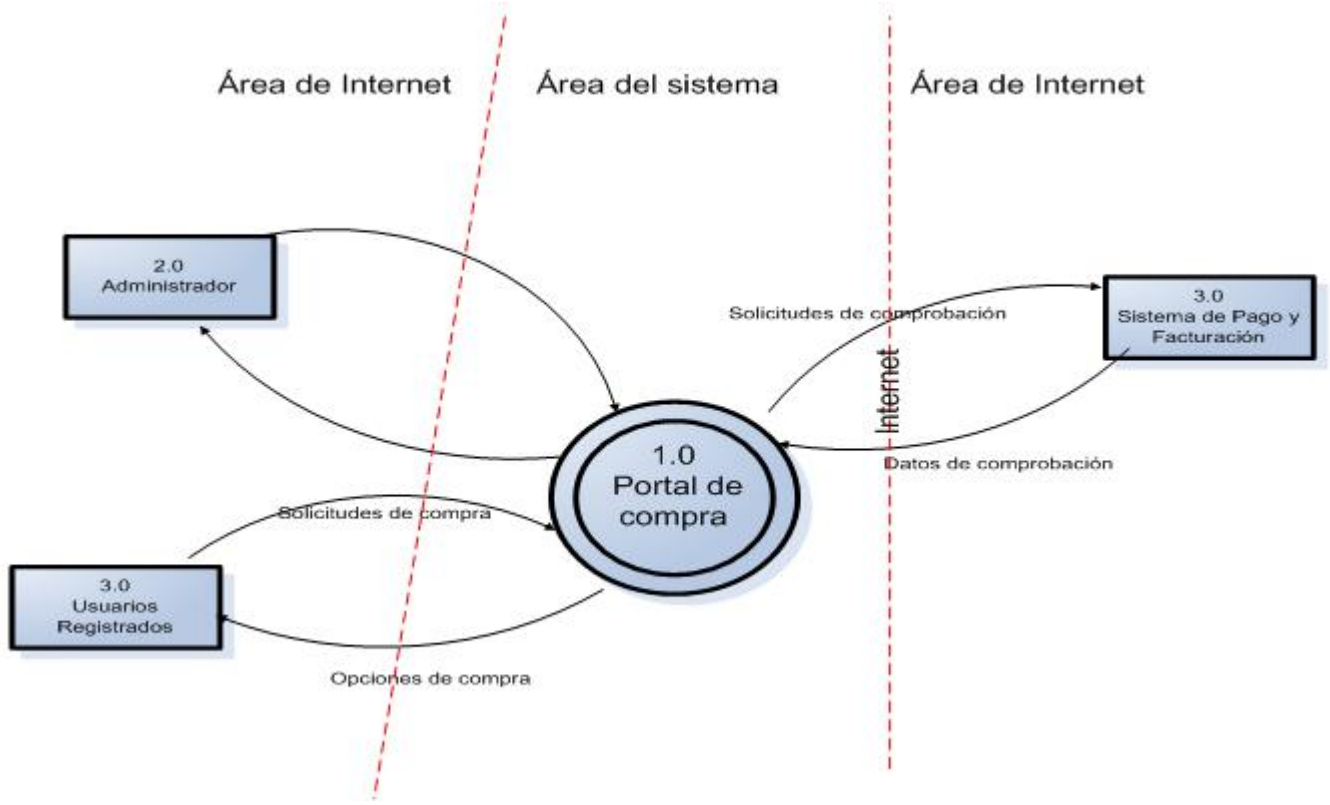
Almacén de datos: Representa un archivo lógico en donde se agregan o de donde se extraen datos. Es una estructura de datos. Pueden ser ficheros de configuración, colecciones, valores del registro.

Conector de flujo: Representa las conexiones entre los procesos, las entidades y los almacenes de datos. Debe contener los mensajes que implican la conexión, el cual deberá elegirse

de forma que sea lo más útil posible a los usuarios que revisen el DFDs. Representa un transporte de paquetes de datos desde su origen hasta su destino, es decir que representa una estructura de datos en movimiento de una parte del sistema a otra. La flecha indica la dirección del flujo. Algunos ejemplos ilustrativos son: tráfico en la red y llamada de funciones.

Interfaz confiable: Representa la frontera de seguridad del sistema a partir de la cual todas las entradas son consideradas seguras y no es necesario validarlas. Para ello se definen puntos de chequeo de seguridad en el límite de esas fronteras de manera que todas las entradas deban pasar por algún punto antes de ser admitidas en la zona de seguridad. Cada uno de estos puntos se encarga de entradas diferentes, ya sean almacenes de datos y entradas de usuarios, entre otros y se notifica de manera adecuada los errores.

El siguiente ejemplo permite ilustrar un Diagrama de Contexto hipotético para un portal de compras por Internet. El portal tiene como función principal realizar ventas de artículos domésticos por Internet. Para ello se registra en el sistema el cliente y mediante el sistema de navegación del portal puede seleccionar aquellos artículos que desea e irlos adicionando a su cesta virtual de productos, una vez que desee terminar la compra introduce el número de su tarjeta de crédito la cual es validada por un sistema de pago y facturación externo finalizando así el proceso.
El problema funcionará como ejemplo durante el resto del capítulo.



Ejemplo 2**.** Diagrama de contexto para el portal de compras de ejemplo.

El Diagrama de contexto proporciona una vista preliminar importante del alcance del sistema mediante un gráfico. Cuando no se realiza esta representación se corre el riesgo de terminar el modelado, perdiendo el tiempo en analizar una gran cantidad de amenazas que posiblemente no estén dentro del alcance real y control del sistema a implementar.

Paso 3. Identificación de los Escenarios de uso del sistema.

Una vez realizado el Diagrama de contexto, el paso siguiente es definir los escenarios críticos desde el punto de vista de la seguridad donde operará el sistema y que a su vez interactuará con las entidades que se reflejaron en el paso anterior. Debe comenzarse identificando los escenarios más comunes y reales, los cuales deben comprenden:

- Locales en que residirá el sistema y el equipamiento de hardware así como sus condiciones.
- Las personas que acceden regularmente a estos locales y sus niveles de acceso.
- Los sistemas externos que, controlados o no por el sistema, influyen directamente en la calidad del funcionamiento de este último.

- Si es un sistema que recibirá entradas desde la red, es necesario identificar las características de la red que se usará.

Estos son solo una pequeña parte del ambiente y posibles escenarios en que deberá operar la aplicación y demás recursos. El tipo de escenario variará como es lógico en dependencia del tipo de aplicación y de los procesos de negocios que se manejen. Los escenarios deben ser documentados mediante un número de identificación y una descripción que refleje claramente lo que representa. Incluso pueden existir relaciones entre escenarios como muestra el siguiente ejemplo:

Ejemplo 3. Relaciones entre escenarios de uso para el portal de compras de ejemplo:

La aplicación será instalada en la oficina general de contabilidad de la empresa X, y recibirá vía fax las facturas de pago de los clientes, generadas por el sistema de facturación del banco Y. Sin embargo el sistema de facturación está fuera del control del sistema nuestro.

En el ejemplo se muestran claramente dos escenarios de uso importantes en que operará la aplicación: oficina general de contabilidad de la empresa X, conexiones de red para recibir vía fax las facturas. Sin embargo el sistema de facturación y el canal de comunicación no lo controla el sistema que se implementará. Los escenarios de uso, permiten encontrar amenazas que no solo pertenecen al software, sino también al entorno en que este sobrevive, y aunque muchas veces no están bajo el control nuestro, si deben ser de conocimiento, incluso darle algún tratamiento en el Plan de contingencia. La tabla de Escenarios de uso provee información acerca del uso esperado de la aplicación y sus dependencias. Usar o desplegar el sistema en una forma que viole un escenario de uso puede tener impacto en la seguridad.

Escenarios de uso.	
Nro.	**Descripción.**
	•
	•

Tabla 2. Escenarios de uso esperados.

Paso 4. Representación de la arquitectura de seguridad del sistema.

Crear arquitectura es lidiar con la complejidad de cualquier sistema. Por consiguiente, el papel de la arquitectura en la seguridad de una aplicación es clave. El objetivo que se persigue con este paso es

definir la estructura global del software e identificar los componentes cuyo correcto funcionamiento es esencial para la seguridad.

La dentificación de técnicas de diseño seguras para la arquitectura tales como el uso de capas. permite organizar el software en componentes bien definidos que se estructuran para evitar dependencias circulares entre componentes. Los componentes de una capa superior pueden depender de los servicios de capas inferiores, pero se prohíbe que las capas inferiores dependan de las capas superiores. La actividad definida por RUP como **Diseño de la arquitectura** y donde participa el Arquitecto de Sistema es el momento ideal para que en conjunto con el Arquitecto de Seguridad se identifique los nodos en que se distribuirán los componentes claves, subsistemas e interfaces. Los aportes del especialista de seguridad pueden ser valiosos, ya que con los conocimientos que posee, contribuye a estructurar y seleccionar componentes confiables que permitan formar la línea arquitectónica del sistema.

Identificación de componentes.

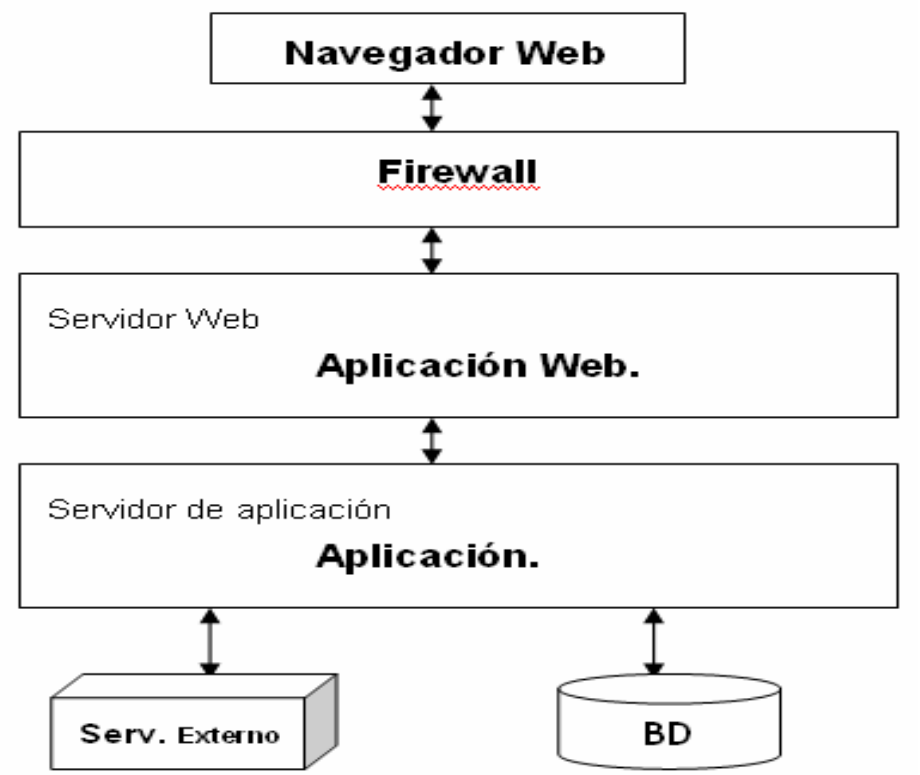
El Arquitecto de seguridad en conjunto con el trabajador Arquitecto del sistema identifica los componentes relevantes para la seguridad, así como las relaciones entre ellos. Los componentes pueden ser servidores de base de datos, de aplicaciones, servicios externos, sistemas intermediarios, paquetes de diseño, entre otros muchos. Debe aclararse que el objetivo de esta tarea, en este momento, no es identificar y mitigar las amenazas que acechan nuestra arquitectura, esto se hace más adelante, cuando sean incluidos en la lista de activos críticos y se realice el proceso de identificación de amenazas sobre los mismos. El objetivo principal radica en definir aquellos componentes de alto nivel que forman la línea base de la arquitectura. Algunos componentes pueden ser:

- Navegador Web.
- Firewall[19].
- Aplicación Web.
- Servidor Web.
- Servidor de base de datos.

[19] Software que controla el acceso de los usuarios a los recursos de una red de computadoras.

- Servicios externos.

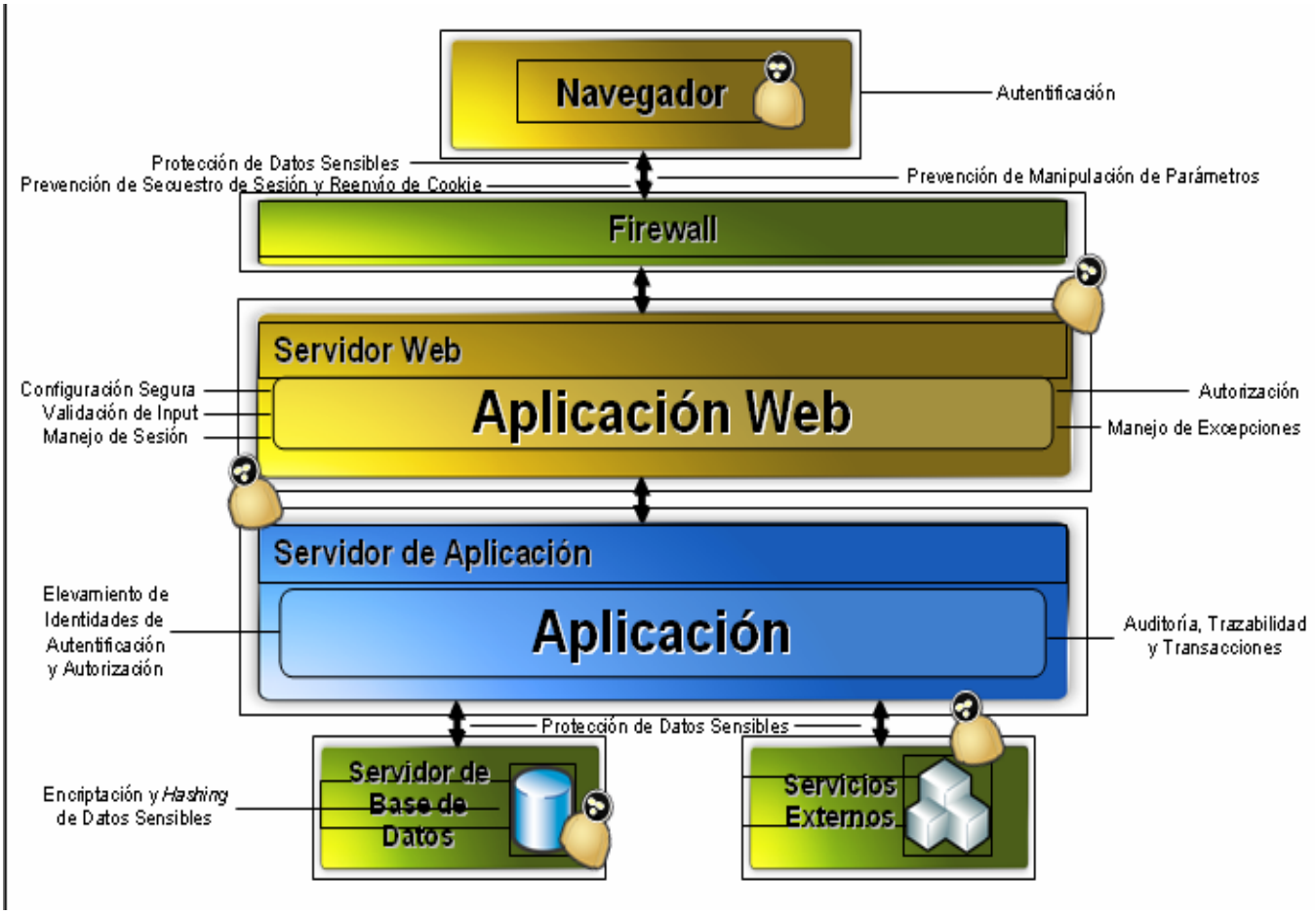
Una vez identificados se procede a integrarlos en un modelo de arquitectura. El siguiente ejemplo muestra la arquitectura inicial para el ejemplo del portal de compras.



Ejemplo 4. Arquitectura inicial con los componentes identificados para el portal de compras de ejemplo.

Estos componentes constituyen la base fundamental sobre la que se mantiene la arquitectura y permiten adquirir un conocimiento clave que acota el área de interés de la seguridad, sobre el cual se volverá más adelante cuando se proceda a identificar amenazas y los mecanismos de mitigación necesarios para garantizar la seguridad de las interfaces de comunicación entre las capas.

A continuación se muestra un ejemplo completo de medidas de mitigación de riesgos por componentes en el diseño de arquitectura del portal de compras de ejemplo.



Ejemplo 5. Arquitectura de seguridad con medidas de mitigación por componentes.

2.2.6.2 Descomposición del diseño.

Una vez que se delimita el alcance del sistema, a partir de los pasos anteriores, procedemos a descomponer el diseño en partes manejables que permitan hacer un análisis más específico con cada uno de los componentes que son críticos para la seguridad.

1. Identificación de activos.

Se denominan activos a los recursos del sistema de información o relacionados con éste, que tienen un valor para la Organización, bien en sí mismo o porque de ellos dependen otros activos de valor. La importancia de identificar activos radica en que todas las amenazas y ataques se realizarán sobre ellos (Madrid 2006).

Los activos importantes para un sistema abarcan desde la información que se manipula hasta los soportes y servicios. Estos se dividen en dos grandes grupos: activos lógicos y activos físicos:

- **Activos lógicos:** Referido a aquellos elementos intangibles que se manipulan mediante un sistema de información o están relacionados con este. Entre estos se encuentran:
    - Las aplicaciones informáticas (software) que permiten manejar los datos.
    - Los servicios que se pueden prestar y los servicios que se necesitan para poder gestionar dichos datos.
    - El código en que está escrito el software.
    - Los protocolos de comunicación de redes.

  En fin todo lo que no está en formato duro, pero que puede tener un impacto determinante en el funcionamiento de la empresa y las aplicaciones. La ayuda del Ingeniero de casos de uso, y el Ingeniero de componentes, contribuye a encontrar activos lógicos de importancia a partir de las clases del diseño, que modelan entidades del mundo real y que en una vez abstraídas pues se convierten en activos lógicos críticos, que hay que conservar.
- **Activos físicos**. Comprenden todo el equipamiento tangible y en formato duro que dentro de una organización manipulan los sistemas de información o se relacionan con este. Este tipo de activo incluye:
    - Los equipos informáticos (hardware) y que permiten hospedar datos, aplicaciones y servicios.
    - Los soportes de información que son dispositivos de almacenamiento de datos.
    - El equipamiento auxiliar que complementa el material informático.
    - Las redes de comunicaciones que permiten intercambiar datos.
    - Las instalaciones que acogen equipos informáticos y de comunicaciones.
    - Las personas que explotan u operan todos los elementos anteriormente citados.

Los elementos identificados durante la actividad de **Diseño de la arquitectura** pasan a formar parte del conjunto de activos físicos que necesitan protección.

Los activos necesitan protección para asegurar las correctas operaciones del negocio y la continuidad de la empresa. La identificación de activos importantes para la gestión de la seguridad, debe hacerse con mucho cuidado y analizar el impacto que puede tener este sobre otros activos. Si un activo no implica ningún riesgo de seguridad, entonces no debe ser contemplado dentro del artefacto Conjunto

de Activos de Seguridad (CAS). Como todos los activos no son de la misma especie, las amenazas también son diferentes[20] (Madrid 2006), de ahí la importancia de clasificar los activos.

No siempre es evidente qué es un activo en singular. Si por ejemplo en el proceso de desarrollo hay 300 computadoras como estaciones de trabajo de programación, todas idénticas en cuanto a efectos de configuración y función que realizan, no es conveniente analizar 300 activos idénticos. Basta analizar una estación genérica cuya problemática representa la de todas. Agrupar simplifica el modelo.

Otras veces se presenta el caso contrario, un servidor central que se encarga de varias funciones: servidor de ficheros, de documentación, del control de cambios, del sistema de gestión documental entre otras tareas. En este caso conviene segregar los servicios prestados como activos independientes.

Es necesario dejar claro que no se busca qué es necesario para que el sistema funcione, sino al revés, se busca dónde puede fallar el sistema o, más precisamente, dónde puede verse comprometida la seguridad de los activos.

Estas consideraciones pueden plantearse con argumentos del tipo:

- Si usted quisiera acceder a estos datos, ¿dónde atacaría?
- Si usted quisiera detener este servicio, ¿dónde atacaría?

Este planteamiento de “póngase en el lugar del atacante” es el que da pie a las técnicas denominadas

“Árboles de ataque” que se verán más adelante (Madrid 2006). Un activo puede ser atacado directamente o indirectamente a través de otro activo del que dependa.

A continuación se propone el formato en que deben quedar documentados los activos mediante el uso de una tabla.

Nro.	Nombre	Activo lógico	Activo físico	Impacto
	•			
	•			

Tabla 3. Representación del artefacto Conjunto de Activos de Seguridad.

[20] No se ataca ni se defiende de igual manera un servicio telemático que un local de trabajo.

Nro: Un número consecutivo para cada activo que se incluya en el CAS.

Nombre: El nombre de cada activo que se incluye en el CAS.

Activo lógico, Activo físico: Se marca con una cruz (X) el grupo donde clasifica el activo identificado.

Impacto: El impacto que provoca sobre la organización u otros activos, puede ser Alto, Medio, Bajo. Los activos que no tienen ningún impacto sobre otros activos o la organización, pues no son importantes para la seguridad, por tanto no se reflejan en el CAS.

En la identificación de amenazas y el plan de mitigación deben tratarse los activos por orden de prioridad, el cual va a estar dado por el nivel del impacto que se reflejó.

2. Definición de roles de usuarios.

Los roles definen los niveles de confianza del sistema y, en primer lugar, son usados para tomar decisiones de autorización. En el momento de identificar y documentar los roles deben tenerse presente varios aspectos:

- Cualquier usuario que interactuará con la aplicación debe ser asignado a un rol de usuario.
- Para cada rol establecer el mecanismo de autenticación.
- Identificar el número de entidades aproximadas que existirán para cada rol encontrado.

Los roles identificados se representan mediante una tabla. A continuación se muestra el formato.

Nro.	Nombre	Mecanismo de autenticación	Número de entidades
	•	•	•
	•	•	•

Tabla 4. Representación de los roles de usuario del sistema.

Nombre: Representa el nombre que identificará al rol. Usuario Registrado, Administrador, son ejemplos de roles de usuario.

Mecanismo de autenticación: Representa el mecanismo de autenticación que emplea un rol para acceder a las funcionalidades del sistema. Autenticación Integrada de Windows, Certificados Digitales, Formulario, entre muchos otros.

Número de entidades: Representa aproximadamente la cantidad de entidades de un determinado rol que harán uso del sistema. Puede ser representado por un rango o por un número si se conoce exactamente. 1-5 es un ejemplo de rango.

Los roles son usados generalmente para establecer permisos de acceso y manipulación de los datos almacenados por ejemplo en un Sistema Gestor de Base de Datos(SGBD). Estos permisos pueden ser de lectura, escritura, actualización y eliminación. El paso siguiente tiene como objetivo identificar esos datos importantes.

3. Definición de los datos almacenados.

Los datos definen el tipo de información que es mantenida o procesada por la aplicación de software y que normalmente es persistente. La tarea de identificación de datos se realiza a partir del Modelo de Datos que se genera. El uso fundamental de esta información estará en la conformación de la Matriz de Control de Acceso, su representación en el documento de seguridad debe tener la siguiente estructura.

Nombre	**Atributos (opcional)**	**Clasificación**
	•	
Breve descripción.		

Nombre: Representa el dato en el modelo.

Atributos: Representar los atributos es opcional, en algunas ocasiones resulta útil mostrar algunos atributos críticos para un determinado dato.

Clasificación: Grado de importancia del dato para la organización. (Alto, Medio, Bajo).

Breve descripción: Muestra un breve resumen del dato en cuestión. Su objetivo, el impacto, entre otros.

4. Matriz de Control de Acceso.

Una vez que se identifican los roles de usuarios y se definen los elementos de datos el siguiente paso tiene como objetivo establecer los permisos que tienen los roles sobre cada tipo de datos, a esta tarea se le define como Matriz de Control de Acceso. Permite tener un control de quién tiene acceso a determinada información, y cuáles son las acciones que puede realizar sobre esta. La siguiente tabla

muestra los elementos que componen la matriz y la estructura que lleva su representación en el Documento de gestión de seguridad.

Matriz de Control de Acceso					
Dato:					
Rol de usuario	**Insertar**	**Leer**	**Actualizar**	**Eliminar**	**Condición**

Tabla 5. Permisos de los roles sobre cada dato protegido.

Rol de usuario: Se escribe cada rol que tiene un determinado permiso sobre un elemento de dato.

Insertar: Se marca con una cruz (X) el permiso que posee sobre el elemento de dato. Lo mismo sucede sobre los demás permisos.

Condición: Existen permisos que dependen de alguna condición específica. Estas condiciones se representan en la columna y se especifica para cual tipo de permiso es que se requiere mediante la primera letra del permiso.

Ejemplo 6: En el portal de ventas por Internet un cliente puede crear, eliminar, leer y actualizar la información de una cuenta solo si es el propietario.

Para este ejemplo la Matriz de Control de Acceso es:

Dato:	Cuenta de cliente				
Rol de usuario	**Insertar**	**Leer**	**Actualizar**	**Eliminar**	**Condición**
Usuario Registrado	X	X	X	X	L,A,E: Solo si es el propietario

5. Diagrama de Flujo de Datos.

El Diagrama de Flujo de Datos es un modelo que describe los flujos de datos, los procesos que cambian o transforman los datos en un sistema, las entidades externas que son fuente o destino de los datos (y en consecuencia los límites del sistema) y los almacenamientos o depósitos de datos a los cuales tiene acceso el sistema, permitiendo así describir el movimiento de los datos a través del sistema. A diferencia del Diagrama de contexto, los DFDs sí reflejan los almacenes de dato (Howard

and LeBlanc 2002). Los DFDs ayudan comprender la lógica de la aplicación y saber cómo puede afectar el tratamiento de los datos a la integridad de los activos.

En síntesis, el Diagrama de Flujo de Datos describe:

- Los lugares de origen y destino de los datos (los límites del sistema).
- Las transformaciones a las que son sometidos los datos (los procesos internos).
- Los lugares en los que se almacenan los datos dentro del sistema.
- Los canales por donde circulan los datos.

Guía para construir Diagramas de Flujo de Datos.

- Primero se deberán identificar las entidades externas ya que ello implica definir los límites del sistema.
- Se deberán elegir nombres con significado tanto para procesos como también para flujos de datos, almacenes y entidades externas. Si es posible a partir del vocabulario del usuario evitando terminologías técnicas.
- Identificar el papel del proceso del sistema, no quien lo realiza.
- Numerar los procesos, mediante un esquema de numeración consistente consecutivo.
- Se deberán, en la medida de lo posible, evitar los DFDs excesivamente complejos. Deberán ser comprensibles, y agradables a la vista sin demasiados elementos.
- Todos los elementos se relacionan entre sí a través de conectores de flujos de datos.
- Procesos: Se relacionarán con:
    - Almacenes de datos.
    - Entidades.
    - Otros procesos.
    - Deberán tener al menos una Entrada y una Salida, no son de solo entrada.
- Almacenes de datos: Se relacionarán solamente con Procesos.
- Para evitar el cruzamiento de las líneas de conectores de flujo de datos, la misma entidad (o el mismo almacén) se podrá dibujar más de una vez en el mismo diagrama.

Los DFDs permiten, además, identificar puntos de entrada o salida dentro del contexto de la aplicación en que se desarrolla un determinado flujo. Los puntos fronteras, son aquellos lugares

donde pueden existir entrada o salida de datos y que muchas veces no se validan porque no se ven a simple vista. Por tanto una decisión importante es establecer medidas de validaciones. La identificación de estos puntos puede contribuir a establecer las fronteras de la aplicación a partir de las cuales se considera seguro el flujo de datos.

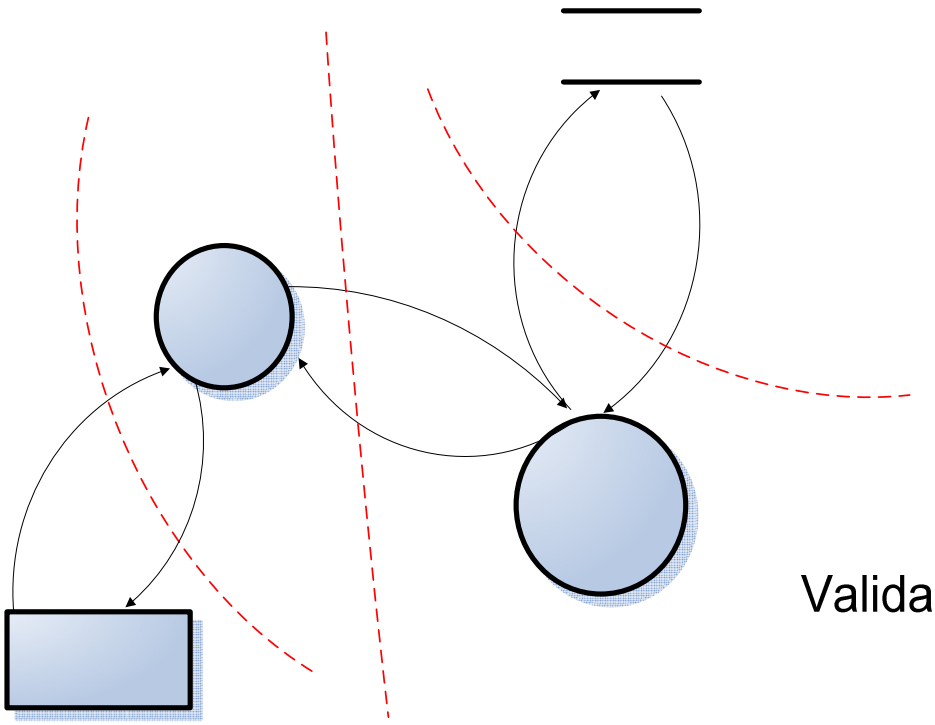


Figura 5. Diagrama de flujo de datos para el proceso de autenticación de usuario.

