VNIVERSIDAD NACIONAL
AVTNOMA DE
MEXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

INCORPORACIÓN DE LA SEGURIDAD EN EL PROCESO UNIFICADO ÁGIL PARA EL DESARROLLO DE SOFTWARE

T E S I S
QUE PARA OPTAR POR EL GRADO DE:
MAESTRO EN INGENIERÍA (COMPUTACIÓN)

PRESENTA:
PATIÑO CORONA JUAN

DIRECTOR DE LA TESIS:
M.C. MARÍA GUADALUPE ELENA IBARGÜENGOITIA GONZÁLEZ
POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

MÉXICO, D. F. AGOSTO 2013

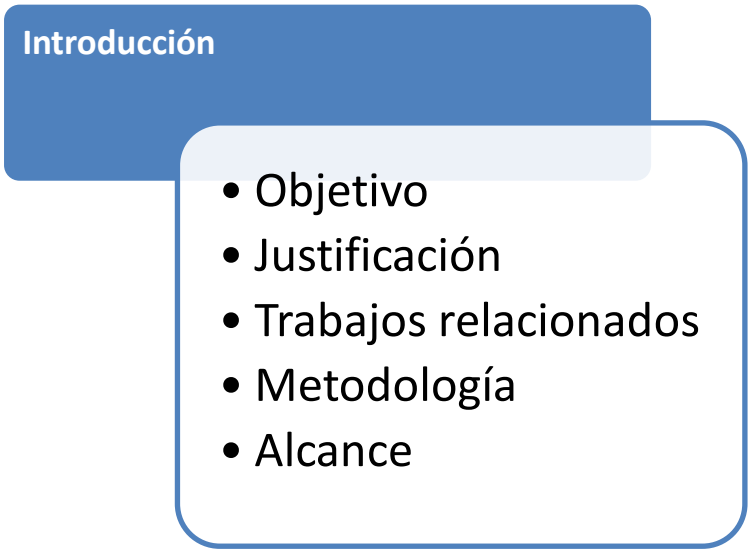
Agradezco a mi familia, quienes me han apoyado con comprensión y sin dudas ni miramientos durante toda mi trayectoria académica, a mi novia, compañera y cómplice de vida, también a mis amigos y compañeros, de quienes aprendo a cada día, a mis honorables profesores, que sin recelo compartieron su tiempo y conocimiento, dentro y fuera de las aulas, a mis sinodales, que critica y sabiamente evaluaron mi investigación, y en especial, agradezco a mi tutora, quien con paciencia y compromiso me acompañó durante esta maravillosa etapa. A todos y a cada uno de ellos, les dedico el esfuerzo y trabajo que este documento representa.

Juan Patiño Corona

ÍNDICE GENERAL

TABLA DE ABREVIATURAS

CERT	Equipo de Respuesta a Incidentes de Seguridad en Cómputo
COBIT	Control Objectives for Information and Related Technologies
IEC	Comisión Electrotécnica Internacional
IDS	Sistema Detector de Intrusos
IPS	Sistema Preventor de Intrusos
ITIL	Information Technology Infrastructure Library
ISO	Organización Internacional de Normalización
OWASP	Proyecto de Seguridad de Aplicaciones Web Abiertas
PU	Proceso Unificado
PUA	Proceso Unificado Ágil
PYMES	Pequeñas Y Medianas Empresas
SGSI	Sistema de Gestión de Seguridad de la Información
TIC	Tecnologías de la Información y las comunicaciones


Introducción
• Objetivo
• Justificación
• Trabajos relacionados
• Metodología
• Alcance

INTRODUCCIÓN

Los procesos de desarrollo de software pueden resultar extensos al ubicarlos dentro del marco de las PYMES (empresas con menos de 25 empleados) relacionadas con el desarrollo de sistemas informáticos. Un gran número de empresas dedicadas a este mercado en México, caen dentro del modelo de negocios pequeños y medianos. Éstas a su vez requieren de procesos que se adapten a sus necesidades, características y posibilidades, lo que impulsa la alternativa de la adopción de metodologías ágiles.