6. Identificación de amenazas sobre los activos.

Con la información recopilada y en función del contexto y el escenario de la aplicación se procede a la identificación de las amenazas más importantes. Las amenazas representan lo que el atacante desea y deben ser descritas con un nombre claro que identifique lo que realmente se persigue. Existen una serie de consideraciones que es fundamental tener cuenta antes de iniciar el proceso de gestión y representación de amenazas.

- Asignar más de un especialista del ES para realizar la tarea de identificación y clasificación.
- Involucrar a todo el personal del EP. Esta es una tarea de todos y no solo del ES.

- Tener listado todos los activos críticos y por grado de importancia, teniendo en cuenta su impacto en la organización, además de los puntos de entrada y salida proporcionados por los DFDs.
- Definir un método de clasificación de las amenazas, según el efecto que tengan sobre los activos que se desean proteger. El método de clasificación propuesto en el modelo es el método STRIDE (Ver Anexos 1).
- Definir un método de puntuación para las amenazas según el riesgo que suponen. El método propuesto por el modelo es el método DREAD (Ver Anexos 2).
- Disponer de una herramienta de modelado que permita generar árboles de ataques.
- No intentar mitigar las amenazas en este paso, el objetivo debe estar centrado en identificarlas y clasificarlas y priorizarlas.

A continuación se especifican los pasos para la identificación y tratamiento de las amenazas de una manera efectiva.

- Seleccionar el activo a analizar en orden de importancia e impacto de mayor a menor.
- Ponerse en la posición del atacante lo que significa pensar como un verdadero atacante.
- Identificar las amenazas que ciernen sobre el activo.
- Clasificar cada amenaza usando el método de clasificación STRIDE para identificar cuales son los efectos que puede tener una amenaza sobre un activo.
- Puntuar cada amenaza usando el método de puntuación DREAD para priorizar la amenaza según el riesgo que supone.
- Especificar si tiene mitigación inmediata.
- Desarrollar el diagrama de Árbol de ataque.

Desarrollar un Árbol de ataque:

Los árboles de ataque son una técnica usada en el modelado de amenazas que ayuda a identificar las condiciones necesarias para que pueda llevarse a cabo un ataque.

- Se realiza mediante un diagrama preferiblemente.
- El nodo de más alto nivel representa la amenaza y el número de esta.
- Los nodos siguientes reflejan los diferentes caminos que debe seguir un atacante para alcanzar su objetivo.

- Los caminos o condiciones para que se de un ataque pueden ser mutuamente excluyentes o depender de otros.
- Cada nodo se enumera por un número que está formado por el número del nodo anterior más un número que lo identifica en su nivel.

Símbolos para representar el árbol.

Nodo amenaza: Nodo de más alto nivel que representa la amenaza.

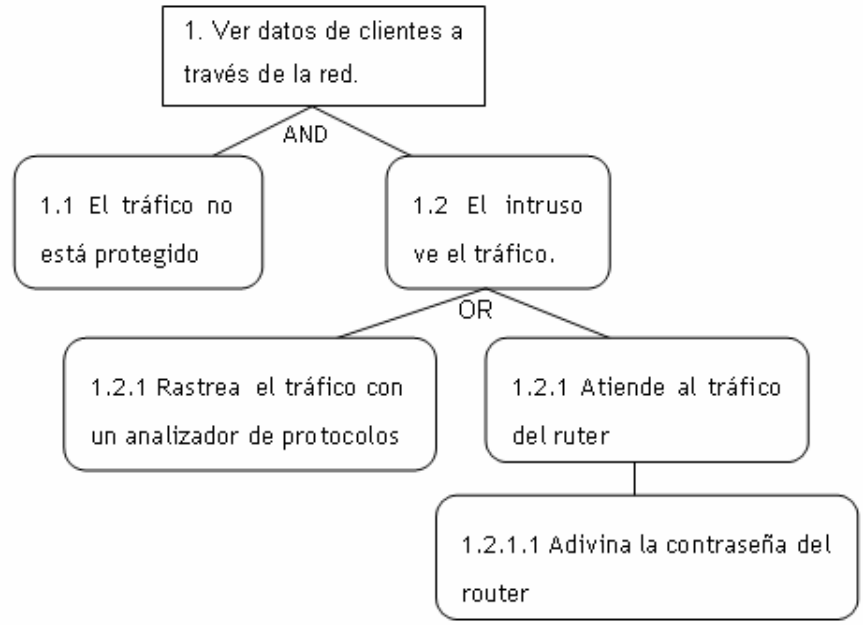
Conector: Enlaza los nodos que forman el árbol.

Operador AND: Representa los caminos que necesitan de otros para poder llevar a cabo el ataque.

Operador OR: Representa los caminos mutuamente excluyentes para llevar a cabo el ataque.

Nodo de condiciones: Nodos de niveles inferiores que representan las condiciones para que de un ataque.

A continuación se muestra un árbol de ataque que ilustra los caminos o condiciones que puede seguir un atacante para ver datos de clientes registrados en el portal de ventas por Internet.
Ejemplo 7. Un atacante intenta ver los datos de los clientes registrados a través de la red.



Representación del Árbol de ataque para el ejemplo.

A continuación se muestra lo que podría ser una típica tabla de representación de amenazas en el proceso de gestión de la seguridad.

AMENAZA				
Nro.		**Activo:**	•	
Nombre.				
Vulnerabilidad.			**Mitigación conocida.**	**Mitigable.**
•			•	
Clasificación STRIDE.	•			
Puntuación DREAD.				
Técnicas de ataque.				
Árbol de ataque (Opcional).				

Tabla 6. Documentación de las amenazas utilizando una tabla plantilla.

Para cada una de las vulnerabilidades se especifica la medida de mitigación conocida. Existen amenazas que atentan contra más de un activo, por tanto deben especificarse dichos activos.

Hasta este momento se han identificado el mayor número de amenazas que atentan contra los activos críticos que intervienen en el buen funcionamiento de un sistema informático. Sin embargo no basta con llegar hasta aquí y tener el conocimiento de su existencia, sino que es necesario pasar a la actividad de identificación de las medidas de mitigación de riesgos y amenazas en el mismo orden de prioridad dado mediante el método DREAD a las amenazas.

2.2.7 Artefacto: Medida de mitigación de riesgos y amenazas.

Llega un momento crucial en el proceso de gestión de la seguridad del sistema. Crucial porque se necesita identificar cada una de las posibles medidas que permitan mitigar las amenazas que sobre los activos existen y con ello sus respetivos riesgos. Esto implica tener mucha experiencia en estos temas y trabajar con minuciosidad para tomar las decisiones más importantes, que en el flujo de implementación garantizarán el cumplimiento y garantía de los requisitos de seguridad que se definieron en el flujo de trabajo inicial y a lo largo de todo el proceso. Identificar las medidas que solucionan o mitigan los riesgos es una tarea compleja, ya que pueden existir varias, pero debe evaluarse cada una para seleccionar la mejor opción en cada caso. Puede darse el caso de que se conozca una medida de mitigación pero sin embargo en este momento no es posible aplicar tal medida por múltiples razones, tal vez el costo, la complejidad de implementación, entre otras causas conllevan a no poder implementarla. En caso que se presente tal situación, se mantiene el campo mitigable en blanco, en caso contrario se especifica con una cruz (X). Cada medida está asociada a un Procedimiento de mitigación, el cual se detalla más adelante. Por lo tanto en la tabla de representación de cada amenaza se especifica la medida y, si su implementación es compleja, se elabora el Procedimiento de mitigación correspondiente.

2.2.7.1 ¿Cómo responder a una amenaza?

Básicamente existen tres opciones en el momento de considerar las amenazas y las técnicas de mitigación:

No hacer nada y callar: Esta pocas veces es la solución correcta porque los problemas están latentes en la aplicación y demás dependencias y en algún momento se van a descubrir y de

cualquier manera algo hay que hacer. Además porque se pone en riesgo a los clientes cuya información es procesada por el sistema y demás servicios. Aquí se cumple fielmente una de las malas prácticas de la Gestión de la seguridad: pretender hacer un sistema seguro ocultando los problemas.

Informar a los clientes de las amenazas: Con esta opción tampoco se hace nada prácticamente, sin embargo se tiene conciencia de que el problema existe y que no es solo tuyo sino de todos a los que afecta. En este caso el cliente o usuario es quien decide si quiere, o no, usar la característica. Un ejemplo simple puede ser en el sistema de autenticación de un sitio Web, cuando el usuario necesita enviar información confidencial a través de un formulario se le informa que no existe un mecanismo de cifrado o protección de esos datos en su tránsito por la red y entonces queda en sus manos decidir si seguir adelante o no. Esta puede ser una solución parcial, sobre todo cuando no es posible mitigar una determinada amenaza a corto plazo, por diferentes razones. Sin embargo hay que planificar bien el medio por el cual se le hace llegar de manera efectiva tal conocimiento al usuario final, no siempre informarlo mediante la documentación es la mejor opción, ya que no siempre la leen.

Mitigar el problema: Mitigar el problema o por lo menos tomar medidas que le hagan más difícil el camino a un adversario es la opción más obvia. Deben agotarse todas las posibilidades para eliminar los riesgos; que nunca quede un problema de seguridad pendiente por falta de gestión.

2.2.7.2 Documentar las medidas.

El conjunto de medidas de mitigación de riesgos y amenazas responde a una pregunta: ¿Cómo hacerle el trabajo más difícil a un atacante? La pregunta anterior no es absoluta en cuanto a eliminar totalmente la amenaza, porque los atacantes siempre buscarán nuevas formas de romper los mecanismos implementados. El proceso tiene dos pasos bien delimitados:

1. El primer paso es determinar qué técnicas pueden ayudar:
2. El segundo paso es determinar qué tecnologías son las más apropiadas.

Las técnicas no son lo mismo que las tecnologías. Por ejemplo, autenticación es una técnica de seguridad para restringir el acceso a determinados recursos, y Kerberos es una tecnología de autenticación específica. El Anexo 4 muestra una lista parcial con técnicas de mitigación por tipo de amenazas (Howard and LeBlanc 2002) que pueden ayudar a tener una visión más clara. Las amenazas deben ser tratadas en orden de prioridad según el grado de riesgo que presentan.

Una vez determinada las técnicas de mitigación para cada amenaza, se procede a actualizar el campo "**Mitigación conocida**" de la tabla de documentación de las amenazas, propuesta anteriormente. Luego, si se ha dibujado el Árbol de ataque asociado a la amenaza, opcionalmente puede incluirse un nivel más en el árbol, y por cada nodo hoja, que representa el medio o camino seguido para materializar la amenaza, se coloca un nuevo nodo conectado y con un color diferente que contiene la medida de mitigación. Esto hace más ilustrativo y es una manera fácil de que otros comprendan mejor modelado. Es importante aclarar que un mismo activo puede estar sujeto a más de una amenaza y para una misma amenaza puede existir más de una medida de mitigación.

2.2.7.3 Análisis de costos y beneficios.

Identificar las medidas que solucionan o disminuyen los riegos sobre un activo no significa mitigar la amenaza. Cuando se identifican las medidas entonces es necesario efectuar lo que se llama el análisis de costos y beneficios. Básicamente consiste en calcular el costo asociado con la inversión en medidas de protección sobre un activo y compararlo con el beneficio que aporta el activo en si. Si el costo supera los beneficios, entonces es bueno considerar si invertir es la mejor opción. No sólo ha de tener en cuenta el costo de cierta protección, sino también lo que puede suponer su implementación, mantenimiento, capacitación del personal, usabilidad del sistema, entre otros. En muchos casos existen soluciones sencillas pero efectivas para prevenir ciertas amenazas. El Líder del ES, en conjunto con el Arquitecto de seguridad y el trabajador Arquitecto del deben hacer una evaluación de las posibles soluciones y en base al costo tomar una decisión al respecto que luego se le informa a la gerencia del proyecto.

Cuando ya se ha realizado este análisis el Líder del ES presenta los resultados del análisis a la gerencia del proyecto, siempre teniendo en cuenta el costo contra el beneficio. Es necesario, sin embargo, hacerle entender a la gerencia que invertir en tecnología para mitigar los riegos muy pocas veces es un costo para la empresa sino una necesidad que en vez de pérdidas trae ganancias a corto o largo período. Los efectos por no invertir en seguridad pueden conllevar desde grandes pérdidas económicas hasta resquebrajar la confianza de los clientes en la empresa. Para aquellas amenazas que no se puedan mitigar inmediatamente, pueden, en ocasiones, encontrarse soluciones a medias, acotando todo el espectro de seguridad, en lo que a planeamientos, plataformas y estrategias se refiere. De esta forma se puede controlar todo un conjunto de vulnerabilidades y aunque no se logre

la seguridad total, esto significa un gran avance con respecto a no hacer nada (A.S.S. Borghello 2001).

2.2.8 Artefacto: Procedimiento de mitigación.

El Procedimiento de mitigación es el artefacto que genera el especialista de seguridad Diseñador de Procedimiento de mitigación y el objetivo fundamental que se persigue con este artefacto es mostrar de forma detallada los pasos necesarios para implementar una media de mitigación, por tanto, cada medida está asociada a un Procedimiento de mitigación y viceversa. Para comprender con más detalle lo que significa este artefacto se muestra a continuación un ejemplo para el portal de ventas por Internet.

Ejemplo 8. Elaboración de un Procedimiento de mitigación para una medida de mitigación.

El servidor que aloja el portal de ventas por Internet tiene como Sistema Operativo Windows 2003 Small Business Server y necesita implementar la siguiente medida de mitigación: **Configurar y activar el Cortafuegos del Sistema Operativo con los puertos 80 y 21 solamente abiertos.**
La anterior constituye la medida que permite mitigar los riesgos que implica poseer otros puertos abiertos de forma innecesaria mediante los cuales pueden introducirse programas maliciosos. Para implementar tal medida se necesita llevar a cabo una serie de pasos o procedimiento detallado. Por tanto es aquí donde entra a jugar un papel importante el artefacto Procedimiento de mitigación. Para este caso sería.

1. Ejecutar el programa Cortafuegos de Windows.
2. En la pestaña Excepciones presione el botón **Agregar Puerto.** Escriba en el campo Nombre la palabra "Puerto HTTP" y en el campo Número de puerto escriba el número 8080 y presione aceptar.
3. Repita el paso anterior para agregar el puerto 21.
4. Una vez terminado observará que los puertos aparecen marcados.
5. Luego en la pestaña General seleccionar la opción **Activar.** Con esto se garantiza que solo los puertos seleccionados previamente están abiertos.

Este conjunto de pasos constituye el procedimiento de mitigación para la medida planteada. Obsérvese que ese mismo procedimiento es reutilizable para desarrollos futuros de soluciones que

necesiten implementar esta medida, solo los puertos para ese caso quizá no sean los mismos, sin embargo solo es necesario cambiar los valores de los números. Esta es una muestra de cuan reutilizable resulta un Procedimiento de mitigación.

2.2.9 Artefacto: Guía de diseño de codificación segura.

En el Flujo de trabajo de diseño de un sistema de software se definen la plataforma de desarrollo y las diferentes tecnologías que en el Flujo de implementación le darán vida a los requerimientos funcionales y no funcionales identificados. Por tanto, el Líder del ES, al inicio del flujo, designa a un especialista, que tiene la tarea de elaborar un documento con las principales directrices de seguridad que regirán el proceso de implementación y codificación segura del sistema una vez seleccionada la plataforma y lenguaje de programación. Es posible que el proyecto tenga un estándar ya definido con los principales elementos de nomenclatura y documentación del código, sin embargo es frecuente que no existan en estas guías criterios de seguridad fuertes, que dicten las normas a seguir por lo desarrolladores para garantizar que el código fuente escrito además de legibilidad y facilidad de comprensión, sea confiable en cuanto a su funcionamiento y tratamiento de los datos que manipula. Si es así, el especialista designado tiene como primera tarea, realizar una revisión al estándar y encontrar aquellos puntos débiles que hacen mella en el buen desempeño del proceso. Corregir un error de seguridad en una aplicación luego de haberla implementado es 100 veces más caro que corregirla en la etapa de diseño (Arellano and Fontes 2004). A continuación se muestran algunos elementos que no deben faltar en una guía de codificación segura.

¿Qué incluir en la guía?

- La guía debe reflejar las restricciones sobre aquellas funciones, librerías o tipos de datos nativos del lenguaje o plataforma de implementación seleccionada, cuyo uso sea inseguro e inapropiado debido a causas conocidas.
- Procedimientos y buenas prácticas para el tratamiento de excepciones.
- Algoritmos de cifrado y encriptación de datos que no deben ser usados debido a vulnerabilidades conocidas en ellos.
- Técnicas seguras para el acceso a información almacenada en bases de datos.
- Directrices para el manejo de mensajes técnicos.
- Patrones de diseño de seguridad.

- Especificación de las herramientas de análisis estático de código cuyo uso es obligatorio para los desarrolladores, como parte de su trabajo sistemático.
- Políticas de validación de entradas, ya sean por parte de los usuarios o de otras fuentes.

La guía debe ser lo más completa posible y de conocimiento obligatorio de todos los desarrolladores. Muchas medidas de mitigación identificadas en el modelado y relacionadas con la escritura de código, deben estar reflejadas en la guía, sobre todo aquellas que se mitigan durante la escritura del código fuente.

Estructura del documento que refleja la guía.

Para un mejor entendimiento y comprensión de la guía por parte de los desarrolladores esta puede estructurase por secciones específicas.

- Reglas de cumplimiento obligatorio para la codificación: Las reglas de cumplimiento obligatorio no deben pasarse por alto en ningún momento. Estas son las reglas que definen las principales pautas para escribir código confiable. La violación de estas reglas pueden desencadenar vulnerabilidades en el código y un ataque contra el sistema que degrade los procesos que maneja.
- Recomendaciones: Las recomendaciones es la sección de la guía que refleja aquellas buenas prácticas para el desarrollo de código seguro, pero que sin embargo el hecho de no usarlas no significa que desencadenen un problema de seguridad en el sistema.
- Guía para el uso de herramientas de análisis de código: La guía debe especificar aquellas herramientas que necesitan ser usadas por los desarrolladores para analizar, durante su rutina diaria de escritura de código, si avanza conforme a lo establecido en el estándar. Por tanto esta sección debe proporcionar una guía rápida, donde se expliquen los pasos para hacer uso de la herramienta de manera eficiente y sacarle un mejor provecho.

La guía debe estar accesible a todos los desarrolladores en cualquier momento. Por lo que es recomendable tenerla publicada en algún lugar de uso masivo. La violación de las reglas de cumplimiento obligatorio implicará sanciones sobre quien incurra en la falta.

2.2.10 Actividad: Preparación del Equipo de Producto para la implementación.

Llegado a este punto del Flujo de análisis y diseño seguro, a partir de los resultados obtenidos, durante el modelado de amenazas y riesgos, del diseño de la guía de codificación y las medidas

necesarias para mitigar los riesgos. Se hace necesario tener un encuentro entre el ES y el EP. El encuentro puede realizarse en forma de seminario, taller o como mejor convenga a ambas partes. El Líder del ES tiene la responsabilidad de planificar las actividades a desarrollar durante la preparación. Es sumamente importante que todas las dudas sean aclaradas, los desarrolladores pueden hacer peguntas, el ES les da a conocer los principales lineamientos contenidos en la guía de codificación y la necesidad de cumplir con estos. Es importante hacer énfasis tanto a los problemas y debilidades que son inherentes a la plataforma de desarrollo, como a las potencialidades que ofrecen. Se muestran además las herramientas de seguridad usadas para análisis de código y se explica su funcionamiento. Si en el adiestramiento del flujo anterior se analizaban temas de seguridad y conceptos generales que permiten al EP adquirir cierta cultura de seguridad, el centro de atención de la preparación en este flujo estará enfocado a los lenguajes y plataformas específicas de desarrollo. Una vez terminado el periodo de preparación, se está listo para presentarle a la dirección del proyecto el Informe final de resultados y comenzar el próximo flujo de trabajo.

2.2.11 Artefacto: Informe final de resultados del diseño.

El Informe final de resultados, es el documento que resume el trabajo del ES durante el flujo actual. La responsabilidad de elaborar el informe recae sobre el Líder del ES. Una reunión del ES, permitirá dejar definidos los principales puntos a incluir en el documento a partir de los resultados obtenidos en la generación de los artefactos de seguridad. Existen varios elementos que no pueden faltar: resumen de las principales amenazas encontradas durante la realización del Modelo de Amenazas, principales activos que necesitan protección, resultados del análisis de costos y beneficios. También deben incluirse los resultados obtenidos durante las sesiones de Preparación del Equipo de Producto para la implementación. El informe termina con la presentación de los objetivos que se persiguen con el siguiente flujo de trabajo.

2.2.12 Resumen del Flujo de trabajo de Análisis y Diseño seguro.

Hasta aquí hemos visto las principales actividades que, en el Flujo de trabajo de diseño seguro, desarrolla el ES. El Modelo de Amenazas, el Diseño de la arquitectura de seguridad, la identificación de las medidas de mitigación de riesgos y la Guía de diseño de codificación segura son los resultados más importantes de este flujo. La modelación de amenazas llevado a cabo, da como resultado la identificación de los activos más valiosos que necesitan ser protegidos así como las amenazas que

sobre ellos se ciernen. Las medidas de mitigación de riesgos encontradas permitirán darle un tratamiento inmediato a las amenazas críticas. La preparación de un estándar de codificación seguro constituye una entrada fundamental para el proceso de codificación del siguiente flujo. La realización del análisis de costos y beneficios permite planificar tanto la distribución del presupuesto para la inversión en seguridad como las medidas alternativas a implementar. El encuentro final de este flujo entre el ES y todo el EP significa un momento importante al final del flujo, ya que prepara las condiciones necesarias para darle vida a los requerimientos de seguridad en Flujo de trabajo de implementación y codificación segura.

2.3 Flujo de trabajo de Implementación y Codificación segura.

La implementación comienza con los resultados del diseño y se implementa en términos de componentes, es decir, ficheros de código fuente, ejecutables y similares (Booch, Jacobson et al. 2001). El grueso de la arquitectura del sistema fue capturado y estructurado durante el diseño. Siendo el propósito de la implementación el desarrollar la arquitectura y el sistema como un todo.

2.3.1 Objetivos del Flujo de trabajo de Implementación y Codificación segura.

Los principales objetivos que se traza el EP, en cuanto a la funcionalidad del sistema, durante la implementación son (Booch, Jacobson et al. 2001):

- Planificar las integraciones de sistema necesarias en cada iteración.
- Distribuir los componentes ejecutables en los diferentes nodos del Diagrama de despliegue.
- Implementar las clases y los subsistemas encontrados durante el diseño. Esto significa generar el código fuente que da funcionalidad a la aplicación en su conjunto.
- Probar los componentes individualmente y a continuación integrarlos en uno o más ejecutables

Entonces ¿Cuál es la función y objetivos del ES durante la implementación del sistema? De manera resumida puede decirse que, garantizar la implementación correcta y confiable de las contramedidas identificadas durante el diseño, constituye el objetivo básico del ES durante la implementación. Para alcanzar estas metas es necesario velar porque se cumplan las reglas establecidas en la guía de codificación planteada en el diseño, aplicar herramientas de comprobación de seguridad y análisis de código, monitorear en una primera aproximación el funcionamiento de los protocolos y estándares

establecidos para la transmisión de datos, uso de bibliotecas de lenguajes óptimas, chequear la distribución e integración de los componentes de manera confiable.

Los pasos destinados a eliminar los errores de seguridad o a impedir que se incluyan desde el principio son de gran utilidad, ya que reducen considerablemente la probabilidad de que las vulnerabilidades de seguridad lleguen a la versión final del software que se entregará a los clientes.

Los resultados del modelado de amenazas ofrecen una orientación especialmente importante durante la fase de implementación. Los programadores deben asegurarse de que escriben correctamente el código para mitigar las amenazas de alta prioridad, mientras que los encargados de las pruebas en este momento deberán asegurarse de que se reducen perceptiblemente estas amenazas.

Para el ES el Flujo de trabajo de Implementación y Codificación segura constituye una etapa de revisiones, chequeo y pruebas iniciales.

2.3.2 Actividad: Asignar responsabilidades a los miembros del ES para la implementación.

Durante la implementación, algunos de los puestos de trabajo y responsabilidades asignadas a los miembros del ES variarán. Muchas de las responsabilidades estarán meramente centradas en tareas de verificación, integración de componentes y revisiones del código escrito. Esta tarea corresponde al Líder del ES, así como la elaboración del artefacto Informe final de resultados, para este flujo, el cual se detalla más adelante. A continuación se muestran cada uno de los roles de los especialistas de seguridad para la implementación.

2.3.3 Trabajador: Líder del ES.

El Líder del ES como máximo representante del mismo, continua desempeñando las responsabilidades y actividades ya especificadas en el punto anterior.

2.3.4 Trabajador: Arquitecto de seguridad.

El trabajador Arquitecto de Seguridad mantiene su nombre y la responsabilidad en este flujo estará en garantizar la aplicación de los diferentes protocolos y tecnologías de seguridad entre los componentes que forman la arquitectura. Por ejemplo el uso de protocolos seguros de red para la

transmisión de datos como SSL[21] o IPSec[22]. Verificar que no se violen las normas de relación entre capas definidas en el diseño. Seguir de cerca el funcionamiento de los componentes de la arquitectura además de verificar que los permisos sobre los datos correspondan con los roles de usuarios definidos anteriormente. Una tarea importante es la verificación constante de la correcta implementación de los paquetes de servicios identificados en el diseño de manera que garanticen seguridad en sus componentes para futuras reutilizaciones. El trabajo aquí es necesario hacerlo en conjunto con el trabajador Arquitecto de Sistema.

2.3.5 Trabajador: Revisor.

El trabajador Redactor, encargado de implementar la guía de codificación segura en el diseño, ahora desempeña el rol de Revisor y se encarga de velar por el cumplimiento de las reglas del estándar por parte de los desarrolladores. Es importante hacer notar que para esta tarea pueden ser designados varios especialistas, en dependencia de la cantidad de desarrolladores y el volumen del código fuente. Su trabajo va a realizarse en conjunto con los Ingenieros de componentes encargados de los Subsistemas de implementación y la generación del código fuente de los componentes.

2.3.6 Trabajador: Ingeniero de pruebas de seguridad.

La función específica de este trabajador es realizar las revisiones del código mediante herramientas automatizadas y revisiones manuales, aquí se deben asignar especialistas en programación con conocimientos importantes en seguridad del código que tenga habilidad para detectar posibles vulnerabilidades de seguridad. Estas revisiones constituyen un paso fundamental para eliminar las vulnerabilidades de seguridad del software durante el proceso de desarrollo.

---

[21] Secure Sockets Layer (SSL) y Transport Layer Security (TLS) -Seguridad de la Capa de Transporte-, su sucesor, son protocolos criptográficos que proporcionan comunicaciones seguras en Internet. Existen pequeñas diferencias entre SSL 3.0 y TLS 1.0, pero el protocolo permanece sustancialmente igual

[22] La seguridad del Protocolo de Internet (*Internet Protocol Security*, IPSec) es un marco de estándares abiertos para lograr comunicaciones privadas seguras a través de redes con el Protocolo de Internet (IP) mediante el uso de servicios de seguridad criptográfica.

A continuación se muestra la estructura básica del flujo en cuanto a trabajadores y actividades de seguridad que se realizan.