Modelos ya muy maduros como lo es el Proceso Unificado (PU) cuentan ahora con una versión reducida y ágil que permiten adaptarse a empresas PYMES sin un gran costo de transición, o bien ser adoptadas fácilmente por organizaciones recién creadas. Este proceso es muy atractivo para el tipo de mercado de las fábricas de software en México y Latinoamérica.

El Proceso Unificado Ágil (PUA) sigue las mismas fases y un subconjunto de las disciplinas del PU, siendo serial en lo grande e iterativo en lo pequeño, liberando entregables a lo largo del tiempo. Los principios sobre los que se desarrolla el PUA, están basados en una filosofía ágil; en el PUA el personal sabe lo que está haciendo, se busca tener simplicidad, agilidad y enfocar las actividades en un alto nivel (se centra en las actividades que realmente cuentan, no en cada cosa que podría pasarle al proyecto), el PUA brinda una independencia en la selección de herramientas para que éstas, sean las más adecuadas a cada necesidad.

Por su parte, el software representa un conjunto complejo de aspectos de seguridad que deben ser cubiertos por los arquitectos, diseñadores y programadores. Las aplicaciones más seguras y resistentes al hacking son aquellas en las cuales la seguridad se tuvo en cuenta durante todo su proceso de desarrollo. Es necesario abordar el estado de los sistemas, los riesgos y las vulnerabilidades, proponiendo soluciones ante esta problemática, enfatizando que la seguridad no depende exclusivamente de la protección de las redes, usuarios o configuraciones, sino también de las propias aplicaciones.

Múltiples consultorías de seguridad evidencian lo peligroso que resulta la poca atención puesta en la protección, provocando que los usuarios comprometan su información u otros activos al emplearlas. Es esta necesidad la que motiva la propuesta de incorporar un enfoque de seguridad en las metodologías de desarrollo de software explícitamente al Proceso Unificado Ágil (PUA).

Objetivos

El objetivo global del trabajo de tesis es presentar una propuesta de extensión al Proceso Unificado Ágil incorporando prácticas para fortalecer la seguridad dentro del desarrollo de software y que apoyen a la reducción de las vulnerabilidades en las aplicaciones. Emplear

algunas métricas basadas en la calidad para cuantificar los fallos y que estas, ayuden a definir una línea de mejora respecto a la seguridad del producto.

Con esto se busca generar una alternativa, para que las pequeñas y medianas empresas puedan contar con un modelo de desarrollo de aplicaciones seguras, adaptable a sus necesidades, manteniendo la integridad del negocio y proporcionando un valor agregado a la calidad de los productos de software a través de medidas proactivas.

Justificación

En la actualidad, las aplicaciones de software están presentes cada vez con mayor frecuencia en las actividades sociales, económicas, de gobierno, empresariales, etc. Muchos de los sectores estratégicos de un país dependen cada vez más de la tecnología a través de sistemas de software y estos controlan grandes volúmenes de información crítica que es creada, borrada, leída o modificada a cada segundo, lo que motiva la intención de algunas personas u organizaciones por ingresar a estos sistemas informáticos y tomar control de ellos para diversos fines.

La mayoría de los ataques en contra de las aplicaciones son descubiertos y documentados por empresas de seguridad y no por los propietarios de los desarrollos vulnerados. Es imposible asegurar que una aplicación es totalmente invulnerable, ya que cada línea de código, configuración o dato almacenado incrementa el riesgo potencial dirigido hacia los activos que estén al alcance de los sistemas comprometidos, lo cual incluye dispositivos de hardware, software, activos económicos e inclusive la integridad y calidad de vida de un conjunto de personas.

Para poder hacer frente a esta problemática, es necesario trabajar a través de métodos, modelos y normatividades que aborden la seguridad como una característica particularmente importante de la calidad y que sean eficientes y eficaces durante todo el ciclo de vida del software, después de la liberación del producto es igualmente necesario darle un seguimiento, generar el análisis, estudio y pruebas para continuar con el fortalecimiento de la seguridad.