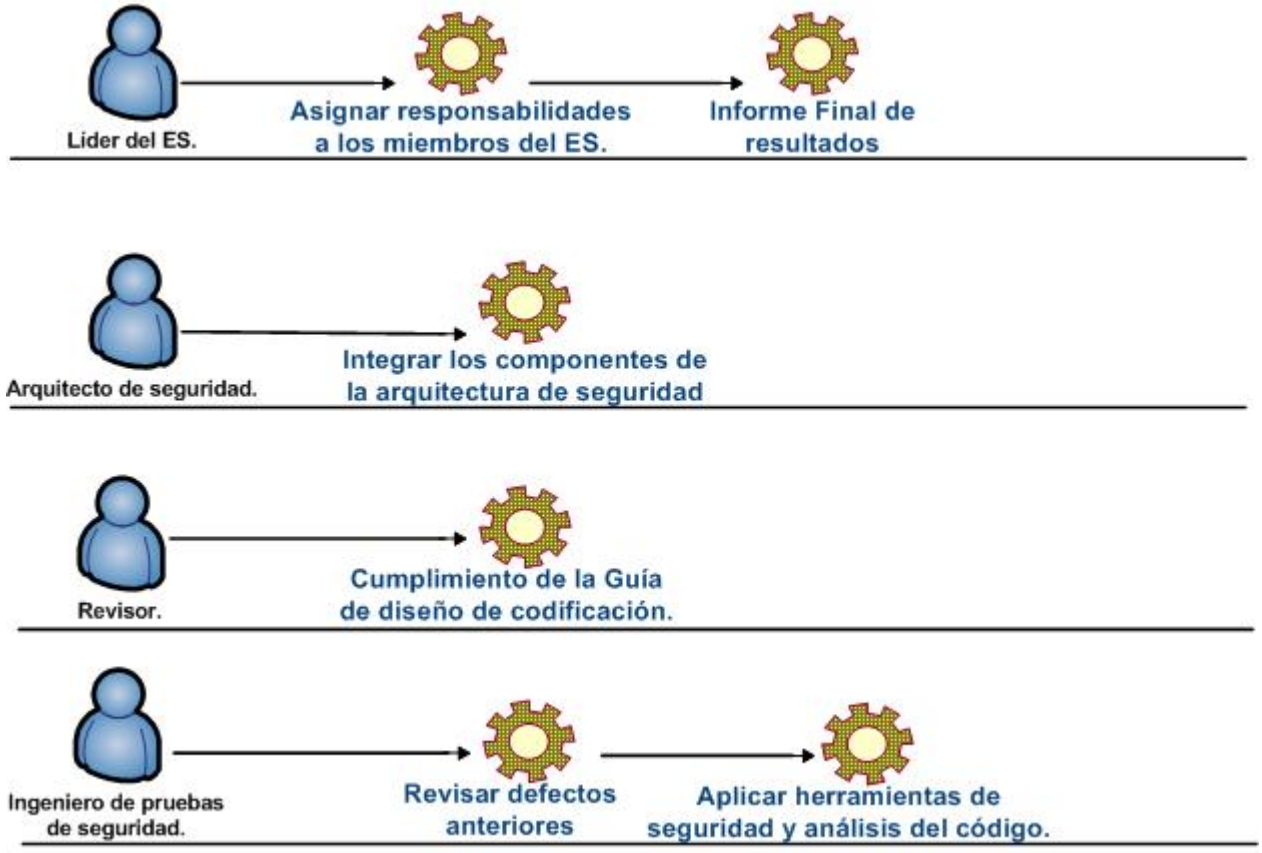


Figura 6. Trabajadores y artefactos de seguridad en la Implementación y Codificación del sistema.

2.3.7 Actividad: Revisar el cumplimiento de la Guía de diseño de codificación segura.

A continuación se presentan algunos elementos que deben tenerse en cuenta para esta actividad.

- El proceso de verificación del cumplimiento del estándar de codificación debe hacerse por partes y no de una sola vez. Dividir el proceso de revisión por reglas es una buena opción que permite centrar el trabajo en objetivos específicos.
- El uso de herramientas automatizadas que apoyen la tarea es un elemento imprescindible, por lo que el especialista Revisor debe dominar las herramientas y la mejor manera de sacarle

provecho. La generación de reportes organizados por tipos de reglas son funcionalidades importantes que tienen incorporadas la mayoría de estas herramientas.

- Elaborar un informe con los problemas encontrados que refleje la regla específica que se incumplió, el desarrollador que incurrió, el componente donde se encontró el problema, y el riesgo que implica la violación de la regla.

A continuación se muestra el formato que debe seguir una tabla típica de registro de incidencias durante el chequeo del estándar.

Desarrollador:	**Regla que se incumple:**	**Riesgo:**	**Componente:**
•			
•			
•			
Revisor:		**Fecha:**	

Tabla 7. Tabla de registro de incidencias durante el chequeo del estándar.

La tabla de registró permitirá llevar un control de las reglas que más se violan así como los desarrolladores que más incumplen en la aplicación de la Guía de diseño de codificación segura, lo cual permitirá realizar análisis individuales con los desarrolladores y por consiguiente tomar decisiones al respecto.

2.3.8 Actividad: Aplicar herramientas de comprobación de seguridad y análisis de código.

Si bien durante el proceso de codificación del sistema un especialista del ES tiene la tarea de velar por el cumplimiento de las directrices reflejadas en la Guía de diseño de codificación segura, existe otra tarea muy importante realizada por el Ingeniero de pruebas de seguridad: aplicar herramientas de comprobación de seguridad en los componentes. Las herramientas para comprobar la calidad del código escrito son muy diversas:

- Herramientas destinadas a realizar chequeos de uso indebido o excesivo de la memoria del ordenador debido a la mala escritura del código:

- Herramientas de confusión que permiten la entrada de juegos de datos falsos, para confundir o provocar errores de inconsistencia en los recursos manejados y de esta manera monitorear el comportamiento del sistema frente a estos errores.
- Herramientas de verificación y notificación de puntos débiles dentro del código que provocan la formación de “nudos” o “cuellos de botellas“.

Muchas son las labores de chequeo que pueden hacerse en esta etapa del proceso de desarrollo sin embargo las principales actividades estarán dirigidas al código que se escribe. La razón fundamental de realizar tales acciones en este flujo, está en identificar el mayor número de problemas posibles y tratar de resolverlos para enfrentar un flujo siguiente de pruebas que profundizará, en un ambiente lo más real posible y muchas veces hostil, la realización de pruebas de todo tipo.

2.3.9 Actividad: Revisar defectos anteriores.

En el proceso de implementación de un sistema de software frecuentemente se utilizan paquetes de servicios y componentes reutilizables de desarrollos anteriores. Muchas veces estos componentes no fueron diseñados de una manera adecuada ni sometidos a chequeos y pruebas de seguridad. La situación se torna más difícil aún, en aquellos casos en que fueron creados por terceros, donde ni siquiera tenemos conocimiento de su funcionamiento interno. Estas razones hacen de la implementación un momento ideal para replantearse la necesidad de realizar revisiones y chequeos a estos paquetes. Por tanto el ES y sus ingenieros de prueba deben tener identificados estos elementos y en el transcurso de la implementación realizar las pruebas correspondientes. No deben darse márgenes a errores que pueden ser difíciles de encontrar, pero cuyo efecto podría ser desastroso en un determinado momento.

2.3.10 Actividad: Integrar los componentes de la Arquitectura de Seguridad.

Un aspecto elemental de la implementación del sistema es la integración de los componentes que forman la arquitectura y su distribución en los diferentes nodos. El Arquitecto de Seguridad tiene la responsabilidad de garantizar la correcta implementación de los protocolos que garantizan el flujo seguro de la información a través de los componentes y capas que forman la arquitectura. La participación en conjunto con el Arquitecto de Sistema es fundamental.

2.3.11 Resumen del Flujo de trabajo de Implementación y Codificación segura.

Al término de la implementación, el Equipo de Seguridad garantiza, que el Equipo de Producto haya implementado correctamente las directrices de seguridad definidas en el diseño. Las actividades de revisión del código permitirán entrar a la etapa de pruebas con un sistema no solo completamente funcional sino con un sistema que de cumplimiento al Conjunto de Requisitos de Seguridad identificados.

2.4 Flujo de trabajo de Pruebas de seguridad.

Las pruebas que comenzaron en el flujo de trabajo anterior continúan durante el flujo actual. Sin embargo la profundidad de las revisiones se hace más evidente. La implementación proporciona como entrada al flujo de pruebas un producto funcional, que necesita probarse no solo en términos de su capacidad funcional sino desde el punto de vista de capacidad operativa confiable que permita decidir su salida o no al mercado.

2.4.1 Objetivos del Flujo de trabajo de Pruebas de seguridad.

No es objetivo durante el desarrollo de este modelo hacer una propuesta de cómo implementar un Plan de Pruebas completo. Existen buenas fuentes que estudian este tema en profundidad (Lewis 2005). Sin embargo, sí es objetivo del modelo ofrecer los principales aspectos que cubren la etapa de desarrollo de las pruebas de seguridad por parte del ES.

El primer aspecto que debe tenerse en cuenta es que el desarrollo de las pruebas de seguridad no debe estar separado del Plan de Pruebas elaborado por el EP.

El Líder del ES aconseja al Equipo de Producto sobre el ámbito de las pruebas que requiere el software y solicita una lista de recursos antes de la revisión. El EP proporciona al ES los recursos y la información necesarios para llevar a cabo la revisión final (Lipner and Howard 2005).

Con las pruebas de seguridad no se pretende detectar todas las vulnerabilidades de seguridad que quedan en el software, lo cual no sería factible ni posible, sino proporcionar al Equipo de Producto y a la administración superior de la organización una idea global del nivel de seguridad del software y la garantía de que pueda resistir ataques una vez que se haya entregado a los clientes.

Las comprobaciones de seguridad chequean que el probador no pueda suplantar la identidad de otro usuario, que no pueda manipular los datos a los cuales no tiene acceso, además de no poder acceder a ellos, que no pueda denegar el servicio a otros usuarios, que no tenga forma de borrar las

evidencias que lo hagan responsable de determinado acto, que no pueda alcanzar privilegios prohibidos mediante el uso indebido de la aplicación o servicios.

Es conveniente dividir el equipo en grupos de ingenieros que desarrollen pruebas específicas en determinadas áreas, por ejemplo:

- Designar un grupo para la realización de pruebas al código fuente y validación de entradas de datos.
- Designar un grupo para las pruebas de red y los protocolos de comunicación y transmisión de datos.
- Designar un grupo para la realización de pruebas de integración entre componentes.

Esta separación de tareas permite que los ingenieros concentren sus esfuerzos en encontrar la mayor cantidad de errores posibles en un área específica. Un principio fundamental que deben seguir los Ingenieros de Pruebas de Seguridad, es que deben ponerse en todo momento en el lugar de un atacante y actuar como tal. Es importante evitar contratar personas para realizar las pruebas, siempre que sea posible los mismos miembros del ES deben ser quienes las realicen.

2.4.2 Trabajador: Ingeniero de pruebas de seguridad.

El Ingeniero de Pruebas de Seguridad es el trabajador designado por el Líder del ES para llevar a cabo la realización de pruebas de este tipo. El trabajador diseña los casos de prueba según el área u objetivo a analizar.

2.4.3 Trabajador: Diseñador de Procedimiento de Prueba.

Algunos integrantes del Equipo de Seguridad pasarán en este flujo de trabajo a desempeñar el rol de Diseñador de Procedimiento de prueba. En el epígrafe 2.4.5 se explica en detalle qué es un Procedimiento de prueba. Es importante aclarar que este trabajador no es el encargado de diseñar los casos de prueba de seguridad y su posterior ejecución, sino que su función se limita a diseñar de forma detallada los pasos a seguir y entregarle este procedimiento al Ingeniero de pruebas de seguridad para que le incorpore los valores de entrada y salida durante el diseño del caso de prueba.
A continuación se muestra la estructura básica del flujo:

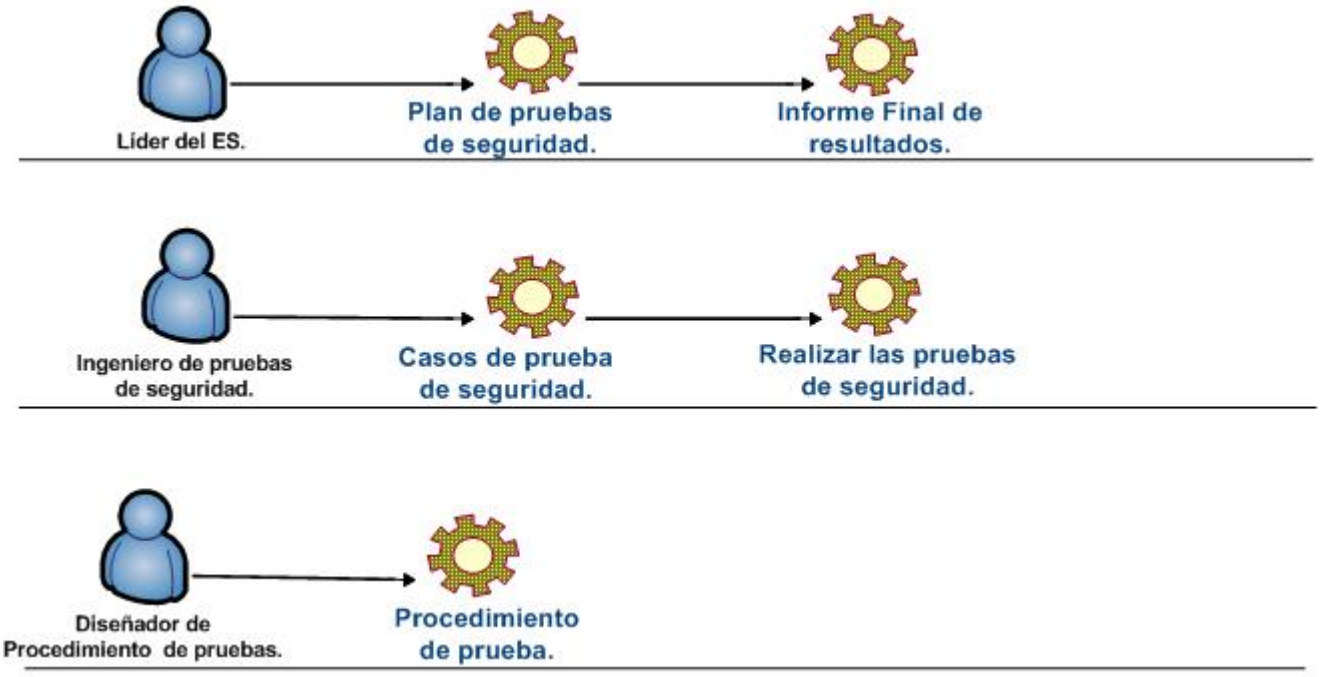


Figura 7. Trabajadores, actividades y artefactos de seguridad en las Pruebas de seguridad,

2.4.4 Artefacto: Caso de Prueba de Seguridad.

Los Casos de Pruebas de Seguridad constituyen el mecanismo utilizado por el Ingeniero de pruebas de seguridad para llevar adelante las acciones planificadas durante el Flujo de trabajo de pruebas. El diseño de los Casos de Prueba estará basado fundamentalmente en la información que proviene del modelado de amenazas. El formato o plantilla que debe llevar el diseño de un Caso de Prueba de Seguridad, debe ser estándar para todos los implicados en las pruebas. El Líder del ES debe analizar la plantilla adoptada por el EP y determinar si con los elementos que contiene es suficiente para adoptarla en el proceso de pruebas de seguridad. Existen elementos que no pueden faltar en el diseño del Caso de Prueba.

- Condiciones necesarias para realizar las pruebas.
- Tipo de prueba a realizar.
- Activo al que está dirigido la prueba.
- Procedimiento de prueba de seguridad.
- Resultados esperados con la realización de la prueba.

- Resultado actual alcanzado con la prueba.

A continuación se muestra una plantilla para la elaboración y diseño de Casos de Prueba de Seguridad (Lewis 2005).

<table>
<tr><td colspan="6">Caso de Prueba de Seguridad.</td></tr>
<tr><td>Fecha:</td><td colspan="2">___ ____ ____</td><td>Probado por:</td><td colspan="2"></td></tr>
<tr><td>Sistema:</td><td colspan="2"></td><td>ID Prueba:</td><td colspan="2"></td></tr>
<tr><td>Activo:</td><td colspan="2"></td><td>Tipo de Prueba:</td><td colspan="2">(Aplicación, Red, Hardware)</td></tr>
<tr><td colspan="6">Condiciones para la prueba:</td></tr>
<tr><td colspan="6"></td></tr>
<tr><td colspan="6">Procedimiento de prueba:</td></tr>
<tr><td colspan="6">•</td></tr>
<tr><td colspan="6">Resultados esperados:</td></tr>
<tr><td colspan="6">•</td></tr>
<tr><td colspan="4">Resultado actual:</td><td colspan="2"></td></tr>
<tr><td>Pasado:</td><td></td><td>Fallido:</td><td></td><td colspan="2"></td></tr>
<tr><td colspan="6">Comentarios:</td></tr>
<tr><td colspan="6">1.</td></tr>
</table>

Tabla 8. Plantilla de diseño de Casos de Prueba de Seguridad.

A un mismo Ingeniero de pruebas probablemente se le asignen varios casos, por lo que al finalizar cada prueba, debe actualizarse en el Caso de Prueba correspondiente el campo de resultado alcanzado. En caso de no obtener el resultado esperado para una prueba determinada no debe detener el proceso de revisión de los siguientes Casos de Prueba.

Al finalizar todas sus revisiones cada Ingeniero de pruebas debe elaborar un documento resumen con información de todos los Casos de Prueba y las incidencias detectadas.

2.4.5 Artefacto: Procedimiento de prueba de seguridad.

Un Procedimiento de prueba de seguridad define de forma detallada cada uno de los pasos para llevar a cabo uno o varios casos de prueba de seguridad. Teniendo en cuenta que un caso de prueba puede llevarse a cabo de forma manual o mediante el uso de herramientas automatizadas, también un Procedimiento de prueba puede ser una instrucción para un individuo sobre cómo a de realizar un caso de prueba manualmente, o puede ser una especificación de cómo interactuar con una herramienta de automatización de pruebas (Jacobson, Booch et al. 2001). Con el objetivo lograr un mejor entendimiento de este artefacto a continuación se muestra un ejemplo.

Ejemplo 9. Procedimiento de prueba de seguridad.

Se necesita un procedimiento de prueba para que el Ingeniero de Pruebas de seguridad lleve a cabo una prueba que verifica si el patrón de generación de contraseñas establecido para el sitio de compras por Internet funciona de manera correcta. Como el patrón de contraseñas abarca varios aspectos pueden diseñarse varios casos de prueba de seguridad donde todos usarían el mismo procedimiento.

Supongamos que el Caso de prueba de seguridad se llama "**Verificar si el sistema admite contraseñas de 5 caracteres".** Por tanto el procedimiento de pruebas sería el siguiente:

1. Iniciar sesión en una PC desde la que se pueda acceder al sitio de ventas.
2. abrir el navegador Web y escribir la dirección URL del sitio (www.ventasalmomentto.com).
3. iniciar sesión con un usuario y contraseña válidos.
4. Acceder a la funcionalidad de Cambiar contraseña actual.
5. Escribir en el campo **Nueva contraseña** la cadena **"nueva"**.
6. Presionar el botón **Cambiar.**

Entonces, si ahora se elabora otro Caso de prueba de seguridad cuyo nombre sea "**Verificar si el sistema admite contraseñas sin números".** Entonces el procedimiento sería el mismo que el anterior, solo que en el campo contraseña se incluiría la cadena **"Ab\cdegrs"** sin número en la secuencia. Esto demuestra que el mismo procedimiento es reutilizable en otros casos de pruebas similares solo que los valores de entrada y resultados serían específicos del caso de prueba donde se utilice. Esto demuestra que los procedimientos de pruebas de seguridad son completamente

reutilizables no solo en varios casos de prueba de un mismo proyecto sino en proyectos posteriores que diseñen casos de prueba con los mismos objetivos.

2.4.6 Actividad: Realizar pruebas de seguridad a partir del Modelo de Amenazas.

El papel que desempeña el Modelo de Amenazas, realizado en el Flujo de trabajo de Análisis y Diseño seguro, es determinante durante la etapa de pruebas de seguridad al sistema.

2. Las actividades destinadas a realizar las pruebas deben comenzar por los Casos de Pruebas de Seguridad que reflejan los activos de mayor impacto y sobre los cuales persisten las amenazas de mayor riego (DREAD) (Ver Anexos 2).
3. Las actividades de pruebas están orientadas a mitigar amenazas del tipo STRIDE (Ver Anexo 1).
4. Los Diagramas de Flujo de Datos realizados durante el diseño, aportan una serie de informaciones imprescindibles para realizar las pruebas de manera efectiva. Proveen una forma fácil de identificar puntos de entrada y frontera dentro del dominio de la aplicación así sus interfaces. Estos puntos son usados durante las pruebas para realizar chequeos de validación en estas áreas.
5. La descomposición del diseño y la arquitectura en componentes más manejables permitirá durante las pruebas realizar monitoreo y pruebas de ataques a dichos componentes para comprobar el correcto funcionamiento o no de los mecanismos de protección implementados en estos.
6. Los árboles de ataques permitirán validar la certeza o no del buen funcionamiento de las técnicas de mitigación de riesgos destinadas a contrarrestar los posibles caminos que puede seguir un atacante para explotar posibles vulnerabilidades del sistema.

Las actividades y jornadas de trabajo destinadas a la realización de pruebas de seguridad tienen como objetivo probar cada uno de los casos de prueba diseñados y evaluar los resultados obtenidos con los resultados esperados. Al final de la actividad, para cada caso probado debe registrarse algunas observaciones o comentarios presentes durante el proceso.

2.4.7 Artefacto: Informe de resultados de las pruebas de seguridad.

Las pruebas de seguridad terminan según el programa establecido en el cronograma de pruebas del EP. Una vez terminada la etapa de comprobación del software el Líder del ES tiene la tarea de preparar un informe final a partir de las incidencias encontradas por los especialistas de seguridad. El

informe debe ser preciso en cuanto la información que transmite. Dejando claro tanto el total de pruebas que se realizaron como los casos que no pasaron la revisión de forma satisfactoria. En base a los requisitos contenidos en el documento CRS, debe evaluarse el cumplimiento o no de los mismos. Si algunos de los requisitos no fueron garantizados el Líder del ES debe hacer ver su preocupación en el informe. Una vez conformado el documento se programa una reunión en conjunto entre la gerencia del EP y todo el ES. En esta reunión se presenta el Informe de resultado de las pruebas de seguridad y el Líder del ES rinde cuentas de los resultados reflejados en el documento. Con el Conjunto de Requisitos de Seguridad en la mesa, el encuentro debe girar alrededor de la siguiente pregunta: Desde el punto de vista de la seguridad, ¿está el software preparado para los clientes? El solo incumplimiento de al menos un requisito de seguridad capturado en el CRS, es una razón suficiente para que el sistema no se libere y entregue a los clientes. Esto no significa que todas las vulnerabilidades deben ser encontradas, se ha planteado anteriormente que garantizar un software completamente seguro es una tarea imposible, porque incluso, aunque se lograra en un momento determinado, las soluciones o mecanismos de seguridad usados en ese caso, llegará el día que también son vulnerables.

Conclusiones del capítulo.

En este capítulo se desarrolló la propuesta de un modelo que permite integrar la gestión de la seguridad al Proceso de desarrollo de software en la UCI, de una manera controlada y continua hasta la finalización de las pruebas realizadas al sistema. El modelo quedó estructurado en cuatro flujos de trabajo fundamentales que, básicamente, siguen los cuatro de los flujos propuestos por el Proceso Unificado de Desarrollo. Todos los flujos tienen una estructura similar conformada por cinco elementos: Artefacto de Seguridad, Actividad de Gestión de Seguridad, Especialista en Seguridad, Trabajador y Artefacto de RUP. Los integrantes del Equipo de Seguridad participan en una serie de actividades durante todo el flujo que permiten generar los artefactos de seguridad que reflejan los principales resultados desde este punto de vista al finalizar cada etapa. La participación y trabajo en conjunto con los trabajadores de RUP, generan resultados concretos a partir de los Artefactos desarrollados por estos últimos los cuales constituyen entradas de información valiosas para cumplir con los objetivos propuestos en cada uno de los flujos.

El Flujo de trabajo de inicio y Captura de requisitos de seguridad constituyó el punto de partida de la Gestión de la Seguridad en el Proceso de desarrollo de software. Es el momento en que se integró el

Equipo de Seguridad del proyecto y que laborará durante las etapas siguientes. El Conjunto de Requisitos de Seguridad es el resultado más importante de este flujo, ya que a los pasos posteriores centrarán su actividad en garantizar su cumplimiento.

El Flujo de trabajo de Análisis y Diseño seguro recibió el artefacto Conjunto de Requisitos de Seguridad y centró sus labores en desarrollar la propuesta del Modelo de Amenazas como técnica de Ingeniería de Software que permite identificar aquellos puntos vulnerables de todos los activos críticos que pueden afectar la confiabilidad del sistema.

El Flujo de trabajo de Implementación y Codificación segura definió como objetivo fundamental implementar las contramedidas que permiten mitigar los riesgos de las amenazas recogidas en el Modelo de Amenazas.

En el Flujo de trabajo de Pruebas de Seguridad se definió como objetivo fundamental, la realización de las pruebas de seguridad necesarias que permitan comprobar la implementación correcta del diseño del sistema desde el punto de vista de una capacidad operativa confiable del mismo. Una conclusión importante del modelo propuesto en la realización de las pruebas de seguridad es que no se garantiza la salida de un sistema 100% seguro, sino, la entrega de un producto de software capaz de soportar ataques una vez puesto en un ambiente generalmente hostil, un producto que garantice una manera de fallar en determinados momentos de manera robusta.

3

CAPÍTULO.

3. PROPUESTA DE APLICACIÓN DEL MODELO EN LA PLANIFICACIÓN DE LA SEGURIDAD DEL MÓDULO REGISTRO MERCANTIL.

El proyecto Registros y Notarías (RN) perteneciente a la UCI, desarrolla en estos momentos la solución de software SAREN con el fin de automatizar los procesos asociados a los registros mercantiles e inmobiliarios de la República Bolivariana de Venezuela. Se desarrolla sobre dos plataformas diferentes (Java, .NET). Las soluciones de los diferentes módulos que se implementan sobre .NET (Módulo Registro Mercantil, Módulo Registro Inmobiliario, Módulo Administración Financiera) constituyen aplicaciones de escritorio que serán desplegadas en las oficinas venezolanas, no siendo así en el caso del Módulo de Servicio Autónomo que constituye una aplicación Web desarrollada sobre Java.

En las oficinas mercantiles se encontrará básicamente una aplicación que integra los módulos Mercantil y Administración Financiera. De la misma forma en las oficinas inmobiliarias se encontrará básicamente una aplicación que integra los módulos Inmobiliario y Administración y por último en el nivel central se encontrará la aplicación de Servicio Autónomo y otras.

3.1 Objetivos y alcance del capítulo.

Las soluciones que, en este caso, se implementan sobre .NET presentan una estructura de desarrollo común que permite gestionar la seguridad de forma similar, desde el punto de vista de las amenazas específicas que acechan a las aplicaciones de escritorio y a la plataforma .NET. Esto significa que las acciones que se llevan a cabo en este sentido para un módulo, serán válidas para los demás, en la mayoría de los casos, por lo que se ha establecido como objetivo fundamental, para el desarrollo de

este capítulo, realizar una propuesta de gestión de la seguridad para el módulo Registro Mercantil desde la identificación del CRS hasta el diseño de los Casos de prueba de seguridad, a partir del modelo propuesto. La planificación se centrará fundamentalmente en gestionar la seguridad de los activos lógicos, no constituyendo una prioridad para este trabajo el tema relacionado con la seguridad de los activos físicos.

3.2 Flujo de trabajo de Planificación Inicial y Captura de requisitos de seguridad.

La identificación de requisitos de seguridad es una tarea continua que comienza en este flujo, pero que no termina aquí. El artefacto CRS es uno solo y a medida que van surgiendo nuevos elementos se vuelve sobre él y se actualiza. Para identificar el CRS, al menos en un primer momento se realizan encuentros con el Analista de Sistema del módulo Registro Mercantil para determinar, a partir de los conocimientos adquiridos sobre los procesos de negocio que se llevan a cabo en los Registros Mercantiles, aquellas necesidades que, en materia de seguridad, son imprescindibles y deben tratarse.

Los artefactos generados por el Analista contenidos en el documento que refleja el Modelo de negocio, constituyen una fuente de información fundamental para generar el CRS.

3.2.1 Artefacto: Conjunto de Requisitos de Seguridad del Módulo Registro Mercantil.

A continuación se muestra el artefacto Conjunto de Requisitos de Seguridad.

Conjunto de Requisitos de Seguridad.			
GRUPO	**REQUISITO**	**CU ASOCIADO**	**ACTOR**
Confidencialidad	• La información de clientes manipulada por la aplicación en los procesos de gestión diaria del Registro Mercantil tales como Inscripción de Actos de comercio y agregados, acceso a expedientes y libros de los comerciantes inscritos, entre otros debe mantener su confidencialidad durante el tránsito por los canales de red.	•	•

	• El código fuente de la aplicación no debe ser conocido por terceros en ningún momento.	•	•
	• Si ocurre algún error en el funcionamiento interno de la aplicación no debe mostrarse información confidencial y técnica, tales como nombre de campos, atributos o tipos de datos.	•	•
	• El acceso a la información almacenada en la BD debe ser personalizado mediante permisos previamente establecidos y no mediante entradas de datos por los usuarios.	•	•
	• El acceso a la red interna solo puede ser admitido a personas y computadoras definidas previamente.	•	•
Integridad	• La información de clientes manipulada por la aplicación en los procesos de gestión diaria del Registro Mercantil, debe mantener su integridad durante el tránsito por los canales de red.	•	•
	• El funcionamiento interno de la aplicación, una vez desplegada en los Registros Mercantiles, no debe ser alterado por terceros para obtener algún provecho.	•	•
	• La integridad de los datos almacenados en la BD no puede verse afectada por la acción maliciosa de usuarios no autorizados ya sea mediante entradas de usuarios o cualquier otro medio.	•	•
Disponibilidad	• Los servicios prestados por la aplicación no pueden verse afectados por errores de funcionamiento interno de la misma.	•	•
	• EL sistema operativo de los servidores y estaciones de trabajo debe estar debidamente protegido, de forma que se garantice su disponibilidad para poder operar la aplicación y gestionar la información diaria almacenada en el SGBD.	•	•
	• La aplicación Registro Mercantil necesita, para su	•	•

	funcionamiento diario, la disponibilidad constante de la red interna destinada a este fin.		
	• Las actividades de mantenimiento de la Base de Datos no debe entorpecer o limitar el funcionamiento diario del sistema.	•	•
	• Los programas antivirus y cortafuegos deben estar debidamente actualizados y configurados de forma que garanticen la no existencia de intromisiones de programas malignos o adversarios, que atenten contra la disponibilidad de la información y demás antes.	•	•
Otros	•	•	•

Tabla 9. Conjunto de Requisitos de Seguridad del Módulo Registro Mercantil.

3.3 Flujo de trabajo de Análisis y Diseño seguro.

El objetivo fundamental que se persigue con el desarrollo de este flujo es realizar el Modelo de amenazas para los activos lógicos del módulo Registro Mercantil y proponer finalmente una serie de contramedidas y técnicas de mitigación de los riesgos para cada una de las amenazas encontradas y proponer una guía de seguridad para la codificación del sistema por parte de los desarrolladores.

3.3.1 Artefacto: Modelo de Amenazas para el Módulo Registro Mercantil.

El proceso de modelado de amenazas que se realizará transcurre por una serie de pasos ordenados que conllevarán a asignar prioridades a cada una de estas según el nivel de riesgo que impliquen. A continuación se detallan los pasos.

3.3.1.1 Alcance de la aplicación Registro Mercantil.

Esta tarea permitirá delimitar el alcance de la aplicación Registro Mercantil a partir de la descripción de los objetivos que se persiguen con el mismo, del contexto en que operará y los escenarios de uso esperados. Estos elementos se representarán mediante los diagramas propuestos, para cada caso, en el capítulo anterior.

1. Objetivos que se plantean con la automatización de los Registros Mercantiles.

Los Registros Mercantiles de la Republica Bolivariana de Venezuela tienen como función principal realizar las tareas de inscripción, legalización y centralización de todos los actos relativos a los comerciantes individuales y sociales en los términos previstos por las leyes mercantiles. En tal sentido, la inscripción de actos en el Registro Mercantil y su posterior publicación cuando por disposición de la ley ello sea requerido, crea una presunción incontrovertible de veracidad y hace plena prueba, por lo que puede ser opuesto a terceros sin ninguna limitación (Rodríguez and Martín 2006).

Por tanto el principal objetivo que se plantea con el desarrollo de la aplicación Registro Mercantil es llevar a un ambiente automatizado lo procesos antes descritos, de forma tal que se agilicen y controlen, según lo establecido por la ley, las tareas rutinarias que se llevan a cabo en los registros actualmente, relacionadas con Inscripción de Actos de Comercio y Agregados, Acceso Público a los Expedientes, y Sellado de los Libros de los Comerciantes Inscritos.

2. Diagrama de Contexto de alto nivel del sistema Registro Mercantil.

El Diagrama de contexto (Ver Figura 10), proporciona una vista de alto nivel de la interacción entre la aplicación que funcionará en los Registro Mercantiles y las principales entidades que interactúan con el sistema.

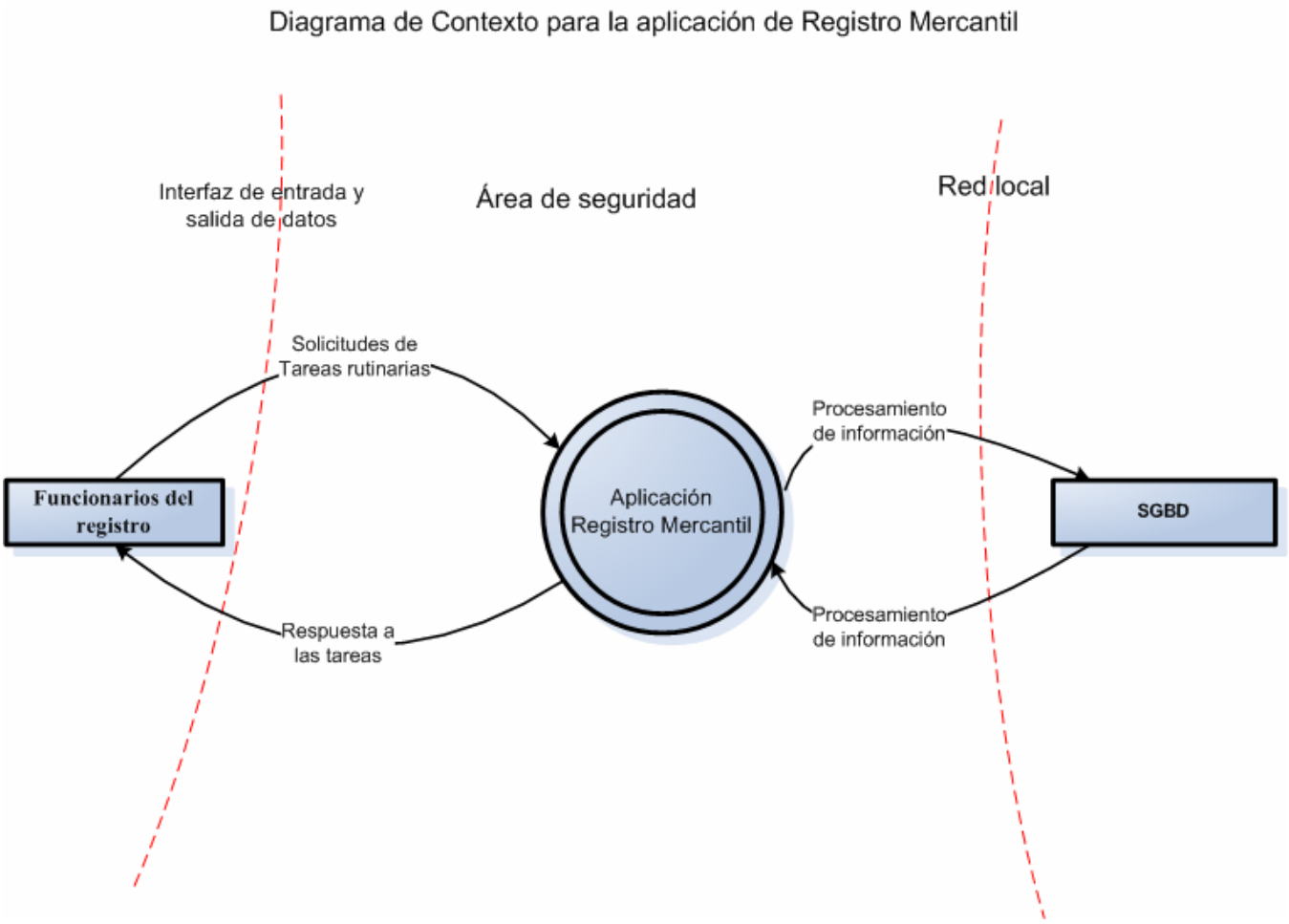


Figura 8. Diagrama de contexto de la aplicación Registro Mercantil.

El Diagrama representa a la aplicación como el proceso múltiple a partir del cual se desprenderán los procesos simples que permitirán desarrollar los DFDs particulares a los diferentes Casos de uso. En este nivel no es importante representar cada uno de los roles de usuarios que hacen uso de la aplicación sino que se resumen en la entidad representada. Los DFDs definirán cada uno de los roles específicamente. Se ha representado al área de seguridad en que debe operar el sistema una vez puesto en funcionamiento en cada uno de los Registros Mercantiles territoriales.

3. Identificación de los Escenarios de uso esperados.

Los escenarios de uso representan escenarios críticos desde el punto de vista de la seguridad donde se espera que opere el sistema Registro Mercantil. Estos escenarios incluyen desde las oficinas territoriales y su estructura por departamentos hasta los diferentes productos de software

seleccionados que forman parte de la solución. Los elementos reflejados tanto en el Diagrama de Contexto como en el documento que refleja la línea base de la arquitectura, constituyen una entrada de información importante para describir los escenarios de uso críticos en los que se espera funcione confiablemente el sistema, una vez desplegado en los Registros Mercantiles correspondientes.

Escenarios de uso esperados.	
Nro.	**Descripción.**
1.	• Las oficinas de los Registros Mercantiles están desplegadas por todo el país y la aplicación operará en dichas oficinas.
2.	• Cada oficina territorial está estructurada por departamentos. o Presentación. o Gestión Documental. o Revisión Legal. o Otorgamiento. o Archivo. o Reportes. o Impresión. Estos departamentos son el escenario esperado donde la aplicación funcionará en cuanto a local se refiere.
3.	• Las entradas de información al sistema se realiza por tres fronteras diferentes: o Entradas de datos por los funcionarios mediante la interfaz grafica de la aplicación. o Entrada de datos almacenados en la BD mediante canales de red. o Entrada de datos procedentes de archivos de datos, de configuración, impresora y escáner.
4.	• El sistema procesa información procedente del gestor de BD, mediante el uso de una red inalámbrica de tecnología 3Com interna o una red LAN en algunos casos, la cual es controlada por un especialista del Registro Mercantil.
5.	• El Gestor de Base de Datos es Oracle Standard Edition One 10.2g para el almacenamiento y administración del flujo de la información de la propia oficina y la proveniente del Centro de Datos.

6.	• El Sistema operativo en las estaciones de trabajo es Microsoft Windows XP en Español con Service Pack 2.
7.	• El Sistema operativo en el servidor de Base de Datos local es el Sistema operativo Microsoft Windows Small Business Server 2003 Standard Edition con Service Pack 1.
8.	• La actualización de la Aplicación Registro Mercantil se realiza solo mediante su servicio de actualización el cual permitirá el despliegue de las correcciones y las actualizaciones a las que sea sometido el software durante el funcionamiento y la puesta a punto del mismo, las cuales serán descargadas desde un repositorio compartido desde el centro de datos al servidor local y este a su vez las compartirá para que sean descargadas por las computadoras clientes.
9.	• El sistema hará uso de los periféricos scanner e impresora, esta última será conectada a un punto de la red del sistema.
10.	• Kit de Administración de Kaspersky en el servidor de Base de datos.
11.	• Antivirus Kaspersky para File Server versión 5.0.74 para la protección del servidor.
12.	• Kaspersky Antivirus para estaciones de trabajo versión 5.0.528 administrado por el Kit de Kaspersky del servidor local.

Tabla 10. Escenarios de uso esperados para la aplicación Registro Mercantil.

Los esfuerzos del Equipo de Seguridad se centraran en garantizar los criterios de seguridad para los escenarios de uso esperados. Aquellos elementos que queden fuera de esta área no forman parte de las responsabilidades del proceso de gestión de los especialistas que integran el ES.

3. Identificación de la Arquitectura de Seguridad.

Para una mejor comprensión de este paso se procederá a representar, inicialmente, la arquitectura del sistema completa incluyendo su conexión con el Centro de datos. Aunque el objetivo principal se limita a la arquitectura correspondiente a las oficinas mercantiles territoriales y la arquitectura de capas de la aplicación, en las cuales se encuentra distribuido el código fuente.

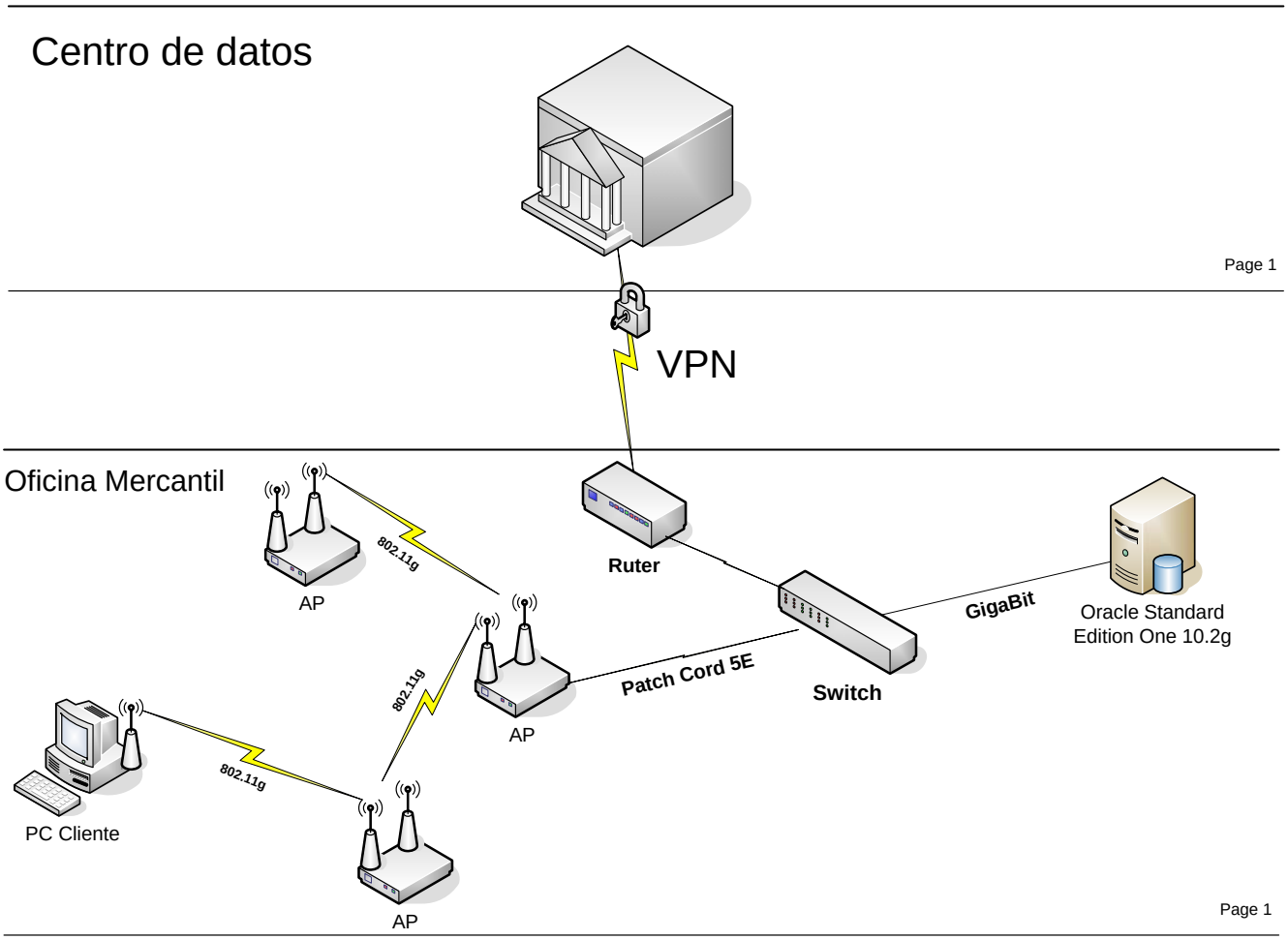


Figura 9. Arquitectura de red de alto nivel de la aplicación Registro Mercantil.

A continuación se muestra la estructura de capas establecida para el desarrollo de la aplicación Registro Mercantil.

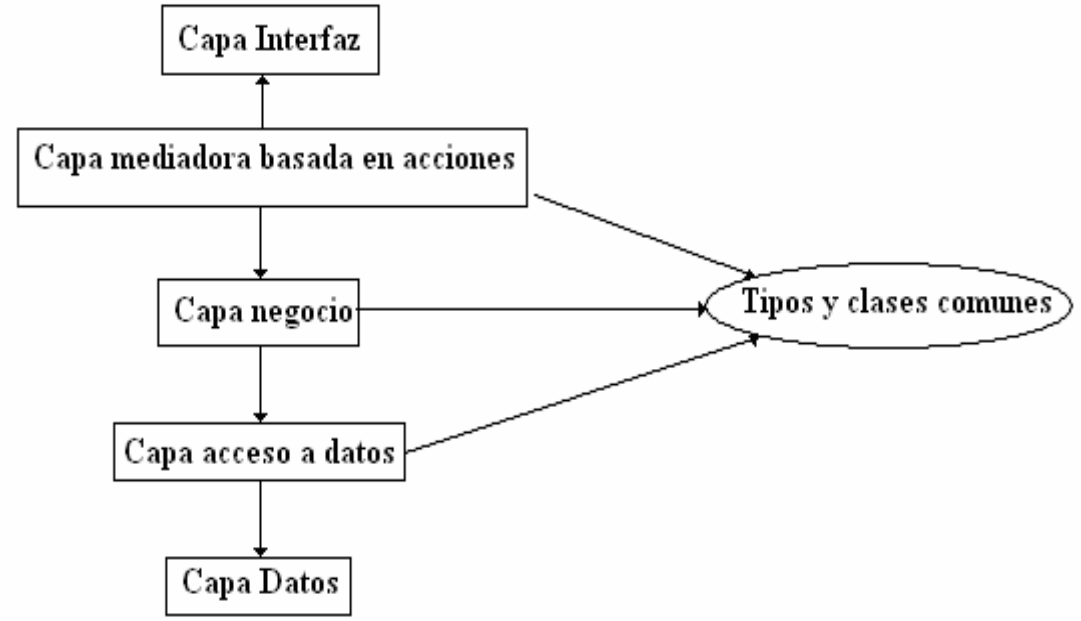


Figura 10. Distribucion del código en capas para la aplicación Registro Mercantil.

Los rectángulos representan las capas, la elipse representa una componente que contiene lo tipos y clases comunes a diferentes capas, el sentido de las flechas indica que la capa origen conoce la entidad que está en el destino de la misma.

Componentes:

- Capa de interfaz: La Capa de interfaz responde a un patrón estándar, todos los módulos tienen prácticamente la misma forma de trabajo con vistas a facilitar el trabajo y la estandarización del sistema completo. Su uso se basa en la explotación de las interfaces de usuarios con funcionalidades de captura o visualización de datos. Sin embargo el manejo de estas funcionalidades no radica en los mismos formularios de entrada o visualización de datos, sino en la Capa mediadora interfaz basada en acciones.
- Capa mediadora interfaz basada en acciones: Una acción es una clase que representa precisamente la ejecución de alguna tarea concreta. Estas tareas no deben ser excesivamente complejas. Las acciones básicamente definen un bloque de código reusable por varias operaciones sobre el sistema. Todo el funcionamiento de la Capa de Interfaz radica en las acciones asociadas a cada formulario.

- Capa lógica de negocio: El diseño de esta capa depende directamente del negocio específico al que se refiera. En ella se implementan las clases del negocio correspondiente y se gestionan los procesos relacionados con estas.
- Capa acceso a datos: Esta capa está compuesta por dos subcapas. La primera se corresponde con una capa compuesta por clases que constituyen factorías de las entidades del negocio que deben persistir. La segunda subcapa se genera con una herramienta y debe garantizar todo el manejo de la información que requiera el negocio del problema en cuestión y la conexión con la base de datos.
- Capa de datos: Esta capa corresponde a los almacenes de datos. A ella pertenece la base de datos disponible en el servidor de bases de datos local de cada oficina territorial donde esté instalado el sistema Registro Mercantil.

El canal de comunicación entre la Capa de acceso a datos y la Capa de datos se establece a través de la red interna del Registro Mercantil. Este canal es crítico para la Arquitectura de Seguridad, ya que a través del mismo fluyen todos los datos e información procesada por el sistema.

3.3.1.2 Descomposición del diseño.

1. Identificación de activos críticos.

Los activos importantes para el Modelo de Amenazas se seleccionarán a partir del artefacto Conjunto de Requisitos de Seguridad identificado anteriormente y de la Arquitectura de Seguridad.

Activos críticos.				
Nro.	**Nombre**	**Activo lógico**	**Activo físico**	**Impacto**
1.	• Código fuente.	X		Alto
2.	• Ficheros de código fuente.	X		Alto
3.	• Ficheros de configuración y de datos.	X		Alto
4.	• Información de clientes.	X		Alto
5.	• Red local inalámbrica 3Com.	X		Alto
6.	• Canal de transmisión de información.	X		Alto
7.	• Base de Datos Registro Mercantil.	X		Alto
8.	• SGBD Oracle Standard Edition One 10.2g.	X		Alto

9.	• Sistema Operativo Microsoft Windows Small Business Server 2003 Standard Edition Service Pack 1.	X		Alto
10.	• Sistema Operativo Microsoft Windows XP en Español con Service Pack 2.	X		Alto
11.	• Kaspersky Antivirus para estaciones de trabajo versión 5.0.528.	X		Alto
12.	• Kit de administración de Kaspersky del servidor local.	X		Alto
13.	• Kaspersky Antivirus para File Server versión 5.0.74.	X		Alto

Figura 11. Tabla de activos críticos para el Modelo de Amenazas.

2. Definición de roles de usuarios.

La aplicación Registro Mercantil está estructurada en 8 paquetes fundamentales los cuales van a corresponder con los diferentes pasos que se llevan a cabo durante las actividades de gestión diarias de las oficinas territoriales.

Sin embargo, por cuestiones de tiempo, para el desarrollo del resto de este capítulo solo se tomará el paquete correspondiente a Denominación Mercantil y por consiguiente El Diagrama de Casos de uso contenido en él. No obstante el proceso para los demás paquetes es similar al que se realizará con el seleccionado. A continuación se muestra el Diagrama de paquetes en su totalidad para lograr una mejor comprensión.

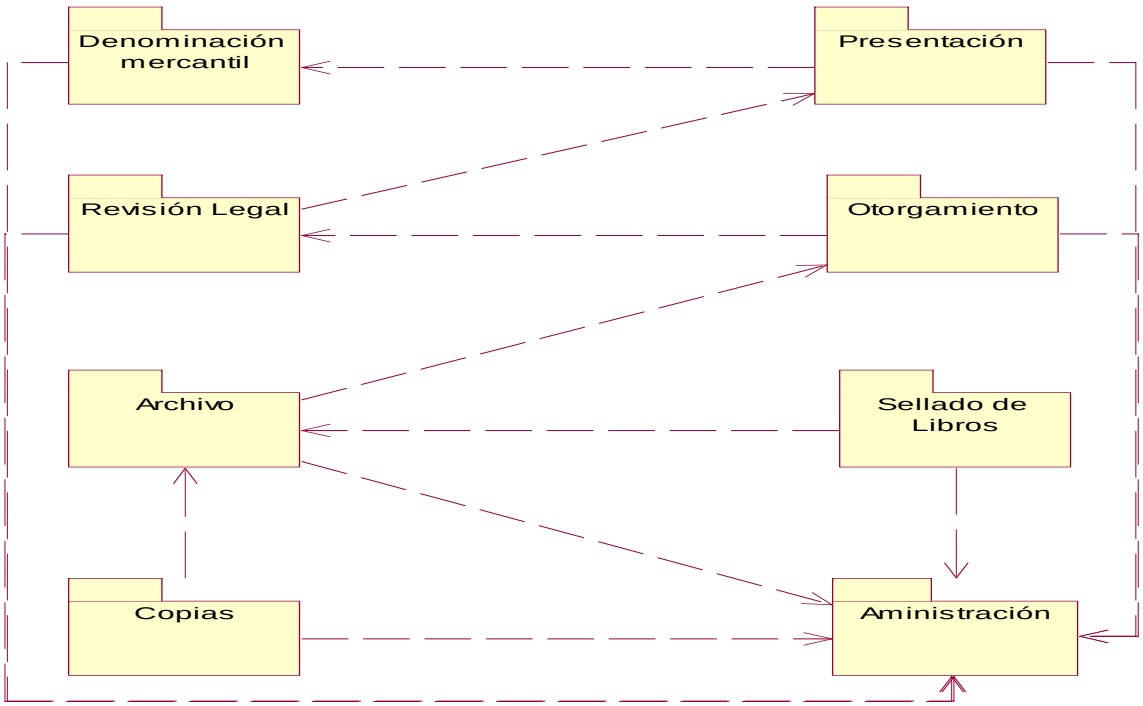


Figura 12. Diagrama de paquetes de la Aplicación Registro Mercantil.

Cada paquete contiene el Diagrama de Casos de uso que refleja la interacción de los actores con las diferentes funcionalidades ofrecidas por la aplicación correspondiente a cada estado en que se encuentra un trámite. Los actores son funcionarios del Registro Mercantil correspondiente, a los cuales se les asignan roles de usuarios. A continuación se muestran estos roles, el método de autenticación y el número de entidades aproximadas que existirán de cada rol.

Roles de usuarios del sistema.			
Nro.	**Nombre**	**Mecanismo de autenticación**	**Número de entidades**
1.	• Registrador.	• Formulario	• 1
2.	• Funcionario de Denominación.	• Formulario	• 1+
3.	• Funcionario de Presentación.	• Formulario	• 1+
4.	• Funcionario de Revisión Legal.	• Formulario	• 1+

5.	• Abogado Revisor.	• Formulario	• 1+
6.	• Funcionario de Otorgamiento.	• Formulario	• 1+
7.	• Funcionario de Archivo.	• Formulario	• 1+
8.	• Funcionario de Copias.	• Formulario	• 1+
9.	• Funcionario de Sellado.	• Formulario	• 1+

Tabla 11. Roles de los usuarios que interactúan con la aplicación Registro Mercantil.

El Diagrama de Casos de uso siguiente es el perteneciente al paquete Denominación Mercantil y con el cual se trabajará en adelante.

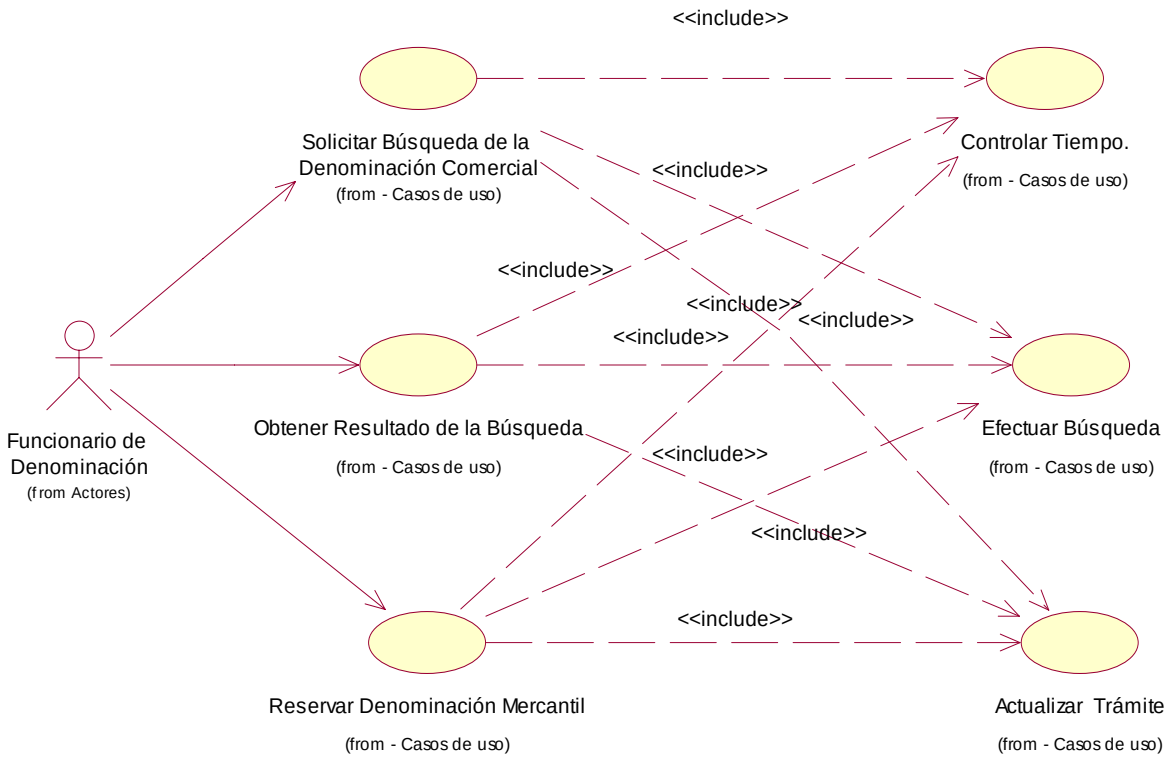


Figura 13. Diagrama de Casos de uso correspondiente al paquete Denominacion Mercantil.

A continuación se muestran los datos que maneja el Funcionario de Denominación durante sus actividades de gestión diarias.

3. Definición de los datos almacenados.

Datos Persistentes.		
Nombre	**Atributos (opcional)**	**Clasificación**
• Ficha de clientes.	•	Alto
• Solicitud de Denominación Mercantil.	•	Alto
• Trámite Constitutivo.	•	Alto
• Razón social.	•	Alto
• Planilla Única Bancaria.	•	Alto
• Recaudos.	•	Alto

Tabla 12. Datos manipulados por el rol Funcionario de Denominación.

4. Matriz de Control de Acceso.

Una vez identificados los datos que manipula el rol Funcionario de Denominación se procede a realizar la Matriz de Control de Acceso para cada uno de los datos. Esta técnica permitirá tener un controlar de los permisos y acciones que sobre los datos almacenados puede llevar acabo el rol, además de las condiciones que deben darse para poder tener tal permiso.