Si bien no es posible asegurar un sistema de software ante todas las amenazas, sí debemos buscar que este riesgo se minimice hasta que el software sea confiable y siga funcionando como se espera y que lo haga en presencia de algún ataque, sin hacer más ni menos las actividades para las cuales se diseñó, registrando la actividad mínima necesaria que permita un rastreo de las acciones realizadas por el perpetrador.

Según el último análisis dado a conocer por el Consorcio de Seguridad en Aplicaciones Web en 2007 (WASC) (1) en el que se sometieron a diversas pruebas cerca de 12,186 aplicaciones, se detectaron 97,554 vulnerabilidades de diferentes niveles de riesgo. El análisis expuso que más del 13% de las muestras pueden ser completamente

comprometidas y cerca del 49% contienen vulnerabilidades con un alto nivel de riesgo (urgente y crítico), así como otros porcentajes de niveles inferiores. En resumen, la probabilidad de detectar vulnerabilidades con un nivel de peligrosidad mayor al medio es superior al 86%.

Trabajos relacionados

Hasta la fecha, existen diversas contribuciones, tanto del sector académico como del industrial, con el objetivo de disminuir el número de fallos y vulnerabilidades, buscando ofrecer garantías objetivas sobre la seguridad del software creado. Para ello se aplican mecanismos que evidencian que el software contiene consistentemente las propiedades de seguridad que se le requieren. Partiendo del aforismo "No existe un sistema completamente seguro", el software seguro es *capaz de resistir la mayoría de los ataques, tolerar aquellos que no pueda resistir y recuperarse con el mínimo impacto de aquellos que no pueda tolerar* (2).

Existen varios modelos y estándares desarrollados sobre las características de calidad, por ejemplo, la reusabilidad, la mantenibilidad, la corrección de errores, integridad, eficiencia y por supuesto la seguridad. Para este último en especial, el interés se ha ido incrementando, surgiendo muchas investigaciones, sin embargo aún falta lograr su aplicación de manera práctica.

El tema de seguridad se enfocaba en un inicio a sistemas considerados como críticos (transporte, servicios, energéticos, etc.), abordando principalmente la manera en la que se administraban los riesgos. En la década de los 80 y 90, a medida que los sistemas de control se utilizaban de forma más extensa, la comunidad de ingenieros de seguridad, desarrolló estándares para la especificación y desarrollo de sistemas de seguridad críticos.

Un ejemplo es el Estándar Internacional para la Gestión de la Seguridad IEC 61508 (3), desarrollado para sistemas como los de control ferroviario y podía emplearse en sistemas generalizados de control, pero resultaba inapropiado en un entorno de sistema de información. En la figura TR.1 Se describen los aspectos de seguridad del estándar pasando desde la planificación, desarrollo y hasta su desmantelación (4).

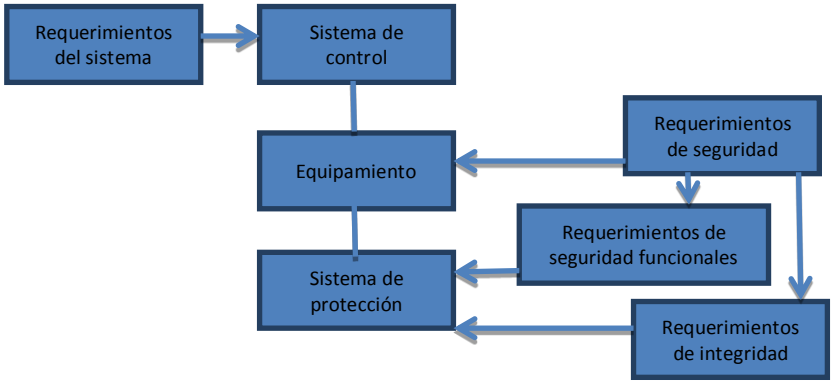


Figura TR. 1 Requerimientos de seguridad de un sistema de control

El siguiente diagrama (Figura TR. 2) es una simplificación de la representación de Redmill del ciclo de vida de la seguridad. En este modelo, el sistema controla dispositivos que tienen asociados requerimientos de seguridad de nivel alto, los cuales generan dos tipos de requerimientos detallados para la protección del sistema (4):