Dato:	Ficha de clientes.				
Rol de usuario	Insertar	Leer	Actualizar	Eliminar	Condición
Funcionario de Denominación Mercantil.	X	X			**I:** Si aún no está registrado cuando se inicie el proceso de Denominación Mercantil.
Dato:	Solicitud de Denominación Mercantil.				
Rol de usuario	Insertar	Leer	Actualizar	Eliminar	Condición
Funcionario de Denominación Mercantil.	X	X	X		**A:** Si el nombre de la Denominación Mercantil no ha sido buscado aún.
Dato:	Trámite Constitutivo.				
Rol de usuario	Insertar	Leer	Actualizar	Eliminar	Condición
Funcionario de Denominación Mercantil.	X	X	X		**A:** Solo si han sido

					gestionados los recaudos obligatorios para el paso en cuestión.
Dato:	Razón social.				
Rol de usuario	Insertar	Leer	Actualizar	Eliminar	Condición
Funcionario de Denominación Mercantil.	X	X	X		
Dato:	Planilla Única Bancaria.				
Rol de usuario	Insertar	Leer	Actualizar	Eliminar	Condición
Funcionario de Denominación Mercantil.	X	X			
Dato:	Recaudos.				
Rol de usuario	Insertar	Leer	Actualizar	Eliminar	Condición
Funcionario de Denominación Mercantil.	X	X	X		

Tabla 13. Matriz de Control de Acceso para cada uno de los datos.

14. Diagramas de Flujo de Datos.

Los siguientes DFDs permitirán definir tanto las fronteras de seguridad durante el flujo de eventos de los Caso de uso como los puntos de chequeos correspondientes.

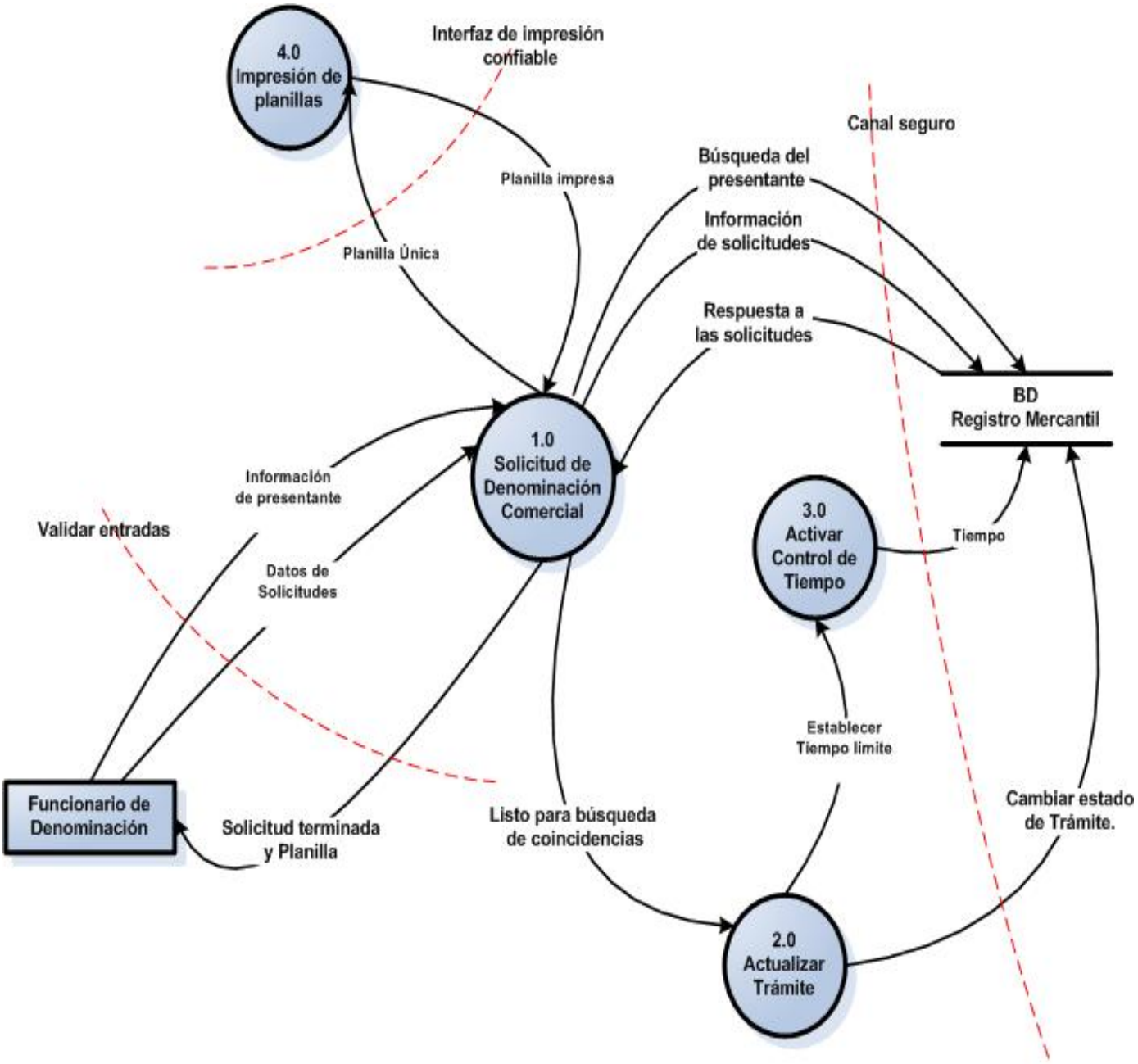


Figura 14. DFDs: Caso de uso Solicitar Búsqueda de Denominación Comercial.

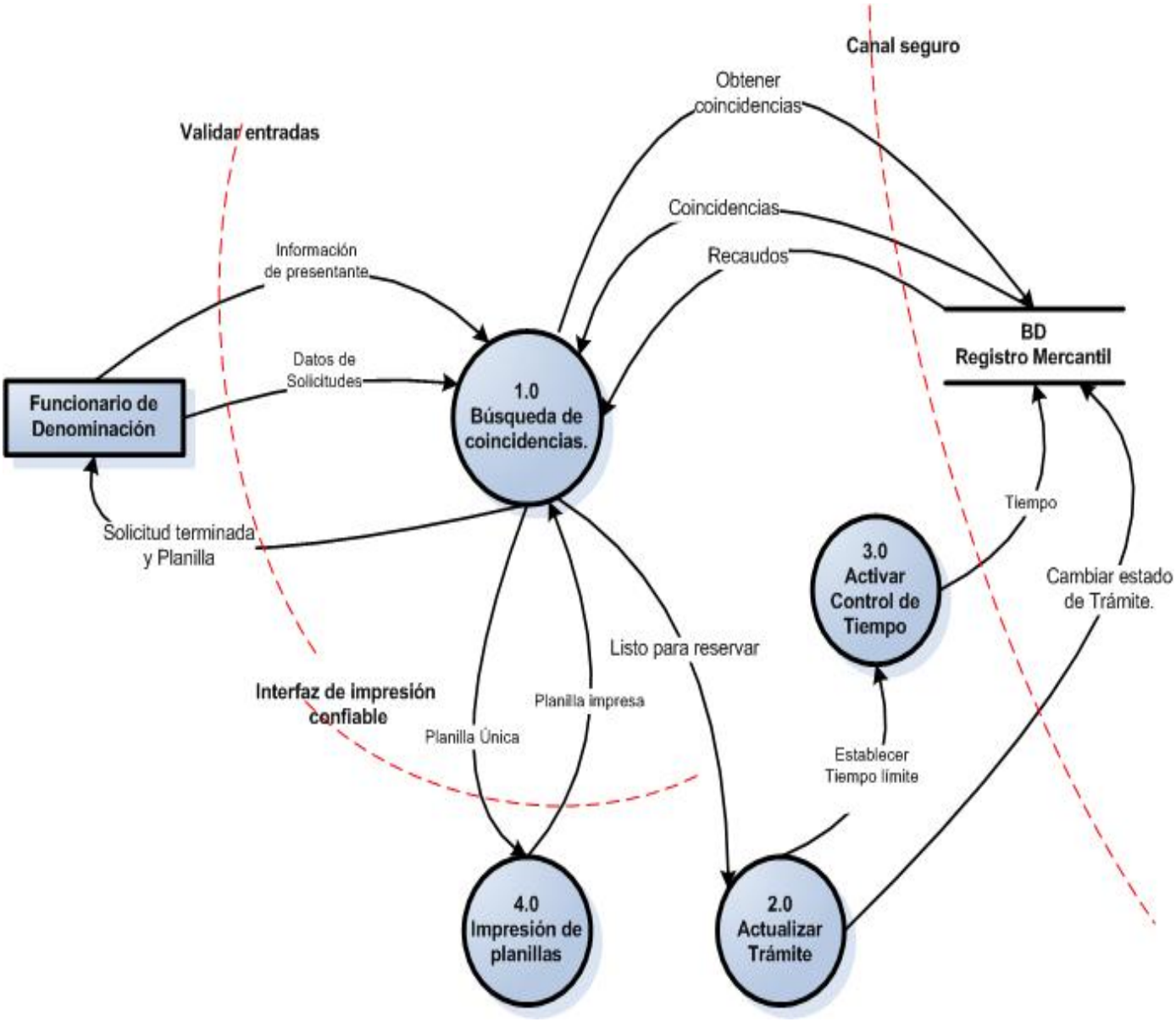


Figura 15. DFDs: Caso de uso Obtener Resultados de la Búsqueda de Denominación Comercial.

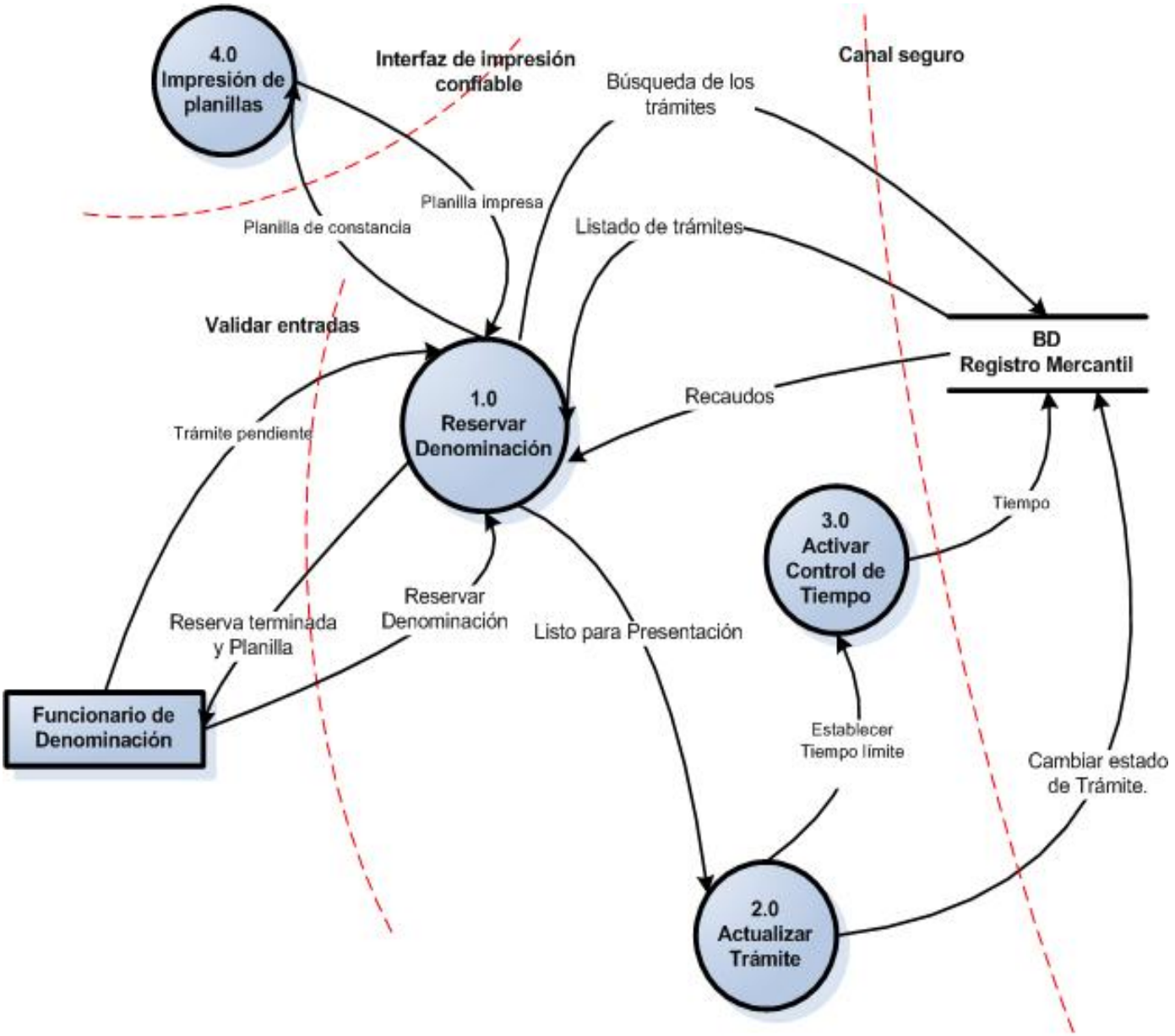


Figura 16. DFDs: Caso de uso Reservar Denominación Comercial.

A continuación se muestran de forma resumida el área de seguridad y los puntos de chequeos correspondientes.

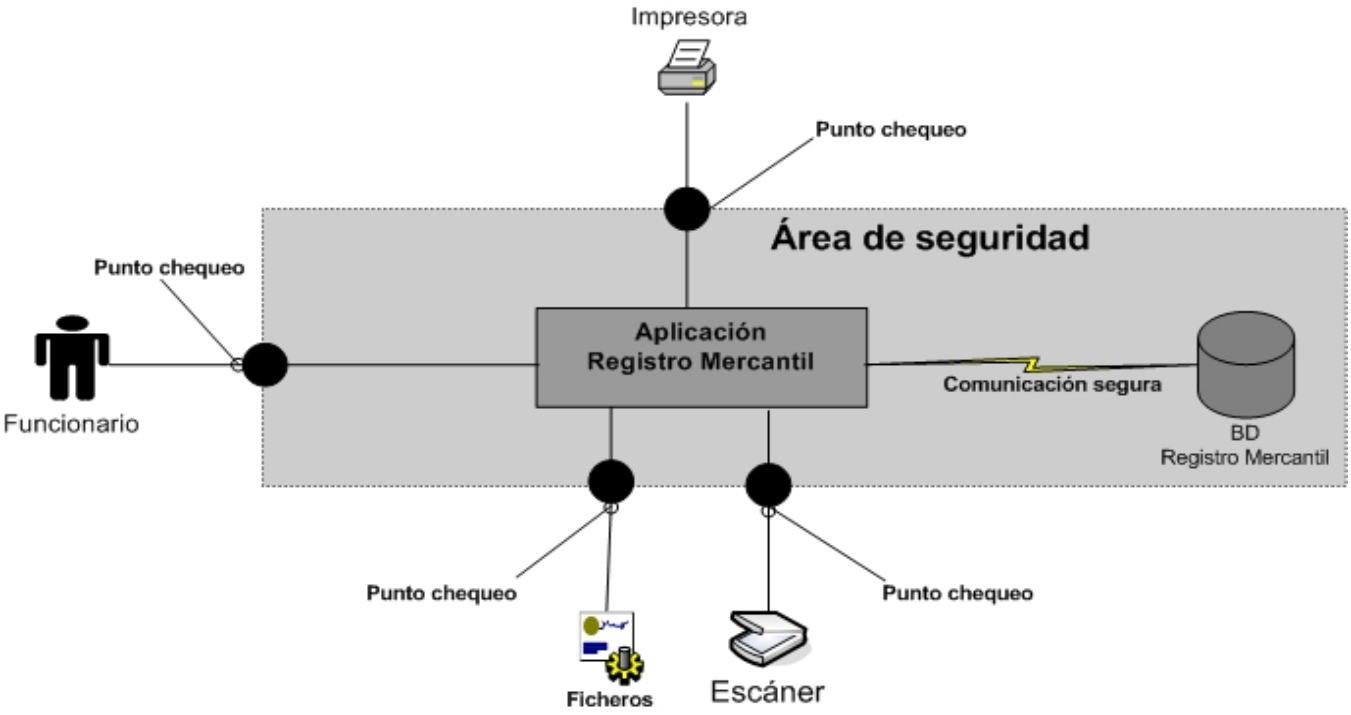


Figura 17. Área de seguridad de la aplicación Registro Mercantil.

15. Identificación de amenazas por activos.

Con la información obtenida hasta este momento se inicia el proceso de modelado de amenazas por activos. Algunas de las amenazas implican más de un activo.

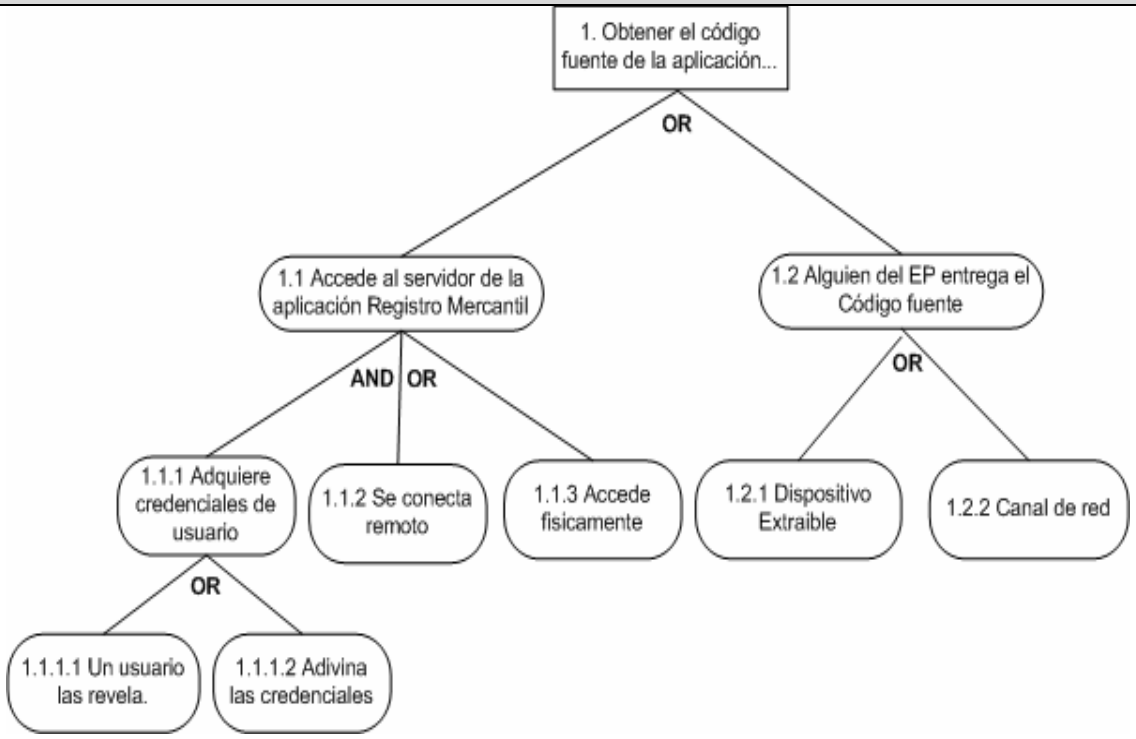
AMENAZA				
Nro.	**1.**	**Activo:**	• **Código fuente.** • **Ficheros de código fuente.**	
Nombre.		Persona no autorizada obtiene el código fuente y analiza su funcionamiento interno.		
Vulnerabilidad.			**Mitigación conocida.**	**Mitigable.**
• Entrada de medios extraíbles de almacenamiento en el área de desarrollo permitida.			• Prohibir la introducción de dispositivos de almacenamiento extraíbles en el entorno de desarrollo.	X

• Habilitados puertos USB, torre de CD y floppy tanto en las estaciones de trabajo de desarrollo como de las oficinas territoriales.	• Configuración del Setup del BIOS **(Ver Anexo 7. PM8).**	X
• Credenciales de acceso al servidor de la aplicación débiles.	Establecer un patrón de contraseñas robusto. **(Ver Anexo 7. PM2).**	X
• Entorno de desarrollo con conexión al exterior.	• Entorno de desarrollo aislado y sin conexión al exterior, de forma que no se filtre el código fuente por los canales de red.	X
• El código fuente contenido en los ensamblados no está protegido.	• Usar la herramienta Dotfuscator Community Edition proveída por la empresa **PreEmptive Solutions** y abalada por Microsoft. Debe ser comprada de forma independiente.	

Clasificación STRIDE.	• **S, I, E.**
Puntuación DREAD.	Riesgo = 1+2+1+3+3 = **10(Medio)**
Mitigación conocida.	•
Técnicas de ataque.	• Obtiene credenciales de usuario con permiso y accede de forma remota o física al servidor donde está instalada la aplicación en las oficinas o al servidor donde se almacenan los proyectos durante el desarrollo. • Un integrante del EP proporciona los proyectos que contienen los ensamblados ya sea a través de la red o por dispositivos de almacenamiento extraíbles. • Usando un editor de texto observa y analiza el código fuente en caso que posea los proyectos. • Si posee solo los ensamblados mediante la herramienta **Reflector**, desensambla los ficheros .dll o .exe y navega por cada uno de los

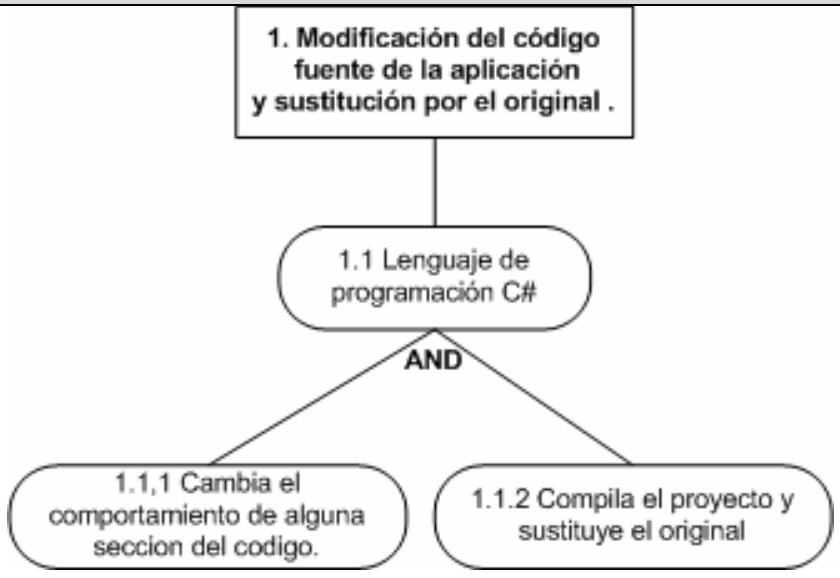
	métodos, clases y demás áreas que contienen el código fuente.

Árbol de ataque (Opcional).

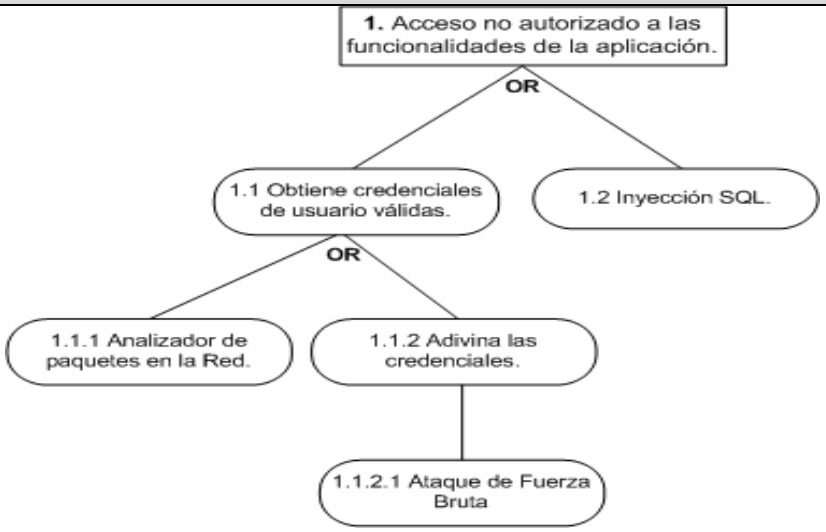


Árbol de ataque (Opcional).

AMENAZA

<table>
<tr><td>Nro.</td><td>2.</td><td>Activo:</td><td colspan="3">• Código fuente.
• Ficheros de código fuente.</td></tr>
<tr><td colspan="2">Nombre.</td><td colspan="4">Persona no autorizada modifica el código fuente de la aplicación y sustituye el ensamblado original por una copia con un comportamiento a su favor.</td></tr>
<tr><td colspan="3">Vulnerabilidad.</td><td colspan="2">Mitigación conocida.</td><td>Mitigable.</td></tr>
<tr><td colspan="3">• Los ensamblados no están firmados con un nombre fuerte.</td><td colspan="2">• Usar la herramienta sn.exe proveída con Visual Studio .NET (Ver Anexo 7. PM9).</td><td>X</td></tr>
<tr><td colspan="3">• Las claves para el firmado no están protegidas.</td><td colspan="2">• Claves usadas para el firmado deben ser controladas por una sola persona.</td><td>X</td></tr>
<tr><td colspan="2">Clasificación STRIDE.</td><td colspan="4">• S, T, R, I, D, E.</td></tr>
<tr><td colspan="2">Puntuación DREAD.</td><td colspan="4">Riesgo = 3+3+3+3+3 = 15(Alto)</td></tr>
<tr><td colspan="2">Técnicas de ataque.</td><td colspan="4">• Obtiene los proyectos (Ver árbol de ataque Amenaza 1).
• Usando el lenguaje C#, el adversario busca la sección del código fuente que aporte un valor importante y lo modifica a su favor, luego compila el proyecto y sustituye el original por la copia.</td></tr>
<tr><td colspan="6">Árbol de ataque (Opcional).</td></tr>
<tr><td colspan="6"> 
</td></tr>
</table>

<table>
<tr><th colspan="5">AMENAZA</th></tr>
<tr><td>Nro.</td><td>3.</td><td>Activo:</td><td colspan="2">• Información de clientes.
• Base de Datos Registro Mercantil.</td></tr>
<tr><td>Nombre.</td><td colspan="4">Persona no autorizada accede a las funcionalidades de la aplicación.</td></tr>
<tr><td colspan="2">Vulnerabilidad.</td><td colspan="2">Mitigación conocida.</td><td>Mitigable.</td></tr>
<tr><td colspan="2">• Mecanismo de autenticación insuficiente y contraseñas débiles.</td><td colspan="2">• Patrón de contraseñas robusto (Ver Anexo 7. PM2).
• Mecanismo de autenticación robusto (Ver Anexo 7. PM7).</td><td>X</td></tr>
<tr><td colspan="2">• Entrada de datos por campos de formularios no se validan.</td><td colspan="2">• Técnica contra sentencias SQL (Ver Anexo 7. PM3).</td><td>X</td></tr>
<tr><td colspan="2">• Canal de comunicación de red desprotegido.</td><td colspan="2">• Establecer directiva de seguridad IPSec para proteger el canal de comunicación de red (Ver Anexo 7. PM4).</td><td>X</td></tr>
<tr><td colspan="2">Clasificación STRIDE.</td><td colspan="3">• S, T, R, I, D, E.</td></tr>
<tr><td colspan="2">Puntuación DREAD.</td><td colspan="3">Riesgo = 3+3+1+3+3 = 13(Alto)</td></tr>
<tr><td colspan="2">Técnicas de ataque.</td><td colspan="3">• Ataque de SQL Injection: consiste en insertar determinadas cadenas en los campos de entrada de datos de los formularios de manera que se formen sentencias SQL válidas las cuales se envían al servidor como consultas y permiten engañar el proceso de verificación de credenciales.
• Coloca un programa SNIFFER en la red para capturar paquetes que contengan las credenciales de un usuario con permiso.
• Entrada constante de posibles cadenas en los campos de autenticación que permitan adivinar la contraseña (Fuerza bruta).</td></tr>
</table>

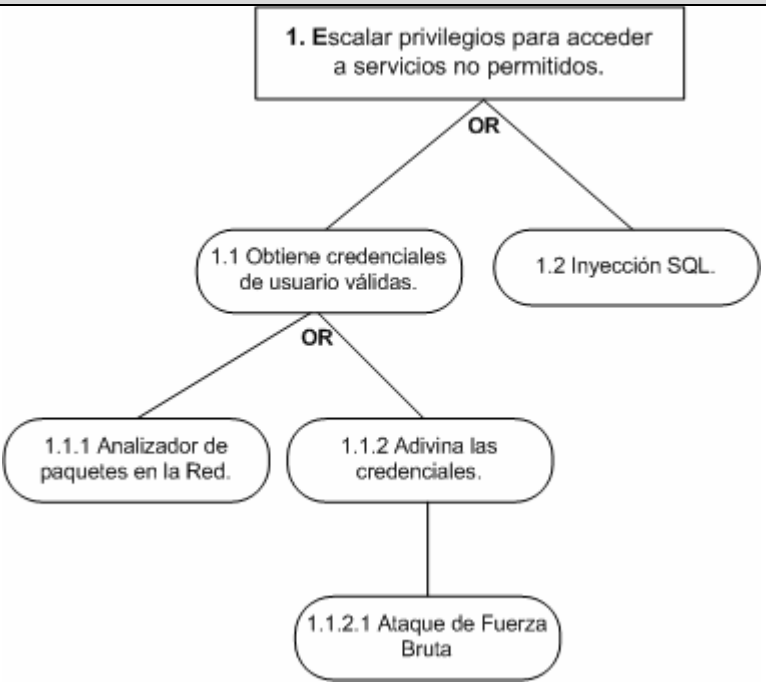
Árbol de ataque (Opcional).


AMENAZA			
Nro.	**4.**	**Activo:**	• **Código fuente.** • **Información de clientes.** • **Base de Datos Registro Mercantil.**
Nombre.	Persona no autorizada escala privilegios para acceder a servicios no permitidos.		

Vulnerabilidad.	**Mitigación conocida.**	**Mitigable.**
• Mecanismo de autenticación insuficiente y contraseñas débiles.	• Patrón de contraseñas robusto **(Ver Anexo 7. PM2).** • Mecanismo de autenticación fuerte **(Ver Anexo 7. PM7).**	X
• Canal de comunicación de red desprotegido.	• Establecer directiva de seguridad IPSec para proteger el canal de comunicación de red **(Ver Anexo 7. PM4).**	X
• No se validan las cadenas de entrada.	• Técnica contra sentencias SQL **(Ver Anexo 7. PM3).**	X
• Mala gestión de los roles y permisos.	• Documentar mediante la Matriz de Control de Acceso todos los roles y permisos	

	necesarios por cada usuario, tanto para manipular datos como para acceder a funcionalidades de la aplicación, y garantizar que solo son aplicados a cada usuario los roles contenidos en la Matriz. Todo permiso que no esté reflejado en la Matriz constituye una elevación de privilegio por parte del usuario.	X

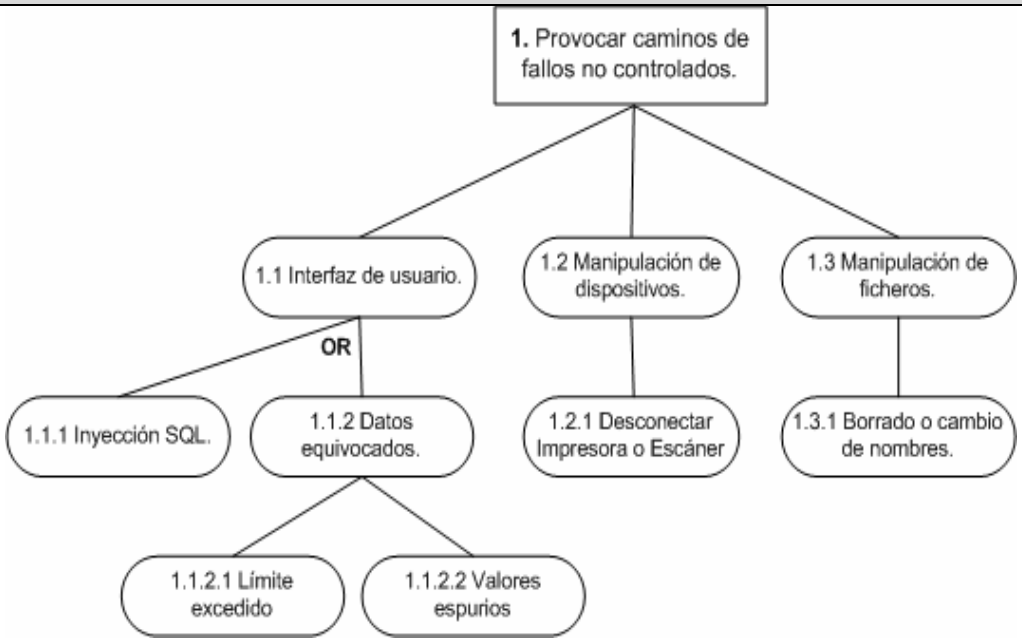
Clasificación STRIDE.	• **S, T, R, I, D, E.**
Puntuación DREAD.	Riesgo = 3+3+2+3+3 = **14(Alto)**
Técnicas de ataque.	• Ataque de SQL Injection, consiste en insertar determinadas cadenas en los campos de entrada de datos de los formularios de manera que se formen sentencias SQL válidas las cuales se envían al servidor como consultas permitiendo alcanzar privilegios no permitidos a nivel de BD o de aplicación. • Colocar un programa SNIFFER en la red para capturar paquetes que contengan las credenciales de otro usuario con privilegios superiores a los del atacante. • Probar el acceso con contraseñas comunes o entrada constante de posibles cadenas en los campos de autenticación que permitan adivinar la contraseña (Fuerza bruta) de otros usuarios que poseen privilegios no permitidos para el atacante.

Árbol de ataque (Opcional).


AMENAZA			
Nro.	**5.**	**Activo:**	• **Código fuente.** • **Información de clientes.** • **Base de Datos Registro Mercantil.**
Nombre.	Un adversario puede conducir la aplicación a ejecutar caminos de fallos no controlados.		

Vulnerabilidad.	**Mitigación conocida.**	**Mitigable.**
• Código fuente escrito con mala calidad.	• Ningún mensaje de error puede mostrar la información técnica contenida en la excepción generada **(Ver Anexo 7. PM10).**	X
	• El tratamiento de un error puede generar otro error, por tanto estos deben controlarse también dentro del **Catch**.	X
	• Utilizar recursos embebidos (ficheros de configuración, de datos, etc.).	X

	• Identificar todos los puntos de entrada y validarlos, ya sean entradas de usuario, impresoras, escáner, ficheros **(Ver Anexo 7. PM11).**	X
	• Validar el tamaño de los buffer y colecciones de datos para evitar desbordamiento de la pila.	X
	• Evitar búsquedas y bucles excesivamente largos, en su lugar establecer límites o extraer los datos a la memoria según se necesiten, para evitar consumo de memoria innecesario y disminución del rendimiento **(Ver Anexo 7. PM12).**	X
Clasificación STRIDE.	• **T, I, D.**	
Puntuación DREAD.	Riesgo = 2+2+2+2+3 = 11**(Medio)**	
Técnicas de ataque.	• Entrar de datos incorrectos (tipo equivocado, longitud mayor que la soportada, etc.). • Manipulación de ficheros de configuración y de datos usados por la aplicación. • Manipulación de dispositivos (impresora, escáner, etc.)	

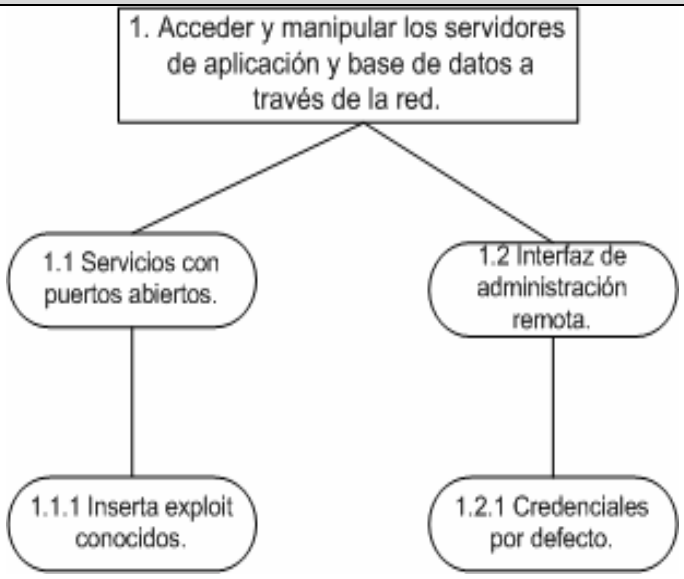
Árbol de ataque (Opcional).


AMENAZA				
Nro.	**6.**	**Activo:**	• **Base de Datos Registro Mercantil.** • **Información de clientes.** • **SGBD Oracle Standard Edition One 10.2g.** • **Sistema Operativo Microsoft Windows Small Business Server 2003 Standard Edition Service Pack 1.** • **Sistema Operativo Microsoft Windows XP en Español Service Pack 2.**	
Nombre.		Persona no autorizada accede y manipula los servidores de aplicación y base de datos a través de la red.		
Vulnerabilidad.			**Mitigación conocida.**	**Mitigada.**
• Aplicaciones y servicios instalados por defecto innecesariamente.			• Todas las instalaciones deben ser personalizadas para seleccionar solo los programas estrictamente necesarios.	X
• Configuraciones por defecto que no se cambian.			• No utilizar Esquemas Definidos por Defecto (Sample Schemas) en Oracle.	X

• Programas y servicios con contraseñas por defecto sin cambiar.	• Al finalizar cada instalación cambiar las credenciales por defecto que traen algunos servicios (SYSTEM, SYSMAN y DBSNMP).	X
• Parches de seguridad no instalados.	• Chequear periódicamente los parches de seguridad disponibles e instalarlos.	X
• Firewall no configurado y puertos abiertos innecesariamente.	• Configurar el Firewall y filtrar solo los puertos estrictamente necesarios **(Ver Anexo 7. PM5, PM6).**	X

Clasificación STRIDE.	• **S, T, R, I, D, E.**
Puntuación DREAD.	Riesgo = 3+2+2+3+3 = **13(Alto)**
Técnicas de ataque.	• Utilizar exploit conocidos para los servicios instalados a través de los puertos abiertos. • Manipular remoto el sistema y servicios usando credenciales por defecto.

Árbol de ataque (Opcional).

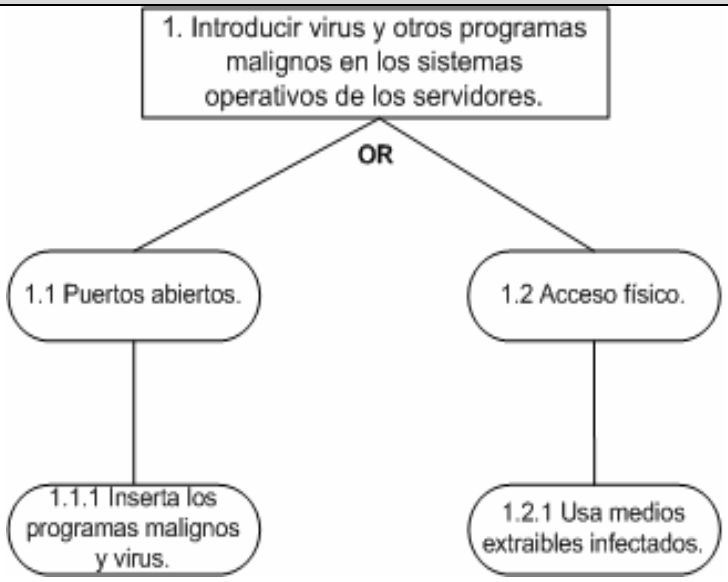


AMENAZA

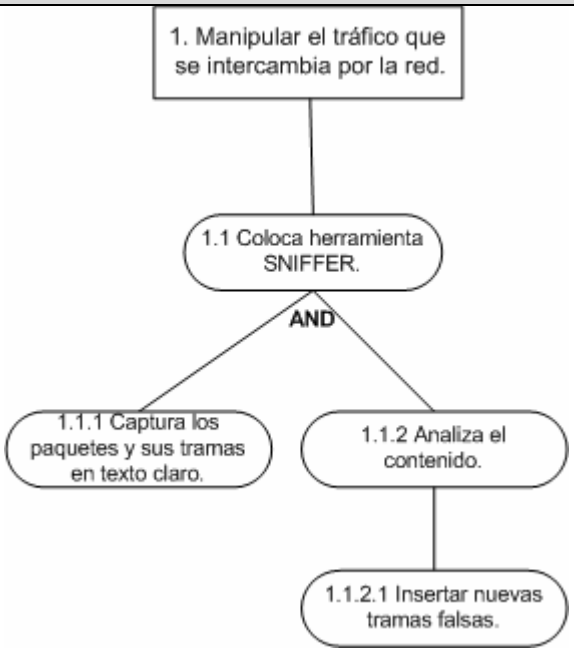
Nro.	**7.**	**Activo:**	• **Sistema Operativo Microsoft Windows XP en Español con Service Pack 2.** • **Sistema Operativo Microsoft Windows Small Business Server 2003 Standard Edition Service Pack 1.** • **Kaspersky Antivirus para File Server versión 5.0.74.** • **Kaspersky Antivirus para estaciones de trabajo versión 5.0.528.** • **Ficheros de código fuente.** • **SGBD Oracle Standard Edition One 10.2g.** • **Información de clientes.**
Nombre.		El adversario puede introducir virus y otros programas malignos en los sistemas operativos de los servidores.	

• **Vulnerabilidad.**	**Mitigación conocida.**	**Mitigable.**
• Antivirus Kaspersky no instalado.	• Instalar el Kit de administración de Kasperky en el servidor de BD, Kasperky para file Server y Kasperky para estaciones de trabajo en las PC clientes y establecer la estructura de administración ya definida.	X
• Antivirus desactualizado durante un tiempo prolongado.	• Actualización cada 3 horas con el centro de datos.	X
• Mala configuración de las tareas del antivirus.	• Configurar las tareas y servicios de actualización.	X
• Puertos abiertos innecesariamente.	• Configuración y activación del Firewall **(Ver Anexo 7. PM5, PM6).**	X
• Entradas para conectar medios extraíbles (USB, CD, FLOPPY) habilitadas constantemente y sin razón alguna.	• Todas las entradas que permitan conexión de medios extraíbles deben estar deshabilitadas por el **BIOS (Ver Anexo 7. PM8)**.	X

Clasificación STRIDE.	• **S, T, I, D, E.**

Puntuación DREAD.	Riesgo = 3+3+3+3+3 = **15 (Alto)**
Técnicas de ataque.	• Escanea puertos abiertos e introduce programas malignos a través de estos. • Accede físicamente e inicia sesión en la PC objetivo (Servidor de BD o Servidor de aplicación RM), conecta un medio extraíble infectado y propaga el virus.
Árbol de ataque (Opcional).	
	

AMENAZA				
Nro.	**8.**	**Activo:**	• **Canal de transmisión de información.** • **Base de Datos Registro Mercantil.** • **Información de clientes.**	
Nombre.	El adversario manipula el tráfico que se intercambia por la red entre la aplicación Registro Mercantil y el servidor local de Base de Datos Oracle.			
• **Vulnerabilidad.**			**Mitigación conocida.**	**Mitigable.**
• Directiva de seguridad IPSec no definida o deshabilitada.			• Establecer y habilitar directiva de seguridad IPSec **(Ver Anexo 7. PM4).**	X
Clasificación STRIDE.	• **S, T, I, D, E**			

Puntuación DREAD.	Riesgo = 3+1+2+3+3 = **12(Alto)**
Técnicas de ataque.	• El atacante usa una herramienta SNIFFER o analizador de paquetes para capturar un volumen considerable de estos cuyo contenido aporte un valor importante, analiza las tramas y se revela información sensible o incluso puede insertar paquetes falsos dentro del canal.
Árbol de ataque (Opcional).	
	

AMENAZA				
Nro.	**9.**	**Activo:**	• **Base de Datos Registro Mercantil.** • **SGBD Oracle Standard Edition One 10.2g.** • **Información de clientes.**	
Nombre.	Adversario manipula el diccionario de datos de Oracle.			
Vulnerabilidad.			**Mitigación conocida.**	**Mitigable.**
• La opción de **Data Dictionary Protection** está deshabilitada, lo que provoca que cualquier usuario con el privilegio de sistema DROP ANY			• Habilitar la opción de Data Dictionary Protection una vez instalado el SGBD mediante parámetro de configuración	

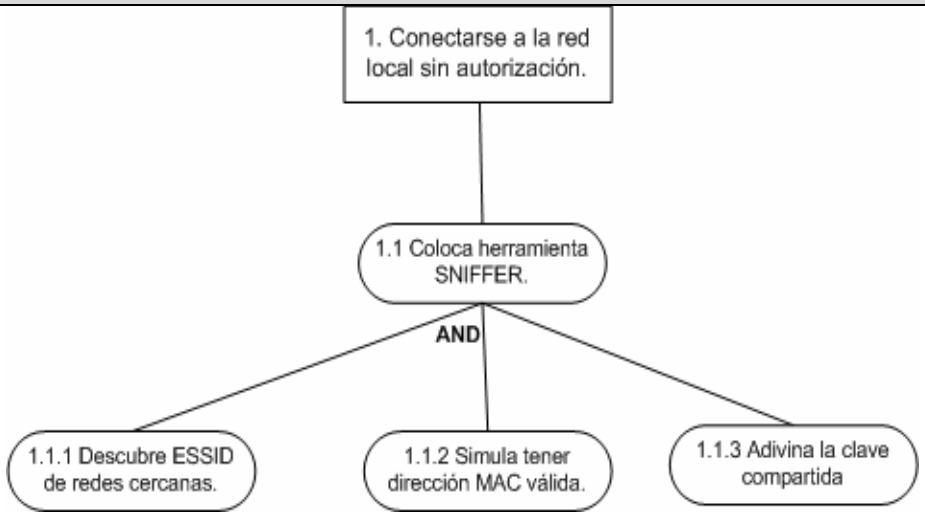
TABLE puede modificar el diccionario de oracle, causando así una lamentable pérdida de información al sistema.	O7_DICTIONARY_ACCESSIBILITY = FALSE. • Si un usuario necesita consultar el diccionario de datos, solo basta con asignarle el privilegio de sistema SELECT ANY DICTIONARY.	X

Clasificación STRIDE.	• **T, I, D.**
Puntuación DREAD.	Riesgo = 3+2+1+3+3 = **12(Alto)**
Técnicas de ataque.	Con el privilegio ANY SYSTEM PRIVILEGES modifica el diccionario de datos de oracle.
Árbol de ataque (Opcional).	

AMENAZA			
Nro.	**10.**	**Activo:**	• **Red local 3Com inalámbrica.** • **Canal de transmisión de información.**
Nombre.	El adversario puede conectarse a la red local sin autorización.		

Vulnerabilidad.	**Mitigación conocida.**	**Mitigable.**
• Mala gestión de la configuración de los AP.	• Establecer los mecanismos de seguridad a la configuración de los AP que forman parte de la red local inalámbrica de tecnología 3Com **(Ver Anexo 7. PM1).**	X

Clasificación STRIDE.	• **S, T, I, D, E.**
Puntuación DREAD.	Riesgo = 3+2+2+3+3 = **13(Alto)**
Técnicas de ataque.	• Utilizar un programa SNIFFERS que descubre los ESSID y redes disponibles en un radio determinado, comenzando la búsqueda a partir de nombres ESSID por defecto. • Simular tener una dirección MAC válida para conectarse. • Adivina las claves compartidas.

Árbol de ataque (Opcional).


AMENAZA				
Nro.	**11.**	**Activo:**	• **Información de clientes.**	
Nombre.	Un usuario realiza acciones sobre la información procesada por la aplicación y luego niega su autoría.			
• **Vulnerabilidad.**			• **Mitigación conocida.**	**Mitigable.**
• No existe implementado un mecanismo que controle cada una de las actividades que los usuarios realizan.			• Implementar un sistema de trazas donde se almacenen las acciones que van realizando los usuarios, la hora, fecha, etc., de forma que sea posible en cualquier momento realizar una auditoria sobre un hecho ocurrido.	X
Clasificación STRIDE.	• **T.**			
Puntuación DREAD.	Riesgo = 3+3+3+3+3 = **15(Alto)**			
Técnicas de ataque.	Simplemente un usuario realiza acciones permitidas para su rol, ya sea insertar, eliminar o actualizar información en la BD.			
Árbol de ataque (Opcional).				

3.3.2. Resultado del modelado de amenazas.

A continuación se muestra una tabla de resultados donde se agrupan las amenazas según el riesgo estimado que implican y consecuentemente la prioridad que se le debe dar en el proceso de mitigación.

Tabla de prioridad.			
Alta(12-15)		Media(8-11)	Baja(5-7)
2		1	
3		5	
4			
6			
7			
8			
9			
10			
11			
Total:	**9**	**2**	**0**

3.3.3 Artefacto: Guía de diseño de seguridad para el Módulo Registro Mercantil.

La guía de diseño de codificación que se propone no tiene como objetivo desarrollar una guía completa, donde se reflejen aspectos de nomenclatura de variables, ni estilos de codificación para el formato o estructura del código. Simplemente comprende algunos criterios de seguridad, que deben ser seguidos por lo desarrolladores en su proceso de implementación y construcción de los componentes ejecutables. El Anexo 6, Guía de diseño de codificación segura para el módulo Registro Mercantil, se muestran los elementos que la componen.

3.4 Flujo de trabajo de pruebas de seguridad.

La realización de pruebas de seguridad a la aplicación Registro Mercantil permitirá validar no solo que la aplicación funcione correctamente desde el punto de vista operativo, sino que las medidas de

mitigación de riesgos de seguridad para cada una de las amenazas documentadas son efectivas y garantizan la confidencialidad, la integridad y la disponibilidad de todos los activos críticos involucrados.

Las pruebas de seguridad se realizarán a partir del Modelo de Amenazas y centrarán su esfuerzo en verificar el cumplimiento de los requisitos de seguridad contemplados en el CRS.

A continuación se diseñarán una serie de casos de prueba de seguridad siguiendo la plantilla propuesta en el modelo del Capítulo 2.

3.4.1 Diseño de Casos de Prueba de Seguridad (CPS).

CPS1: Comprobar la ilegibilidad del código fuente de la aplicación desensamblando los ficheros que lo contienen, mediante la herramienta Reflector.

Caso de Prueba de Seguridad.			
Fecha:	___ ____ ____	Probado por:	Dusniel Horta Centeno.
Sistema:	Registro Mercantil.	ID Prueba:	**1.**
Activo:	• Código fuente. • Ficheros de código fuente.	Tipo de Prueba:	Aplicación. (Aplicación, Red, Hardware)
Condiciones para la prueba:			
• Tener instalada la herramienta Reflector. • Acceso a los ensamblados que contienen el código fuente.			
Datos de entrada/Acciones a realizar:			
• Se ejecuta la aplicación Reflector y se busca el ensamblado con el código fuente en el directorio correspondiente. Luego se intenta acceder a los diferentes métodos y clases que lo componen para ver el funcionamiento interno y encontrar posibles vulnerabilidades o información relevante en los comentarios.			
Resultados esperados:			
• Imposibilidad de ver el código fuente en texto claro.			
Resultado actual:			
Pasado:		Fallido:	
Comentarios:			

CPS2: Comprobar si los ensamblados usados por la aplicación Registro Mercantil están firmados con un nombre fuerte.

Caso de Prueba de Seguridad.			
Fecha:	___ ____ ____	Probado por:	Dusniel Horta Centeno.
Sistema:	Registro Mercantil.	ID Prueba:	**2.**
Activo:	• Código fuente. • Ficheros de código fuente.	Tipo de Prueba:	Aplicación (Aplicación, Red, Hardware)

Condiciones para la prueba:

- Tener instalado Visual Studio .NET 2003 y el lenguaje C# en una PC que no sea ninguna de las estaciones de trabajo donde se encuentra instalada la aplicación Registro Mercantil.
- Tener instalada la aplicación Registro Mercantil y funcionando plenamente.
- Tener acceso al proyecto del ensamblado que contiene el código fuente a modificar.
- Tener usuario y contraseña de una cuenta con permiso de escritura en el directorio donde se encuentra instalada la aplicación de Registro Mercantil.
- Tener usuario y contraseña de una cuenta con permiso para acceder a la aplicación y poder ejecutar las funcionalidades que usan el ensamblado seleccionado.

Datos de entrada/Acciones a realizar:

- Ejecutar Visual Studio .NET y abrir el proyecto del ensamblado.
- Buscar una sección de código donde se ejecute un bucle y modificar el mismo de tal forma que la condición de parada nunca se cumpla.
- Si el proyecto estaba firmado se genera una clave de nombre “clave_falsa.snk” mediante la herramienta **sn.exe** y se firma con la nueva clave el ensamblado.
- Compilar el proyecto con la nueva modificación y sustituirlo en el mismo directorio donde se encuentra la versión original usada por la aplicación.
- Ejecutar la aplicación Registro Mercantil y acceder a una funcionalidad que use dicho ensamblado.

Resultados esperados:

• La aplicación no debe ejecutar las operaciones contenidas en el ensamblado ya que el mismo fue modificado por una tercera persona y sustituido por el original sin el mecanismo de actualización establecido ni la clave original.

Resultado actual:				
Pasado:		Fallido:		
Comentarios:				

CPS3: Comprobar si la aplicación es susceptible a recibir entradas de sentencias SQL mediante los campos de entrada de datos de la interfaz de usuario.

Caso de Prueba de Seguridad.			
Fecha:	___ ____ ____	Probado por:	Dusniel Horta Centeno.
Sistema:	Registro Mercantil.	ID Prueba:	**3.**
Activo:	• Código fuente.	Tipo de Prueba:	Aplicación (Aplicación, Red, Hardware)

Condiciones para la prueba:
• Tener instalada la aplicación Registro Mercantil y funcionando plenamente. • Tener usuario y contraseña de una cuenta con permiso para acceder a la aplicación y poder ejecutar las funcionalidades que permiten entradas de datos por campos de formulario.
Datos de entrada/Acciones a realizar:
• Ejecutar la aplicación Registro Mercantil y acceder a una funcionalidad que permita entradas de datos por formulario. • Insertar el caracter comilla simple (') y enviar la solicitud. • Si la aplicación acepta el paso de dicho caracter, existe el riesgo de recibir ataques SQL Injection.
Resultados esperados:
• La aplicación debe mostrar alguna notificación que indique la existencia de caracteres no válidos en la entrada (mensaje, símbolo, cambio de color del campo, etc.).

Resultado actual:				
Pasado:		Fallido:		

Comentarios:

CPS4: Comprobar si la aplicación garantiza solo las entradas de datos por formulario que cumplan con el tipo que representa dicho campo y longitud máxima permitida.

Caso de Prueba de Seguridad.			
Fecha:	___ ____ ____	Probado por:	Dusniel Horta Centeno.
Sistema:	Registro Mercantil.	ID Prueba:	**4.**
Activo:	• Código fuente.	Tipo de Prueba:	Aplicación (Aplicación, Red, Hardware)
Condiciones para la prueba:			
• Tener instalada la aplicación Registro Mercantil y funcionando plenamente. • Tener usuario y contraseña de una cuenta con permiso para acceder a la aplicación y poder ejecutar las funcionalidades que permiten entradas de datos por campos de formulario.			
Datos de entrada/Acciones a realizar:			
• Ejecutar la aplicación Registro Mercantil y acceder a una funcionalidad que permita entradas de datos por formulario. • Insertar en un campo, cuyo valor debe ser un entero de tipo **int**, el número **(9999999999999999)** y enviar la solicitud. • Intentar insertar en un campo, cuyo valor debe ser un entero de tipo **int**, la cadena **("cadenafalsa")** y enviar la solicitud. • Si la aplicación acepta el paso de dichos valores, entonces la validación está fallando y existe el riesgo de generar tanto errores no controlados como truncamiento del valor.			
Resultados esperados:			
• La aplicación debe mostrar alguna notificación que indique la existencia de caracteres no válidos en la entrada (mensaje, símbolo, cambio de color del campo, etc.).			
Resultado actual:			
Pasado:		Fallido:	
Comentarios:			

CPS5: Comprobar que el servicio de autenticación de la aplicación limita el número de intentos fallidos y no acepta credenciales fáciles o comunes.

Caso de Prueba de Seguridad.			
Fecha:	___ ____ ____	Probado por:	Dusniel Horta Centeno.
Sistema:	Registro Mercantil.	ID Prueba:	**5.**
Activo:	• Código fuente.	Tipo de Prueba:	Aplicación (Aplicación, Red, Hardware)

Condiciones para la prueba:

- Tener instalada la aplicación Registro Mercantil y funcionando plenamente.
- Credenciales de inicio de sesión en una estación de trabajo donde esté instalada la aplicación.

Datos de entrada/Acciones a realizar:

- Iniciar sesión en estación de trabajo y ejecutar la aplicación para autenticar.
- Insertar usuario **"saren"** y contraseña **"registro"**. Enviar solicitud. Si falla seguir abajo.
- Insertar usuario **"registro"** y contraseña **"saren"**. Enviar solicitud. Si falla seguir abajo.
- Insertar usuario **"mercantil"** y contraseña **"registro"**. Enviar solicitud. Si falla seguir abajo.
- Insertar usuario **"rn"** y contraseña **"123"**. Enviar solicitud.

Resultados esperados:

- Tras 4 intentos fallidos, informar al usuario que el número de intentos permitidos ha terminado y se cancele el servicio.
- No permitir iniciar sesión con credenciales tan comunes y poco robustas.

Resultado actual:				
Pasado:		Fallido:		

Comentarios:

CPS6: Probar la eficiente implementación del patrón de creación de contraseñas para los usuarios que acceden a las funcionalidades de la aplicación Registro Mercantil.

Caso de Prueba de Seguridad.			
Fecha:	___ ____ ____	Probado por:	Dusniel Horta Centeno.
Sistema:	Registro Mercantil.	ID Prueba:	**6.**
Activo:	• Código fuente.	Tipo de Prueba:	Aplicación (Aplicación, Red, Hardware)

Condiciones para la prueba:

- Estación de trabajo y aplicación Registro Mercantil funcionando plenamente.
- Tener usuario y contraseña de una cuenta con permiso para acceder a la aplicación y poder ejecutar la acción de cambiar contraseña.

Datos de entrada/Acciones a realizar:

Ejecutar la aplicación Registro Mercantil y seleccionar la opción de cambiar contraseña.