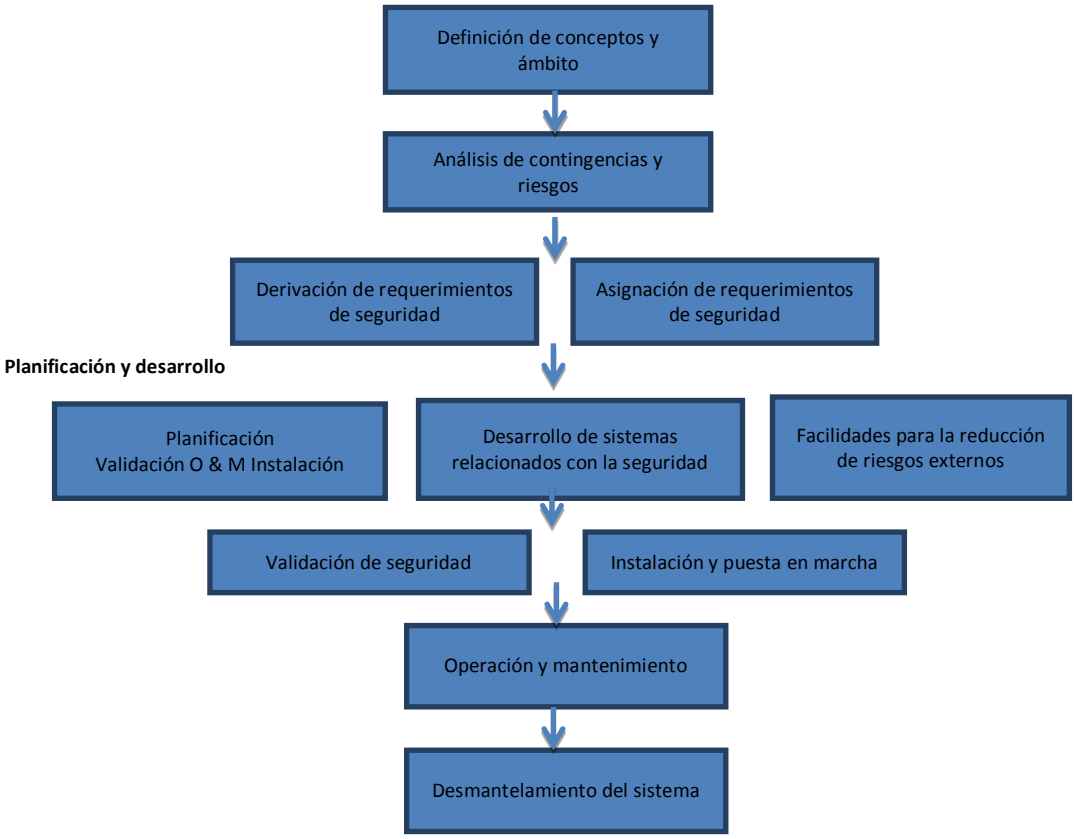


Figura TR. 3 El ciclo de vida de seguridad IEC 61508

Otro de los trabajos realizados sobre el tema de seguridad es el correspondiente a la serie de estándares ISO/IEC 27000, que proporcionan un marco de gestión de la seguridad de la información, utilizable por cualquier tipo de organización, pública o privada, grande o pequeña (5), tomando como artefacto principal al Sistema de Gestión de Seguridad de la Información (SGSI), cuyos requisitos se especifica en el estándar ISO/IEC 27001. Esta colección de estándares considera que la información es un activo vital para el éxito y la continuidad en el mercado de cualquier organización. El aseguramiento de dicha información y de los sistemas que la procesan, es por lo tanto, un objetivo de primer nivel para la organización.