- Insertar la cadena **"contrasenna".** Presionar el botón **Cambiar**.
- Insertar la cadena **"R3g1str0".** Presionar el botón **Cambiar**.
- Insertar la cadena **" ".** Presionar el botón **Cambiar**.
- Insertar la cadena **"R3g1sr+".** Presionar el botón **Cambiar**.
- Insertar la cadena **"R3g1str0+".** Presionar el botón **Cambiar**.

Si acepta cualquiera de las 4 primeras cadenas el patrón no está bien implementado. Sin embargo la última opción debe ser válida para el patrón y por tanto debe garantizar el cambio de la contraseña.

Resultados esperados:

- La aplicación debe mostrar alguna notificación que indique que no se está cumpliendo con el patrón establecido, para cada uno de los 4 primeros intentos realizados.

Resultado actual:				
Pasado:		Fallido:		

Comentarios:

CPS7: Comprobar la respuesta de la aplicación cuando se interrumpe el proceso de impresión de un documento.

Caso de Prueba de Seguridad.			
Fecha:	___ ____ ____	Probado por:	Dusniel Horta Centeno.
Sistema:	Registro Mercantil.	ID Prueba:	**7.**
Activo:	• Código fuente.	Tipo de Prueba:	Aplicación. (Aplicación, Red, Hardware)

Condiciones para la prueba:

- Estación de trabajo con la aplicación Registro Mercantil funcionando plenamente.
- Usuario y contraseña para iniciar sesión en la PC.
- Tener usuario y contraseña de una cuenta con permiso de Funcionario de Denominación de la aplicación.
- Impresora conectada a la PC o a la red y correctamente configurada.

Datos de entrada/Acciones a realizar:

- Iniciar sesión en la PC donde está instalada la aplicación.
- Ejecutar la aplicación Registro Mercantil y acceder a la opción de imprimir Planilla Única Bancaria para una solicitud de Denominación en proceso.
- Enviar la solicitud a la impresora.
- Mientras se imprime el documento, desconectar la impresora y observar la respuesta de la aplicación.

Resultados esperados:

- El fallo debe ser controlado y elegante, evitando inestabilidad en el sistema.
- La notificación que se le muestre al usuario no debe mostrar contenido técnico e interno del funcionamiento de la aplicación o de los datos que se enviaron a la impresora.

Resultado actual:				
Pasado:		Fallido:		

Comentarios:

CPS8: Comprobar que la estación de trabajo donde está instalada la aplicación Registro Mercantil no permite el acceso mediante credenciales por defecto o comunes del sistema operativo.

Caso de Prueba de Seguridad.			
Fecha:	___ ____ ____	Probado por:	Dusniel Horta Centeno.
Sistema:	Registro Mercantil.	ID Prueba:	**8.**
Activo:	• Sistema Operativo Microsoft Windows XP (español) SP2.	Tipo de Prueba:	Aplicación (Aplicación, Red, Hardware)
Condiciones para la prueba:			
• Estación de trabajo, donde está instalada la aplicación Registro Mercantil, disponible.			
Datos de entrada/Acciones a realizar:			
Encender la PC y esperar a que se muestre el cuadro de diálogo de autenticación. o Insertar la cadena **"administrador"** en el campo usuario y la cadena **"administrador"** en el campo contraseña. Presionar el botón **Aceptar**. o Insertar la cadena **"administrador"** en el campo usuario y la cadena **" "** en el campo contraseña. Presionar el botón **Aceptar**. o Insertar la cadena **"Invitado"** en el campo usuario y la cadena **"invitado"** en el campo contraseña. Presionar el botón **Aceptar**.			
Resultados esperados:			
• No se permite la entrada del usuario con ninguna de las opciones utilizadas.			
Resultado actual:			
Pasado:		Fallido:	
Comentarios:			

CPS9: Comprobar que los puertos lógicos abiertos en las estaciones de trabajo de la aplicación Registro Mercantil son solo los necesarios.

Caso de Prueba de Seguridad.			
Fecha:	___ ____ ____	Probado por:	Dusniel Horta Centeno.
Sistema:	Registro Mercantil.	ID Prueba:	**9.**

Activo:	• Sistema Operativo Microsoft Windows XP (español) SP2.	Tipo de Prueba:	Aplicación (Aplicación, Red, Hardware)
Condiciones para la prueba:			
• Estación de trabajo con la aplicación Registro Mercantil instalada y configurada correctamente similar al funcionamiento en un Registro Mercantil. • Herramienta SNIFFER **Nessus 3.0** para escaneo de red instalada en otra PC o en la propia estación de trabajo de la aplicación.			
Datos de entrada/Acciones a realizar:			
• Ejecutar Nessus 3.0 y establecerle la dirección IP de la estación objetivo para escanear los puertos que posee abierto. • Comenzar el proceso de escaneo mediante la opción **Start** y esperar la conclusión del escaneo. • Analizar el reporte que se genera.			
Resultados esperados:			
• Solo se reporten abiertos los puertos estrictamente necesarios, definidos previamente.			
Resultado actual:			
Pasado:		Fallido:	
Comentarios:			

CPS10: Comprobar que los puertos lógicos abiertos en el Servidor de Base de datos Oracle son solo los necesarios.

Caso de Prueba de Seguridad.			
Fecha:	___ ____ ____	Probado por:	Dusniel Horta Centeno.
Sistema:	Registro Mercantil.	ID Prueba:	**10.**
Activo:	• SO Microsoft Windows Small Business Server 2003 Standard Edition Service Pack 1.	Tipo de Prueba:	Aplicativo (Aplicación, Red, Hardware)
Condiciones para la prueba:			
• Sistema Operativo instalado en el Servidor de Base de datos.			

• Servidor de Base de datos Oracle instalado y funcionando plenamente. • Servidor conectado a la red interna. • Herramienta SNIFFER **Nessus 3.0** para escaneo de red instalada en el servidor.
Datos de entrada/Acciones a realizar:
• Ejecutar Nessus y establecerle la dirección IP del servidor de base de datos para escanear los puertos que posee abierto. • Comenzar el proceso de escaneo mediante la opción **Start** y una vez finalizado observar los resultados.
Resultados esperados:
• Solo se reportan abiertos los puertos estrictamente necesarios.

Resultado actual:				
Pasado:		Fallido:		
Comentarios:				

CPS11: Verificar que los paquetes de datos que se envían entre el servidor de Base de datos y las estaciones de trabajo donde radica la aplicación Registro Mercantil no viajan en un formato legible.

Caso de Prueba de Seguridad.			
Fecha:	___ ____ ____	Probado por:	Dusniel Horta Centeno.
Sistema:	Registro Mercantil.	ID Prueba:	**11.**
Activo:	• Canal de transmisión de información.	Tipo de Prueba:	Red (Aplicación, Red, Hardware)
Condiciones para la prueba:			
• Aplicación Registro Mercantil instalada y funcionando plenamente. • Programa **Simple Network Management Protocol** proveído con el Sistema Operativo Microsoft Windows Small Business Server 2003 Standard Edition Service Pack 1 instalado en el mismo servidor de Base de Datos. • Configurada y activada la directiva de seguridad IPSec tanto en el servidor como en la estación de trabajo desde la que se transmitirán o recibirán los paquetes.			

<table>
<tr><td colspan="5">Datos de entrada/Acciones a realizar:</td></tr>
<tr><td colspan="5">• Ejecutar el programa Simple Network Management Protocol y configurarle los filtros para comenzar a monitorear la red.
• Comenzar el proceso de monitoreo y supervisión de la red.
• Ejecutar la aplicación Registro Mercantil e intentar iniciar sesión haciendo una solicitud al servidor.
• Observar el reporte generado por el programa de monitorización y analizar los paquetes capturados para intentar obtener el nombre de usuario y la contraseña.</td></tr>
<tr><td colspan="5">Resultados esperados:</td></tr>
<tr><td colspan="5">• Los paquetes capturados estén en formato no legible.</td></tr>
<tr><td colspan="3">Resultado actual:</td><td colspan="2"></td></tr>
<tr><td>Pasado:</td><td></td><td>Fallido:</td><td></td><td></td></tr>
<tr><td colspan="5">Comentarios:</td></tr>
<tr><td colspan="5"></td></tr>
</table>

Conclusiones del capítulo.

En este capítulo se desarrolló la propuesta de aplicación práctica del modelo desarrollado en la construcción del módulo Registro Mercantil como parte de la solución SAREN. Teniendo como entrada diferentes artefactos de RUP ya generados por los analistas que laboran en el módulo, se realizó un análisis detallado de los principales criterios de seguridad a tener en cuenta en cada uno de los flujos de trabajo definidos en el modelo. Estos criterios quedaron documentados y recogidos bajo el término de artefactos de seguridad. El capítulo termina con la propuesta de 11 casos de prueba de seguridad que permiten establecer la forma de probar el sistema no solo desde el punto de vista del funcionamiento correcto de los flujos de negocios, sino desde la perspectiva de su capacidad operativa confiable.

CONCLUSIONES GENERALES.

Con el desarrollo de este trabajo, se considera que han sido cumplidos todos los objetivos propuestos, ya que:

1. Se elaboró un modelo, donde se definieron cuatro flujos de trabajo fundamentales, que permite integrar la gestión de la seguridad al proceso de desarrollo de software desde el inicio hasta las pruebas del sistema, de una forma continua y controlada.
2. Para cada uno de los flujos se detallaron los artefactos generados por los especialistas de seguridad en conjunto con los miembros del Equipo de Producto, establecidos por RUP para cada flujo.
3. El modelo es fácilmente comprensible para cualquier Equipo de Seguridad que desee tomarlo como guía durante el proceso de gestión. Para lograr este objetivo se representaron los artefactos generados mediante diagramas y tablas que hacen el análisis de los mismos más fácil a la vista.
4. Como prueba práctica de uso del modelo se desarrolló una propuesta de gestión de la seguridad para el módulo Registro Mercantil, que comprende la generación de los artefactos de seguridad más importantes para cada flujo de trabajo previsto para el desarrollo del módulo.

Además de los elementos mencionados anteriormente, durante el desarrollo de este trabajo se arribaron a un conjunto de conclusiones que poseen una marcada importancia, las mismas se relacionan a continuación:

1. Aunque aún persiste el método clásico de gestión de la seguridad, posterior al desarrollo, este ha demostrado no ser la mejor opción, por lo que está despertando un gran interés entre los fabricantes de software, incluir de forma paralela al proceso de Ingeniería de Software un mecanismo continuo que permita identificar y controlar desde los primeros momentos los principales riesgos de seguridad más importantes que pueden afectar al sistema una vez liberado y puesto en marcha en un entorno cliente.
2. El seguimiento de un proceso de gestión de seguridad controlado durante todo el ciclo de vida que comprende el desarrollo de soluciones de software no garantiza la liberación de un producto 100% seguro, ya que esto solo se lograría con un tiempo de prueba infinito, incluso para un sistema poco complejo.

3. Las metodologías de evaluación y análisis de riesgos existentes actualmente, aunque aportan muchos elementos que pueden aplicarse y adaptarse a proyectos específicos, no cubren, de manera general, todos los aspectos de seguridad y son desarrolladas por los proveedores fundamentalmente para su gestión interna.
4. Modelar las amenazas poniéndose en el lugar de un atacante, es una actividad imprescindible que permite conocerlas y determinar a cuales está expuesta la aplicación y los riesgos que representa cada una.
5. Designar recursos y especialistas que gestionen los principales criterios de seguridad durante el desarrollo, es una tarea imprescindible que no representa costos adicionales, si se tiene en cuenta que los beneficios que se obtienen, en la generalidad de los casos, son mayores que los gastos que implica reparar una vulnerabilidad detectada tardíamente.
6. Es una necesidad impartir periódicamente cursos de adiestramiento al personal que forma el Equipo de Producto con el objetivo de educarlos en una cultura de desarrollo donde la meta no sea solo alcanzar capacidad funcional plena sino que dicha capacidad sea operativa, pero confiable.
7. A pesar de que este modelo constituye una guía para gestionar de manera controlada e iterativa la seguridad del proceso de desarrollo de software no representa por ello un marco rígido a implantar, sino que por el contrario, puede ser adaptado a las características y necesidades específicas de cada proyecto en particular.

Finalmente, se considera que la utilización del modelo propuesto contribuirá a desarrollar soluciones de software en la UCI que cumplan con los requisitos de seguridad exigidos por los clientes, al permitir llevar un control continuo y efectivo de las amenazas, riesgos y medidas de mitigación correspondientes, en cada uno de los flujos de trabajo establecidos por la metodología RUP.

RECOMENDACIONES.

En el desarrollo de un trabajo siempre quedan un conjunto de aspectos relevantes, que por cuestiones muy diversas, no siempre pueden ser analizados o al menos con la profundidad que se requiere. Este trabajo no queda exento de ello; por lo que a continuación se relacionan un conjunto de ideas que se consideran necesarias para darle continuidad a este trabajo, contribuyendo a un modelo más acabado:

1. La extensión del modelo a los aspectos relacionados con la seguridad de los activos físicos que también forman parte de una solución de software los cuales no fueron abordados dentro del ámbito de este trabajo.
2. Que el modelo desarrollado se retroalimente, constantemente, de los resultados obtenidos en su aplicación en proyectos de desarrollo de software, de forma tal que muchos de estos resultados constituyan una fuente importante de información a reutilizar en procesos posteriores con características similares.
3. Elaborar una herramienta que permita dar soporte al modelo, llevando de forma automatizada el control y actualización constante de los artefactos generados en cada flujo y que permita, incluso, generar de forma instantánea reportes con resúmenes de todos los artefactos gestionados.
4. Estudiar otras metodologías de desarrollos de software existentes, diferentes de RUP, de manera que se pueda adaptar el modelo a otros casos particulares.
5. Estudiar otros flujos de trabajo, posteriores a las pruebas, con el fin de extender el modelo y su aplicación durante la etapa operativa del sistema una vez liberado a un entorno cliente, sobre todo tareas relacionadas con soporte de seguridad.

BIBLIOGRAFÍA.

Anderson, Ross J. (2001). Security Engineering. New York: Wiley.

A.S.S. Borghello, C. F. (2001). Seguridad Informática. Sus implicaciones e implementación.

Aldegani, G. M. (1997). Seguridad Informática. MP Ediciones. Argentina.

Aldereguía, C. (2002). "Prevención Automatizada de Errores."

Arellano, G. and A. d. B. Fontes (2004). "Desarrollo de código seguro."

Asteasuain, F. and L. A. Schmidt (2002). Aplicación de la Programación Orientada a Aspectos como Solución a los Problemas de la Seguridad en el Software**:** pp. 2-4.

Bertino, E., Sandhu, R. (2005). “Database security - concepts, approaches, and challenges.” Dependable and Secure Computing, IEEE Transactions on Volume 2, Issue 1.

Brown, Keith. (2000). Programming Windows Security. Reading, MA: Addison-Wesley.

Corporation, M.(2006). "Microsoft TAM v2.1.", from http://www.microsoft.com.

Devanbu, P. T. and S. Stubblebine (2000). "Software Engineering for Security: a Roadmap”, in Proceedings of the conference on The Future of Software Engineering".

Dutra, E. G. (2006). "Norma ISO 17799 Vs. ISO 27001." Retrieved 22/02/2007, from http://seguridadit.blogspot.com/2006/01/norma-iso-17799-vs-iso-27001.html.

Estrada, A. C. (2006). "ISO-27001: LOS CONTROLES (Parte I)."

Fernández, G. (2005). "Estándar codificación DOTNET." Retrieved 23/02/2007, 2007, from http://www.elguille.info/colabora/NET2005/giovannyfernandez_EstandarCodificacionNET.htm.

Group, C. D. (2004). "Metodología CORAS " , from http://coras.sourceforge.net.

Howard, M. and D. LeBlanc (2002). Writing Secure Code, Microsoft Press.

ISO/IEC-17799. (2005). "Information technology - Security techniques - Code of practice for information security management."

ISO/IEC-27001 (2005). "Information technology - Security techniques - Information security management systems - Requirements."

Jacobson, I., G. Booch, et al. (2000). El Proceso Unificado de Desarrollo de Software.

Lewis, W. E. (2005). Software Testing and Continuous Quality Improvement, AUERBACH PUBLICATIONS.

Lipner, S. and M. Howard (2005). "El ciclo de vida de desarrollo de seguridad de Trustworthy Computing."

Lippert, Eric. (2002). Visual Basic .NET Code Security Handbook. Birmingham, UK: Wrox Press.

Litchfield, D. (2006). "Which database is more secure? Oracle vs. Microsoft.".

Lockhart, A. (2006). Network Security Hacks O'Reilly.

Madrid, M. d. A. P. d. (2006). MAGERIT. Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información.

Manunta, D. G. (2005). SEGURIDAD: UNA INTRODUCCIÓN.

Marañón, G. Á. (2005). "Amenazas deliberadas a la seguridad de la información." Retrieved 18/02/2007, 2007, from http://www.iec.csic.es/criptonomicon/seguridad/amenazas.html.

Margini, D. A. (2006). "Seguridad, por qué hay ataques y cómo protegernos."

Menezes, Alfred J. et al. (1997). Handbook for Applied Cryptography. Boca Raton, FL: CRC Press.

Molpeceres, A. (2002). "Procesos de desarrollo: RUP, XP y FDD." 11.

P.F, D. (2006). Análisis y Modelado de Amenazas.

Rodríguez, I. A. and A. B. Grimaldi (2003). Seguridad y Software Libre. Universidad de Chile Facultad de Ciencias Físicas y Matemáticas Departamento de Ciencias de Computación.

Rodríguez, Y. R. and J. G. Martín (2006). "Modelación del Negocio Registro Mercantil."

Sandoval, A. N. (2005, 23/02/2007). "Estándares de seguridad en la información." from http://www.enterate.unam.mx/Articulos/2005/febrero/seguridad.htm.

Scheibelhofer, K. (2005). "Security in the analysis and design phase." Retrieved 01/04/2007, 2007.

Schneier, Bruce. (1996). Applied Cryptography: Protocols, Algorithms, and Source Code in C. 2d ed. New York: Wiley.

Shreyas, D. (2001). “Software Engineering for Security: Towards Architecting Secure Software” ,Information and Computer Science Dept., University of California, Irvine, CA 92697,USA., abril de 2001.

Trike, S. O. d. l. M. (2005). "Metodología de Análisis de Riesgos Trike." Retrieved 01/02/2007, from http://www.octotrike.org.

Viega, J., J. T. Bloch, et al. (2001). " “Applying Aspect-Oriented Programming to Security”, Cutter IT Journal, febrero de 2001."

Viega, J., J. T. Bloch, et al. (2000). ITS4: A Static Vulnerability Scanner for C and C++ Code”, in Proceedings of the 16th Annual Computer Security Applications Conference (ACSAC 2000). IEEE, 2000.

Wikipedia. (2007). "Normas ISO/IEC 17799.", from http://es.wikipedia.org/wiki/ISO/IEC_17799.

Williams, L. (2005). Security Testing. Retrieved 01/02/2007, from http://www.microsoft.com/whdc/driver/security/threatmodel.mspx.

Win, B. D., B. Vanhaute, et al. (2001). ""Security through Aspect Oriented Programming”, in Proceedings of the 1st working conference on Network Security, noviembre de 2001."

ANEXOS.

Anexo 1. Método de clasificación STRIDE.

Seleccionar el activo y para cada amenaza a que está expuesto, analizar si es susceptible a los siguientes elementos:

- **S**poofing (Suplantación de Identidad): ¿El atacante puede obtener acceso utilizando una identidad falsa?
- **T**ampering (Manipulación de datos): ¿Podrá el atacante modificar los datos de la aplicación?
- **R**epudiation (Repudio): ¿Podrá el atacante no dejar rastros del ataque?
- **I**nformation disclosure (Descubrimiento de información): ¿El atacante accederá a datos privados o potencialmente perjudiciales?
- **D**enial of service (Denegación de servicio): ¿El atacante puede reducir la disponibilidad del sistema?
- **E**levation of privilege (Elevación de privilegios): ¿El atacante puede asumir la identidad de un usuario privilegiado?

Anexo 2. Método de puntuación DREAD.

Método utilizado para calcular el riesgo que supone una determinada amenaza sobre un activo atendiendo a los siguientes criterios:

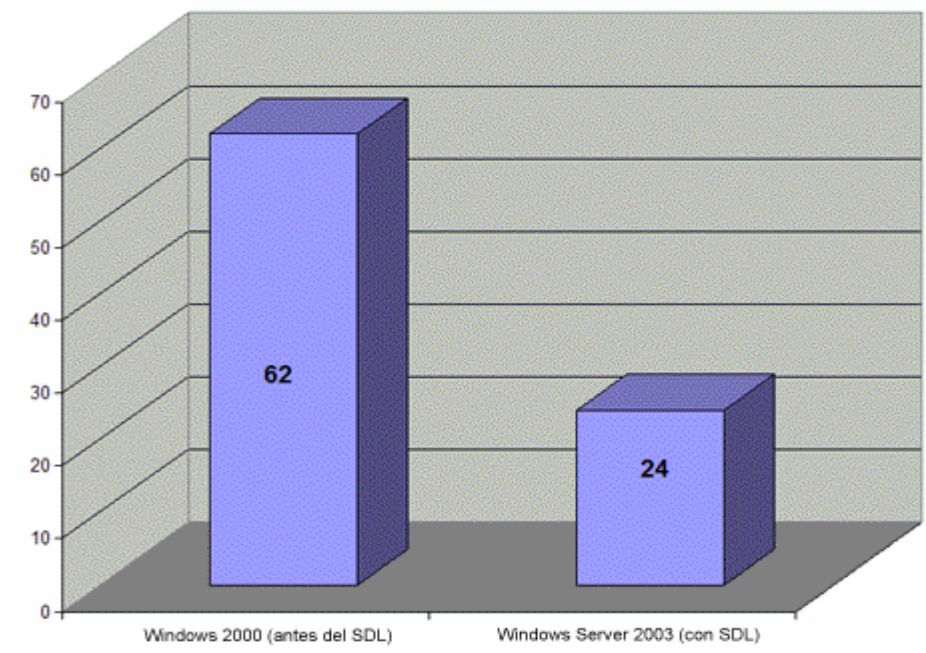
- **D**amage potencial (Daño potencial): ¿Cual es el daño que puede originar la vulnerabilidad si llega a ser explotada? ¿Cuáles son las consecuencias de un ataque exitoso?
- **R**eproducibility (Reproducibilidad): ¿Es fácil reproducir las condiciones que propicien el ataque? ¿Podría realizarse el ataque en cualquier momento o sólo bajo ciertas circunstancias?
- **E**xploitability (Explotabilidad): ¿Es sencillo llevar a cabo el ataque? ¿Qué tanto conocimiento tendría que tener el atacante para explotar la vulnerabilidad?
- **A**ffected users (Usuarios afectados): ¿Cuantos usuarios se verían afectados?
- **D**iscoverability (Descubrimiento): ¿Cuán fácil le resulta al atacante conocer que existe la vulnerabilidad?

A cada criterio se le asigna un número en el rango **1-3**.

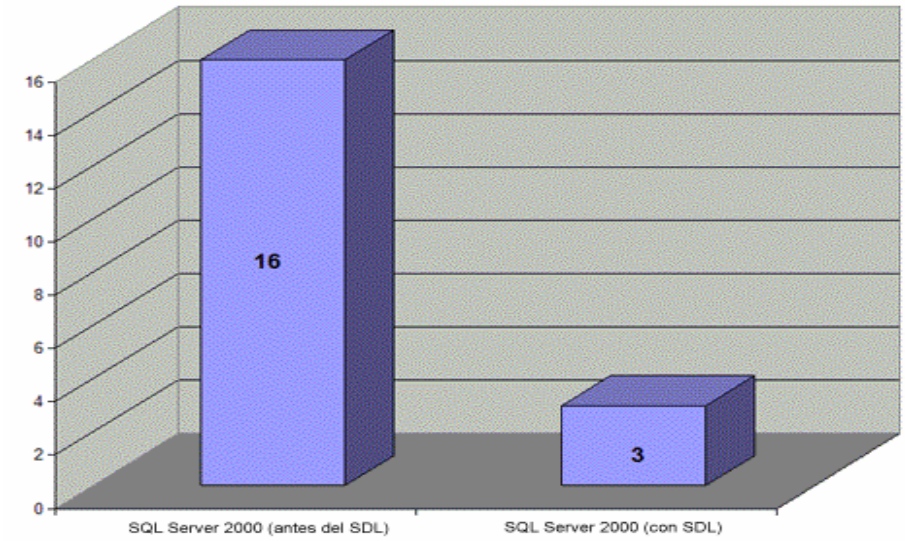
- 1 (Riesgo bajo).
- 2 (Riesgo medio).
- 3 (Riesgo alto).

El resultado de la suma de los 5 valores supone el riesgo que existe sobre el activo y se procede a dar prioridad a la amenaza. 5-7(Riesgo bajo), 8-11(Riesgo medio), 12-15(Riesgo alto).

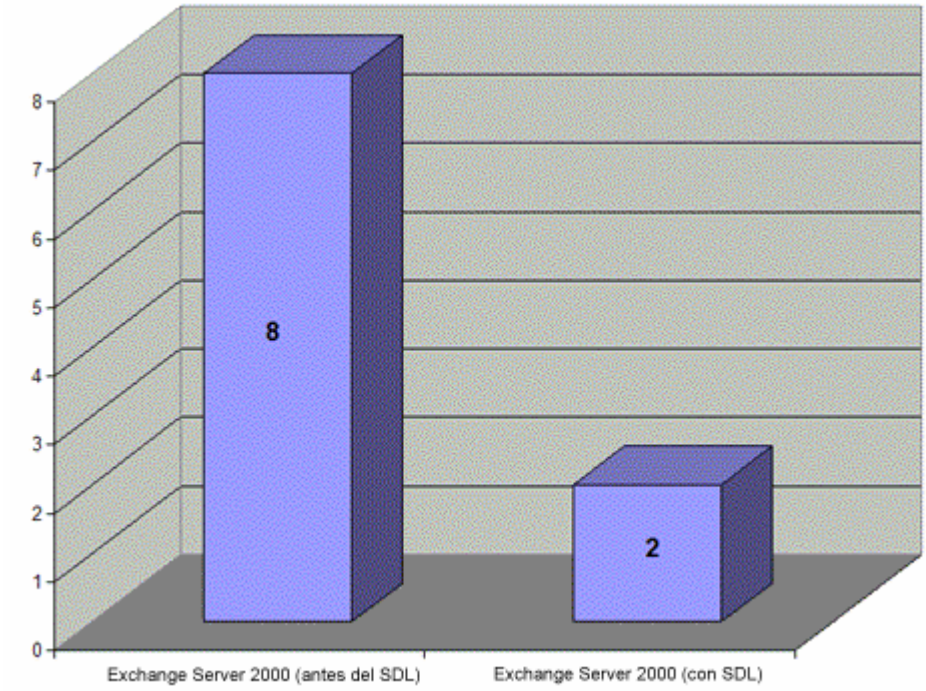
Anexo 3. Resultados de la implementación del ciclo de vida de desarrollo de seguridad en Microsoft.



1. Boletines de seguridad publicados en un período de un año tras el lanzamiento de los dos sistemas operativos de servidor de Microsoft más recientes: Windows 2000 y Windows Server 2003.



2. Boletines de seguridad de SQL Server 2000 antes y después del SDL.

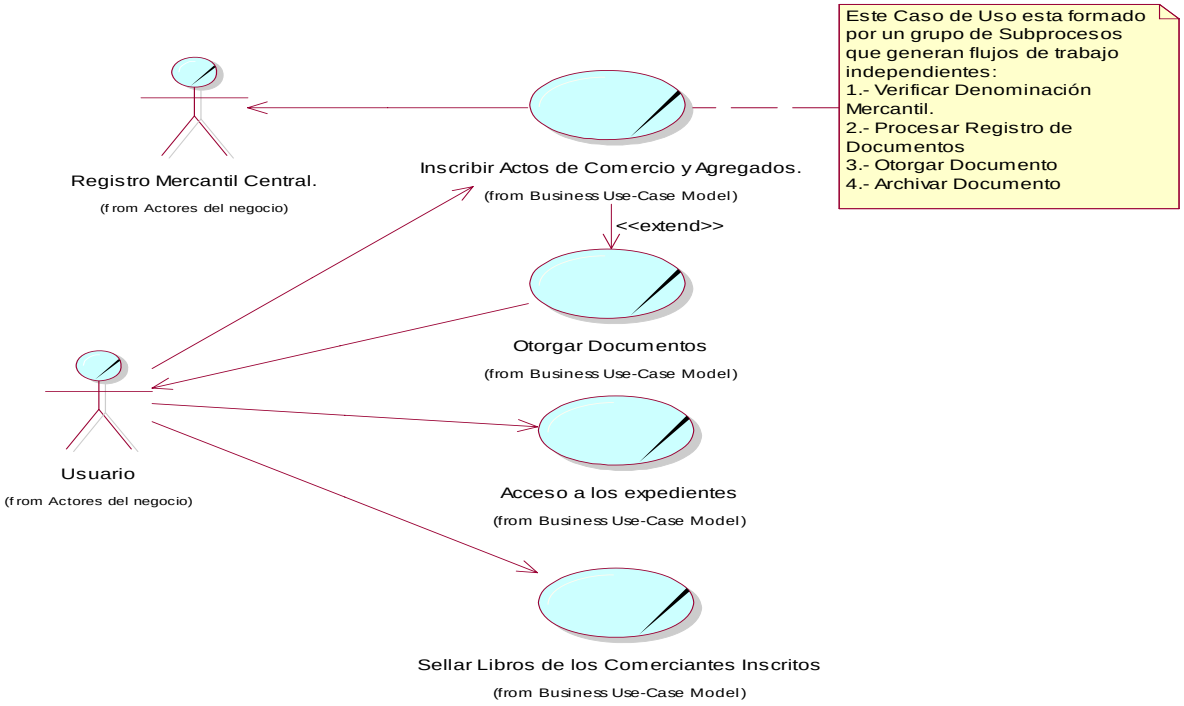


3. Boletines de seguridad para Exchange Server 2000 antes y después del SDL.

Anexo 4. Lista con técnicas de mitigación de riesgos.

TIPO DE AMENAZA	TÉCNICAS DE MITIGACIÓN.
Suplantación de identidad.	4. Autenticación apropiada. 5. Protección de datos secretos. 6. No almacenar secretos.
Manipulación de datos.	• Autorización apropiada. • Uso de Hash. • Firma digital. • Protocolos resistentes a la manipulación.
Repudio.	• Firma digital. • Auditoria. • Trazas.
Información descubierta.	• Autorización. • Protocolos Privados. • Encriptado de datos. • No almacenar secretos.
Denegación de servicio.	• Autenticación apropiada. • Autorización apropiada. • Filtrado de datos. • Calidad del servicio.
Elevación de privilegios.	• Ejecutar con privilegios mínimos.

Anexo 5. Diagrama de Casos de Uso del Negocio Registro Mercantil.



Anexo 6. Guía de diseño de codificación segura propuesta para el módulo Registro Mercantil.

- **Reglas de cumplimiento obligatorio para la codificación.**
  - Todos los atributos de clases que manipulan cadenas las cuales representan direcciones URL deben ser de tipo **Uri** perteneciente al Namespace **System**.
  - Los métodos que retornan arreglos deben devolver un clon de estos o retornarlos mediante una colección de las proveídas por el Namespace **System.Collections** o de una que herede de estas.
  - Todos los ensamblados deben ser firmados con la herramienta **sn.exe** proveída con Visual Studio .NET.
  - Los comentarios escritos durante el proceso de escritura del código fuente no pueden tener información confidencial tales como contraseñas y nombres de usuarios entre otros.

- Todo el acceso a la información almacenada en la Base de Datos debe realizarse mediante la capa de acceso a datos generada con la herramienta Tier Developer y no mediante concatenación de cadenas que formen sentencias SQL.
- El manejo de excepciones debido a errores internos de funcionamiento de la aplicación debe garantizar que nunca se muestren errores con información técnica a los usuarios.
- Los campos que representan las entradas de datos por los usuarios deben ser validados en su totalidad, garantizando que los datos que pasan al interior de la aplicación sean los correctos.
- Las entradas de información que recibe la aplicación a partir de ficheros deben ser validadas, así como la existencia de los directorios donde estos se almacenan y los permisos que sobre ellos se tienen.
- Las interfaces de acceso a dispositivos de impresión, escaneo, etc., con los cuales interactúa la aplicación, deben ser validadas por código en los puntos de chequeos correspondientes.
- El manejo de excepciones dentro de un bloque **try - catch** debe garantizar que los errores manejados dentro del bloque catch no generen otros errores que formen una cadena infinita de errores que pongan en peligro la estabilidad de la aplicación.
- El manejo de excepciones no debe realizarse usando solo la clase base Exception contenida en el namespace **System.** En su lugar se debe usar las excepciones específicas para el tipo de error que puede generar la sección del código fuente en cuestión. Solo al final incluir un bloque catch con la excepción del tipo base, para garantizar que si la excepción que se genera no es de ninguno de los tipos esperados pues sea capturada por la base.

```
private void button1_Click(object sender, System.EventArgs e)
{
      // Todas las clases excepciones heredan de su tipo base.
      try
      {
       // Código Fuente que puede generar una o varias excepciones.
      }
      catch(System.FormatException formatEx)
      {
       //Tratamiento del error si el formato no es válido.
      }
      catch(System.StackOverflowException  DPex)
      {
```

```
            //Tratamiento del error si hay desbordamiento de la pila.
        }
        catch(System.ArgumentOutOfRangeException aFRex)
        {
         //Tratamiento del error si el argumento está fuera de rango.
        }
        catch(System.FieldAccessException aFex)
        {
          //Tratamiento del error si el acceso a determinado fichero no
         //está permitido.
        }
        catch(System.DivideByZeroException ex)
        {
        //Tratamiento del error para división por cero.
        }
        catch(System.Exception ex)
        {
         //Tratamiento del error en caso que fallen los catch anteriores.
        }
}
```

- **Uso de herramientas de análisis de código.**
  - **FxCop** es una herramienta para el análisis del código fuente que comprueba los ensamblados para saber si hay conformidad con las mejores prácticas de diseño para el .NET Framework“. Permite encontrar más de 200 tipos de defectos introducidos por los desarrolladores, tanto errores de seguridad como de diseño, entre otros. Esta herramienta será establecida en el módulo Registro Mercantil para realizar tareas de este tipo.
  - Cada desarrollador debe conocer el funcionamiento de la herramienta FxCop y usarla para realizar, durante el proceso de codificación, análisis estático del código fuente.
- **Recomendaciones**.
  - No desarrollar el proceso de escritura del código fuente, usando una sesión con permisos de administrador de la estación de trabajo. Las cuentas de administrador tienen acceso completo a todos los recursos de la maquina y tal vez la aplicación acceda a recursos no autorizados (directorios, ficheros, entre otros) los cuales bajo cuentas de administrador son difíciles de detectar.
  - Establecer los ficheros de configuración de la aplicación como recursos embebidos.

Anexo 7. Procedimientos de mitigación.

PM1: Configuración segura de los AP de la red inalámbrica local.

Son cuatro elementos básicos que deben seguirse para garantizar los aspectos mínimos de seguridad de la red interna inalámbrica de tecnología 3Com:

1. **Cambiar la contraseña por defecto de la interfaz Web de configuración del Acces Point.**

   La interfaz Web que permite la configuración y administración del Acces Point una vez instalada establece por defecto la contraseña **admin.** El paso inmediato luego de la instalación consiste en acceder mediante el explorador a la interfaz de configuración y cambiar la contraseña **admin** por una que cumpla con un patrón de contraseñas robusto. Si este paso no se cumple, cualquier usuario que tenga acceso a la red puede modificar la configuración de seguridad establecida y obtener el control de la red.

2. **Cambiar el ESSID por defecto (3Com) y ocultarlo.**

   El ESSID es el nombre único que permite identificar a toda una red inalámbrica. Cuando se instala el Acces Point, inicialmente se le asigna a la red que este controla un ESSID por defecto llamado 3Com.

   - El paso inmediato consiste en sustituir el nombre 3Com por uno que sea difícil de adivinar por un adversario. Una buena manera de establecer este nombre es utilizar una combinación de caracteres alfanuméricos con una longitud mayor de 12 caracteres y menor de 32(límite máximo permitido por la tecnología inalámbrica 3Com). Incumplir este paso puede traer como consecuencia que un adversario detecte la red simplemente colocando un programa Snnifer que realice búsquedas constantes de Redes inalámbricas comenzando con los nombres por defecto.
   - Sin salir de la interfaz actual ir a la opción **SSID Broadcast** y marcar la opción **Disable,** esto previene que clientes no autorizados puedan detectar el SSID e intentar conectarse a la red.

3. **Habilitar la encriptación.**

   Con el mecanismo de **encriptación** deshabilitado cualquier cliente con un adaptador inalámbrico puede conectarse a la red. Inicialmente este es el estado en que se encuentra el

mecanismo. Es recomendable que en un primer momento, mientras se configuran todos los demás servicios de la red, esta opción siga deshabilitada, sin embargo una vez que todo esté listo, se debe proceder a habilitar el mecanismo de seguridad como se explicará a continuación.

El Acces Point soporta dos tipos de encriptación:

- Wifi Protected Acces(WPA: Es un método de encriptación de 256 bit con claves que cambian cada cierto tiempo.
- Wireless Equivalent Privacy (WEP): Es un método de encriptación de 64 bit o 128 bit con claves configurables por el usuario.

WPA provee un alto nivel de seguridad debido a la longitud de la clave y su cambio dinámico. Los fabricantes de 3Com recomiendan que se use WPA con clientes que soporten el mecanismo.

A continuación se propone la forma de configurar la seguridad usando WPA.

- En la interfaz **Security,** seleccionar la opción WPA y seguir a la interfaz WPA.
- En la opción **Chipre Suite,** seleccionar TKIP.
- En la opción **Authentication,** seleccionar Pre-Shared Key.
- En la opción **Pre-Shared Key** type, seleccionar Passphrase.
- En el cuadro **Pre-Shared Key** escribir una frase o cadena de entre 8 y 64 caracteres. Esta frase será usada para generar una clave de 256 bit.
- Click en **Apply** para generar la clave.

Las PC clientes que se conectarán al Acces Point deben poseer esta misma clave generada y el método WPA.

4. Habilitar el Control de Acceso.

Esta opción permite que solo las PC con red inalámbrica autorizadas puedan usar al Acces Point. Para esto se selecciona la opción **Acces Control** para mostrar la interfaz **WLAN MAC Filtering Table.**

- En la opción **Enable MAC Filtering**, seleccionar **Yes.**
- En la opción **Acces Rule,** seleccionar **Affirm.**
- En la tabla MAC Filtering entrar la dirección MAC de las PC clientes (máximo 32) que tendrán acceso a la red.

Esta lista puede ser usada como una lista de denegación. Si se desea esto pues solo se selecciona en la opción **Acces Rule** la opción **Deny.**

PM2: Patrón de generación de contraseñas.

Determina los requisitos de complejidad que las contraseñas deben cumplir.

- Tener 8 caracteres de longitud, como mínimo.
- No debe contener parte o todo el nombre de la cuenta del usuario.
- Estar compuesta por combinaciones de caracteres de tres de las siguientes categorías:
    - Letras mayúsculas, de la A - Z.
    - Letras minúsculas, de la a – z.
    - Dígitos en base 10, de 0-9.
    - Caracteres no alfanuméricos (ej. !, $, #, %).

PM3: Evitar ataque de Inyección SQL.

- Usar siempre procedimientos almacenados para el acceso a la base de datos.
- No concatenar cadenas para formar sentencias SQL mediante el código.
- Generar la capa de acceso a datos con la herramienta Tier Developer 4.0.
- Validar todas las entradas de usuario que impliquen conexión a base de datos mediante expresiones regulares que filtren caracteres y cadenas que forman sentencias SQL válidas.
    - Insert, delete, drop, comilla simple (‘), >,<, --, etc.

PM4: Asegurar el canal de comunicación usando directiva IPSec.

- Abrir la Consola de Administración Microsoft (MMC) usando el comando **mmc**.
- En el menú archivo seleccionar Agregar o quitar complemento.
- Agregar el complemento **Directivas de Seguridad IP en Equipo local**.
- Seleccionar la directiva **Servidor seguro.**
- Configurar la regla de seguridad **Todo el trafico de IP.**
    - Usar método de autenticación **clave previamente compartida** e introducir la clave.
    - Usar acción de filtrado **Requerir alta Seguridad** y como método de seguridad seleccionar **Negociar la seguridad.**

- Seleccionar las opciones **Aceptar comunicación no segura, pero responder usando IPSec** y Confidencialidad directa perfecta de clave de sesión.
- Luego establecer como Algoritmo de cifrado **3DES** y de integridad **SHA1**.
- En Lista de filtros IP seleccionar **Todo el tráfico IP** y establecer como dirección de origen la opción **Una subred IP determinada.** Escribir en el campo **Dirección IP** la dirección de subred específica y la máscara de subred 255.255.255.
- En Dirección de destino seleccionar la opción **Una subred IP determinada.** Escribir la dirección de la subred específica y la mascara de subred 255.255.255**.**
- Establecer como protocolo de filtrado **TCP** y seleccionar la opción **Desde este puerto** y escribir **1521.**
- Asignar la directiva configurada al servidor. Para esto selecciona en la consola principal la directiva configurada y en el menú **Acción->Todas las tareas->Asignar.**

Esta directiva de IPSec es para el servidor de Base de datos. Sin embargo para los servidores de aplicaciones es similar, solo se cambian algunos detalles de los dos últimos pasos**,** quedando de la siguiente forma:

- En Lista de filtros IP seleccionar **Todo el tráfico IP** y establecer como dirección de origen la opción **Dirección IP específica** y escribir la dirección IP del servidor de Base de datos y como dirección de destino **Mi direccion IP.**
- Establecer como protocolo de filtrado **TCP** y seleccionar la opción **Desde este puerto** y escribir **1521.**

PM5: Configuración del Firewall en el servidor de Base de datos local.

El firewall del servidor de Base de datos local de las oficinas debe estar activado y tener la siguiente configuración.

Pasos:

- Ir a Panel de control y ejecutar **Firewall de Windows.**
- En la pestaña Excepciones seleccionar la opción **Agregar Puerto.**
- Adicionar cada uno de los siguientes puertos según corresponda:
  - Puerto 13000 y 14000. TCP/IP (kit de administración).
  - Puerto 1521 TCP/IP (Listener del servidor Oracle).

- Puerto 3938 TCP/IP (Oracle DataBase Control).
- Puerto 1158 y 5500 TCP/IP (Enterprise Manager).
- Puerto 445 TCP/IP. (Carpetas compartidas).
- Puerto 3389 TCP/IP (escritorio remoto).
- Puerto 15000 UDP. (Sincronización forzada del Kaspersky).
- Puerto 123 UDP (Servidor de tiempo)
- Puerto 21 y 20 TCP/IP (Servicio de FTP).
- Puerto 80 TCP/IP (Servicio HTTP).

- Regresar a la pestaña **General** y seleccionar la opción **Activado** y verificar que la opción **No permitir excepciones** esté desmarcada.

PM6: Configuración del Firewall en las estaciones de trabajo del Registro Mercantil.

El firewall de las estaciones de trabajo donde está instalada la aplicación Registro Mercantil en las oficinas debe estar activado y tener la siguiente configuración.

Pasos:

- Ir a Panel de control y ejecutar **Firewall de Windows.**
- En la pestaña Excepciones seleccionar la opción **Agregar Puerto.**
- Adicionar cada uno de los siguientes puertos según corresponda:
  - Puerto 13000 y 14000. TCP/IP (kit de administración).
  - Puerto 3389 TCP/IP (escritorio remoto).
  - Puerto 445 TCP/IP. (Carpetas compartidas).
  - Puerto 15000 UDP. (Sincronización forzada del Kaspersky).
- Regresar a la pestaña **General** y seleccionar la opción **Activado** y verificar que la opción **No permitir excepciones** esté desmarcada.

PM7: Sistema de autenticación para la aplicación.

El sistema de autenticación de usuarios que posee la aplicación Registro Mercantil debe cumplir con los mecanismos y requisitos básicos de seguridad que garanticen un acceso confiable a las funcionalidades que ofrece la aplicación. A continuación se muestra una propuesta del mismo.

- Al ejecutar la aplicación debe mostrarse una ventana donde se solicite nombre de usuario y contraseña para acceder.
- El campo donde se escribe la contraseña debe garantizar que los caracteres se representen mediante símbolos y no en texto claro.
- Encriptar la contraseña usando algoritmo SHA1 antes de enviarse por la red.
- Almacenar en la Base de datos solo el Hash devuelto por el método que encripta la contraseña.
- Chequear el patrón de generación de contraseñas definido.
- Limitar el número de intentos fallidos del usuario a un máximo de 4 oportunidades.
- El mensaje que se le muestre al usuario en caso de intentos fallido no debe ofrecer el detalle de cual campo está incorrecto, simplemente limitarse a **"Credenciales no válidas"**.
- Al completar el número de intentos fallidos notificarlo al usuario y terminar el servicio.

PM8: Configuración del Setup del BIOS para evitar el uso de dispositivos extraíbles en estaciones de trabajo.

La configuración del Setup del BIOS en las estaciones de trabajo donde se encuentra instalada la aplicación Registro Mercantil en las oficinas debe tener la siguiente configuración.

- Establecer el Setup con contraseña.
- Despertar solo por disco duro (HDD).
- Deshabilitados todos los puertos USB menos uno en la estación donde se use el Escáner.
- Deshabilitar la unidad de disco floppy.
- Deshabilitar la unidad lectora de CD, DVD.

PM9: Firmar los ensamblados con nombre fuerte.

- Abrir el proyecto de ensamblado usando Visual Studio .NET y lenguaje C#.
- Abrir la consola de comandos de Visual Studio .NET 2003.
- Mediante el comando **cd** navegar hasta el directorio donde se desea almacenar la clave generada para el firmado.

- Escribir el comando sn.exe –k “ClaveDeFirma.snk” para generar las claves y almacenarlas en el directorio seleccionado.
- Escribir en el atributo **“(Lockhart 2006)**`del` **AssemblyInfo** del ensamblado la dirección donde se encuentra la clave de firma generada, de la siguiente manera:
    - **[assembly:AssemblyKeyFile("..\\..\\ClaveDeFirma.snk")].**
- Repetir los dos últimos pasos para todos los ensamblados que conforman el proyecto y que no están firmados aún, usando la misma clave de firmado generada anteriormente.
- Compilar el proyecto, quedando de estar forma firmados todos los ensamblados que lo componen.

PM10: Manejar los errores mostrados al usuario.

- Crear una tabla XML personalizada con un listado de mensajes de error predefinidos. Cada mensaje debe poseer la información estrictamente necesaria sin revelar contenido técnico.
- Cada error en la tabla contiene un número único que lo identifica y su descripción correspondiente.
- Establecer la tabla XML como recurso embebido de la aplicación.
- Crear clases de manejo de excepción personalizadas donde cada una posea un método o constructor que reciba como parámetro el número que identifica el mensaje en la tabla.
    - FormatoNoValidoException ex = new FormatoNoValidoException (numero).
- El método o constructor accede a la tabla y carga el mensaje correspondiente.
- Incluir el segmento de código que puede generar error, dentro de un bloque **try { }**y establecer un conjunto de bloques Catch() donde para cada uno se especifique el tipo posible de excepción a generar y el número de mensaje correspondiente.
- Al final de los bloques catch personalizados, se debe incluir siempre un bloque Catch() general que incluya la excepción **Exception**, esto garantiza que si no se generó una excepción del tipo esperado pues esta sea capturada por la excepción base, de la cual heredan todas, en cuyo caso se muestra un mensaje también de alto nivel.

Este mecanismo garantiza que todo el funcionamiento del código que genere una excepción sea tratado de forma personalizada y mediante una tabla de errores que concentra y controla todos

los posibles mensajes que deben mostrarse al usuario y no el generado por la excepción como tal.

PM11: Validar los puntos de entrada de datos a la aplicación.

Partiendo de la información proveída por los Diagramas de Flujo de Datos realizados, se deben validar todos los puntos de entrada identificados. Para el caso de la Aplicación Registro Mercantil existen tres puntos de entrada o salida:

- Entrada de datos por los formularios.
    - Validar que el tipo de datos que se introduce corresponde con el que se espera.
    - Validar que la longitud del valor introducido está en el rango permitido para su tipo.
    - Validar que aquellos campos que admiten cadenas de letras y caracteres no permitan el paso de caracteres o cadenas peligrosas que formen sentencias que son válidas para ejecutarse en el servidor de base de datos como una consulta.
- Intercambio de datos entre la aplicación y el servidor de base de datos, a través de la red local.
    - Validar que la información que llega desde la base de datos, es la que se espera. Ya que no puede tenerse la certeza absoluta de que no ha sido manipulada en el servidor o durante el tránsito por la red.
- Envío de información hacia dispositivos de impresión y escaneo.
    - Cuando se envía un flujo de información a un dispositivo de impresión o escaneo es necesario validar que este se encuentre conectado y correctamente configurado, pues nunca debe pensarse que todo va a funcionar como se espera.
    - Si la información se intercambia con un fichero de datos almacenado en algún lugar de la PC, pues debe validarse en el momento de acceder a él su existencia o no, si se tiene permiso de acceso, de lectura o de escritura.

Estos principios básicos aunque parecen obvios, constituyen los elementos más comunes a través de los cuales se vale un adversario para provocar fallos en el sistema que lo pongan en un estado de inconsistencia, o que revelen información importante a partir de fallos no controlados.

Todos estos puntos deben preverse, chequearse y gestionar los posibles errores que pueden traer los datos una vez pasados por el punto de chequeo.

PM12: Evitar Búsquedas y bucles excesivamente largos.

Las búsquedas de datos excesivamente largas provocan muchas veces disminución del rendimiento de la aplicación debido al uso desmedido de la memoria y los recursos manejados. Algunos pasos para evitar esto se definen abajo:

- Solicitar uno o varios parámetros de búsqueda que limiten el dominio o la cantidad de elementos a extraer.
- Si la búsqueda por parámetros continua siendo excesivamente larga, pues cargue en la memoria solo cierta cantidad de estos y si no se encuentra el elemento buscado, pues continúe la extracción a partir del último y llene la lista en memoria nuevamente con los otros X elementos siguientes.
- Utilice materialización de objetos por demanda. Esto significa que si una clase está compuesta por varios atributos que representan clases persistentes, las cuales muchas veces también son complejas en cuanto a su contenido interior, pues debe establecerse el mecanismo de ir recuperando sus miembros a medida que se necesite acceder a ellos.

TERMINOLOGÍA.

A

Activo: Recurso tangible o intangible del sistema de información o relacionado con éste, necesario para que la organización funcione correctamente y alcance los objetivos propuestos.

Amenaza: Es un evento que pueden desencadenar un incidente en la organización, produciendo daños materiales o pérdidas inmateriales en sus activos.

Antivirus: Programa encargado de evitar que cualquier tipo de virus ingrese al sistema, se ejecute y se reproduzca. Para realizar esta labor existen muchos programas, que comprueban los archivos para encontrar el código de virus en su interior.

Artefacto: Termino general para cualquier tipo de información creada, producida, cambiada o utilizada por los trabajadores en el desarrollo del sistema. Puede ser un modelo, un documento, o un diagrama.

Ataque: Evento, exitoso o no, que atenta sobre el buen funcionamiento del sistema.

B

Buffer Overflow: Error generado cuando un programa recibe una entrada mayor a la que se espera, sobrescribiendo áreas críticas de memoria.

Bug: Un error en un programa o en un equipo. Se habla de bug si es un error de diseño, no cuando la falla es provocada por otro motivo.

C

Cifrar: Ver **Criptografía.**

Clave, Contraseña: Palabra o frase que permite acceder a un sistema, encriptar un dato, determinar privilegios de usuario.

Clave pública: En un Sistema Asimétrico de Cifrado es la clave que todos conocen para cifrar o descifrar un mensaje.

Clave privada: En un Sistema Asimétrico de Cifrado es la clave que solo el emisor conoce para cifrar o descifrar un mensaje.

Cliente: Sistema o proceso que solicita a otro sistema o proceso que le preste un servicio.

Código fuente: Un programa escrito en un lenguaje entendible para el hombre pero no para la computadora. Necesita ser traducido (Compilar) a lenguaje máquina para ser interpretado por esta última.

Colgar: Hacer que un sistema deje de funcionar, logrando la denegación del servicio que esta puede prestar.

Confidencialidad de la información: Se refiere a que la información solo puede ser conocida por individuos autorizados y mediante los mecanismos establecidos.

Cortafuegos: Ver **Firewall.**

Criptografía: Ciencia que consiste en transformar un mensaje inteligible en otro que no lo es, mediante la utilización de claves, que solo el emisor y el receptor conocen.

D

Datagrama: Conjunto de datos que se envían como mensajes independientes a través de la red. Unidad utilizada por el protocolo **UDP**.

Dato: Unidad mínima con la que se compone cierta información.

Defensa en profundidad: Práctica estratégica y principio de seguridad para conseguir seguridad de la información a diferentes niveles. Requiere de la aplicación inteligente de las técnicas y tecnologías que existen. Esta estrategia recomienda un balance entre la capacidad de protección, costo, rendimiento y consideraciones operacionales. Este principio sugiere que en donde un control podría ser razonable, muchos controles que combatan los riesgos en diferentes formas son mejores. Los controles, cuando son usados en profundidad, pueden hacer que varias vulnerabilidades sean extraordinariamente difíciles de que sucedan o incluso imposible.

Denegación de Servicios: Acciones que impiden a un sistema funcionar de acuerdo a su propósito.

Desastre o Contingencia: Interrupción de la capacidad de acceso a información y procesamiento de la misma a través de computadoras necesarias para la operación normal de un negocio.

Disponibilidad de la información: Relacionada con la seguridad de que la información pueda ser recuperada en el momento que se necesite, por el personal autorizado.

E

Encriptar: Ver **Criptografía.**

Exploit: Programa que aprovecha un **bug** de un sistema. Aprovecha el error para conseguir escalar privilegios de un usuario o la caída del sistema.

F

Firewall: Barrera de protección. Es un procedimiento que coloca un sistema de computación programado especialmente entre una red segura y una red insegura. Restringe el acceso a un determinado recurso y filtra el flujo de información tanto entrante como saliente.

Fuerza bruta: Método de ataque que se basa en aprovechar diccionarios para comparar las palabras almacenadas en él con las contraseñas del sistema y obtenerlas.

G

H

HTML (Lenguaje de Marcado de HiperTexto): Lenguaje sobre el que está basada la estructura de las páginas que forman las aplicaciones Web.

HTTP (Protocolo de Transferencia de HiperTexto): Protocolo de la capa de aplicación sobre el que está basado todo el proceso de transferencia de información en el World Wide Web.

I

Información: Agregación de datos con un significado específico más allá de cada uno de estos.

Impacto: consecuencia de la materialización de una amenaza.

Intruso: Aquella persona que con una variedad de acciones intenta comprometer un recurso de hardware o software.

Integridad de la información: Garantía de que la información no ha sido alterada, borrada, reordenada ni copiada por personas no autorizadas.

IP (Protocolo de Internet): Protocolo de comunicación que proporciona el servicio de envíos de paquetes para los protocolos TCP, UDP, ICMP. Pertenece a la capa de red.

IPSec (Seguridad del Protocolo de Internet): Es un marco de estándares abiertos para lograr comunicaciones privadas seguras a través de redes con el Protocolo de Internet (IP) mediante el uso de servicios de seguridad criptográfica. Funciona en la capa de red.

ISO (Organización Internacional para la Estandarización): Organización mundial dedicada al desarrollo de estándares. Uno de sus comités se ocupa de los sistemas de información. Han desarrollado el sistema de referencia OSI y protocolos estándares para diferentes niveles de este modelo.

J

K

Kerberos: Sistema de Seguridad en el que las credenciales viajan encriptadas a través de la red.

KeyLogger: Grabador de teclas pulsadas. Es utilizado cuando se desea conocer las credenciales de usuarios o cualquier otra información, donde se utilice el teclado como vía de entrada al sistema.

L

M

Medida de mitigación: Acción defensiva llevada a cabo para eliminar una vulnerabilidad que puede desencadenar un ataque sobre un activo crítico.

N

O

OSI (Interconexión de Sistemas Abiertos)**:** Programa de estandarización internacional creado por **ISO** para desarrollar normas que faciliten la interoperabilidad entre equipos de diversos fabricantes.

P

Parche: Actualización o modificación de un programa ejecutable para solucionar un problema, corregir un **bus** o para cambiar su comportamiento.

PC: Computadora Personal.

Principios de seguridad: Reglas fundamentales de seguridad que describen el correcto comportamiento y/o diseño de una aplicación, las cuales se pueden mejorar cambiando las posturas sobre seguridad. Estos principios son generales, deben tomarse solo como una guía y no como una manual de procedimientos a seguir.

Protocolo: Conjunto de normas que rige cada tipo de comunicación entre dos computadoras (intercambio de información).

Puerta trasera: Falla de seguridad que permite acceder a una computadora, programa o sistema en general sin usar un procedimiento normal. Estos accesos tienen como objetivo fundamental lograr tanto la salida de información confidencial como introducir daños en el sistema.

Q

R

Riesgo: Posibilidad de que se produzca un impacto determinado en un activo.

S

Servidor: Máquina que ofrece determinados servicios a otras dentro de una red.

Sistema Asimétrico de Cifrado: Sistema mediante el cual se emplea una doble clave kp (privada) y KP (pública). Una de ellas es utilizada para cifrar y la otra para descifrar. El emisor conoce una y el receptor la otra. Cada clave no puede obtenerse a partir de la otra.

Sistema Simétrico de Cifrado: Sistema mediante el cual se emplea la misma clave para cifrar y descifrar. El receptor y el emisor deben conocerlas.

Sniffer: Programa que permite escuchar furtivamente en redes de medios de comunicación compartidos. Se conecta a una PC que está conectada a la red y captura el tráfico de todo el segmento de red.

Sistema informático: Conjunto formado por las personas, computadoras, papeles, medios de almacenamiento digital, el entorno donde actúan y sus interacciones".

T

U

V

Vulnerabilidad: Hueco o debilidad de una aplicación, la cual puede ser una falla de diseño o un bug de implementación.

W

WEP (Wireless Equivalent Privacy**):** Es un método de encriptación de 64 bit o 128 bit con claves configurables por el usuario, usado para proveer seguridad en redes inalámbricas.

WPA (Wifi Protected Acces)**:** Es un método de encriptación de 256 bit con claves que cambian cada cierto tiempo, usado para dar seguridad a las redes inalámbricas.

X

Y

Z