Otras organizaciones han generado proyectos de investigación directamente sobre las aplicaciones, redactando manuales, describiendo buenas prácticas y desarrollando herramientas para evaluar el riesgo y apoyar en la búsqueda de defectos en el software;

algunas de estos proyectos no tienen fines de lucro, tal es el caso del Proyecto de Seguridad de Aplicaciones Web Abiertas (OWASP) el cual ofrece una gran gama de documentos y herramientas enfocadas en la seguridad sobre sistemas web. Muchas empresas de seguridad como Symantec, Kaspersky, organizaciones como los distintos CERTS que existen alrededor del mundo, el SANS Institute o el Consorcio de Seguridad en Aplicaciones Web (WASC), inclusive las mismas desarrolladoras de software como Microsoft o IBM publican continuamente boletines y estudios sobre el estado de la seguridad, las vulnerabilidades predominantes o los vectores de ataque que han adoptado los atacantes.

Systems Security Engineering Capability Maturity Model (SSE-CMM) (6). Es un modelo de referencia para la incorporación de la Ingeniería de Seguridad en las organizaciones, siendo de todos los modelos presentados el que mayor relación tiene respecto al desarrollo de productos de software seguros. El modelo divide a la ingeniería de seguridad en tres áreas básicas: riesgo, ingeniería y aseguramiento, y presenta una serie de actividades para lograr el desarrollo de productos de software confiables, y alcanzar un ciclo de vida para sistemas seguros. La propuesta del SSE-CMM se desarrolla, considerando que la ingeniería de seguridad no es una actividad aislada de otras especialidades de la ingeniería, en especial de la ingeniería de sistemas y de software. (7).

La aplicación de métodos formales para la construcción y verificación de software seguro es una iniciativa relativamente reciente y es capaz de ofrecer las mayores garantías en cuanto a la robustez del software. Ya que estos métodos emplean técnicas que permiten verificar las propiedades y calidad del software, tal y como lo expone Gary McGraw en su libro titulado: *Software Security: Building Security In* (8), en cual contempla la característica de seguridad a lo largo del ciclo del software.

Hay muy poco trabajo concerniente a la integración de aspectos de seguridad desde las primeras fases del desarrollo de software. Aunque se han propuesto ciertas alternativas para la integración de la seguridad, actualmente no hay una metodología completamente definida para ayudar a los desarrolladores que requieran seguridad. Las propuestas no son lo suficientemente completas ni extensas, en el sentido de que se centran en alguna fase en especial como el diseño o la implementación o bien en un aspecto de seguridad particular tal como el control de acceso. Es más, no ofrecen una forma de integrarse en métodos de desarrollo. (9)

Metodología

La metodología a seguir para la elaboración de esta tesis consistió inicialmente en una investigación bibliográfica en libros, revistas y sitios en internet sobre la teoría y situación actual del desarrollo de software seguro, seguido de esto se investigará sobre la normatividad y estándares relacionados.

Una vez organizada esta información se analizaron sus componentes y los trabajos exitosos realizados en el área, para identificar las características generales que deben considerarse durante cada una de las etapas del desarrollo de aplicaciones.

Posteriormente se revisó el Proceso Unificado Ágil para proponer la integración de prácticas seguras que incorporen las características identificadas. Se incluye también, una serie de métricas para analizar cuantitativamente el estado de los desarrollos de la organización y su evolución respecto al número de fallas de seguridad o vulnerabilidades descubiertas durante el ciclo de vida de cada proyecto, con el objetivo de apoyar en la toma de decisiones para la mejora de la calidad del producto.

Alcance

Esta tesis consistirá en el desarrollo de una propuesta teórica, que cubra las principales deficiencias de protección en las aplicaciones de software, y que permita la incorporación en una metodología ágil como una instancia segura de un modelo bien conocido de desarrollo de aplicaciones.

Particularmente este trabajo de tesis se limita al marco de las 10 vulnerabilidades más frecuentes de los sistemas web, consideradas por las características particulares de la plataforma y el gran auge de estos desarrollos, sin embargo las técnicas descritas pueden ser empleadas bajo cualquier otro tipo de software considerando los riesgos, amenazas y peculiaridades de cada proyecto, sin importar la tecnología, plataforma o lenguaje de programación. Trabajos posteriores a ésta tesis podrán profundizar en el desarrollo de una segunda revisión de la metodología, como una versión mejorada en base a los resultados prácticos de su implementación incorporando nuevas medidas de protección, cubriendo los nuevos vectores de ataque o adaptándola a nuevas metodologías de desarrollo; ya que el campo del análisis de vulnerabilidades y producción de aplicaciones son en sí mismos un tema de estudio muy amplio.

Ya que el proceso sobre el cual se basa esta tesis corresponde a una metodología ágil, y para mantener ese carácter, se abordaran exclusivamente las disciplinas de desarrollo y sólo algunos puntos de la administración del proyecto, pues muchos de los problemas de administración referente a la seguridad, pueden solucionarse con las prácticas originales que establece el *Proceso Unificado Ágil (PUA)*.

Así que en el Capítulo 4 se describe el modelado, incorporando los conceptos de casos de uso 2. 0 y rebanadas y diseñando los casos de prueba que luego se implementarán en el los demás capítulos.

De tal forma la tesis se compondrá de dos partes principales: una que introducirá a la teoría del desarrollo de software seguro donde se detalle las características generales y

vulnerabilidades más frecuentes; y otra que describa las adiciones al Proceso Unificado Ágil adaptable al modelo de mercado de fábrica de software para las PyMES en México y Latinoamérica.

Capítulo 1 Seguridad del software (Concepto básicos del problema)

- 1.1 Fundamentos básicos de seguridad informática
- 1.2 La seguridad en los sistemas web
- 1.3 Las 10 vulnerabilidades más comunes en el desarrollo web

1 CONCEPTO BÁSICOS DEL PROBLEMA

La tendencia actual en el desarrollo de sistemas de información es hacia la interconectividad e interoperabilidad entre los sistemas de cómputo, las redes y las organizaciones. Esto ha provocado que la seguridad se convierta en un elemento central y se ha pasado de sólo proteger los datos clasificados de los gobiernos y la milicia, a incluirse en aplicaciones de dominios inimaginables y crecientes, como las operaciones bancarias, comercio electrónico, archivos médicos, etc. Es por ello que la seguridad ha pasado a ser una disciplina cada vez más crítica, necesaria, obligatoria y un componente crucial en cualquier tipo de sistema de información.

La seguridad del software busca que un producto desarrollado continúe funcionando correctamente ante ataques maliciosos, resistiendo a la mayoría de los mismos, tolerando aquellos que no pueda resistir, y recuperándose con el mínimo impacto de aquellos que no pueda tolerar, siendo vista como una medida de robustez de un sistema de software (7).

Es importante aclarar que en ocasiones, el concepto de ***seguridad del software*** se confunde con aspectos de seguridad de la información, lo cual es incorrecto. Si bien es cierto que estos enfoques se encuentran de alguna manera relacionados. Se diferencian en que el objetivo de la seguridad informática se basa únicamente sobre el contexto, de las piezas de software e información, mientras que el enfoque que se busca dar sobre la seguridad del software, abarca todos los aspectos del producto incluyendo el proceso de desarrollo.

El objetivo de este capítulo es exponer algunos conceptos fundamentales empleados a lo largo de la presente tesis, lo cual ayudará a comprender los temas de seguridad de la información para ser aplicados al software.

1.1 Seguridad informática

La seguridad informática plantea un conjunto de conceptos particulares, cuyo conocimiento es necesario para la correcta interpretación del contexto en el que se aplican. A continuación se describe un subconjunto de los mismos, con el objetivo de unificar el lenguaje que se manejará a lo largo de la presente tesis.

La meta de la seguridad de la información es apoyar a que una organización cumpla con todos sus objetivos de negocio o misión, implementando sistemas que tengan un especial cuidado hacia los riesgos relativos a las TIC (Tecnologías de la Información y las comunicaciones) de la organización, a sus socios comerciales, clientes, administración pública, proveedores, etc. (10).

A continuación se describen los objetivos principales de la seguridad:

- **Integridad:** Se refiere a que la información permanezca inalterada a menos que una entidad autorizada lo modifique. La integridad impacta tanto a los datos como al sistema.
- **Disponibilidad:** Consiste en que la información debe estar accesible en el momento que se requiera y respetando los criterios establecidos para el manejo de la misma por el personal autorizado y debidamente autenticado.
- **Confidencialidad:** Es definida como la capacidad de la información para que sólo pueda ser leída o duplicada por los usuarios autorizados.
- **Responsabilidad a nivel individual o registro de auditoría:** Es el registro que permita la traza de las actividades realizadas por una entidad. Este objetivo de seguridad soporta el no repudio, la disuasión, el aislamiento de fallos, la detección y prevención de intrusos, la generación de evidencia y deslinde de responsabilidades.
- **Confiabilidad (aseguramiento):** Es la garantía de que los cuatro objetivos anteriores se han cumplido adecuadamente. Es la base de la confianza en que las medidas de seguridad, tanto técnicas, como operacionales, funcionan tal y como se idearon para proteger el sistema y la información que procesa.

Los cuatro objetivos primeramente descritos, se cumplen mientras el sistema tenga todas sus funcionalidades correctamente implementadas y estas cumplan con la suficiente calidad. Cuando esto ocurre se logra el objetivo de confiabilidad y debe de existir a lo largo de todo el ciclo de vida del sistema.

Los objetivos conservan una relación mutua. Por ejemplo: La confidencialidad guarda una dependencia con la integridad, si esta última se pierde, no hay forma de garantizar la validez de los mecanismos de confidencialidad, ya que se ha podido acceder ilegalmente a la información. De manera recíproca, la integridad depende de la confidencialidad, por ejemplo en el caso de la divulgación de una contraseña, los mecanismos de integridad podrían ser saltados.

El principal objetivo de la seguridad informática y en general de cualquier tipo de seguridad, es implementar una protección particular a través de un análisis, donde se dé respuesta puntual a tres preguntas que iremos explicando a lo largo de este capítulo.

¿Qué se quiere proteger?
¿De quién se quiere proteger?
¿Cómo se va a proteger?

A continuación se define una serie de conceptos bajo el contexto de seguridad de la información, útiles para una correcta interpretación de este trabajo de tesis.

1.1.1 Dato

Un dato es una información breve y concreta que representa una condición o situación de un sujeto o idea más amplia (11). Se Considera un dato, como una representación simbólica de algún atributo de un objeto sin considerar su importancia, por ejemplo la altura de un edificio, el nombre de alguna persona, la marca de alguna mercancía, etc.

1.1.2 Información

Se define a la información como cualquier señal que recibamos a la que asociemos un significado y aumente nuestro conocimiento (11). De esta manera se considera información a un dato cuando posee cierto valor o importancia para quien lo recopila y lo procesa. Por ejemplo el nombre de una persona podría ser insignificante y considerado como sólo un dato para alguna organización mientras que para otra podría significar parte fundamental de la información de algún cliente.

1.1.3 Activo. Respuesta a la pregunta ¿qué se quiere proteger?

Conjunto de todos los bienes y derechos con algún valor, que son propiedad de una empresa, institución o individuo, y que se reflejan en su contabilidad (12). Es cualquier objeto de importancia relativa a quien los posee. Por ejemplo:

- Personal
- Información
- Mobiliario
- Edificaciones
- Prestigio

Los activos dan respuesta a una de las tres preguntas ¿qué se quiere proteger?. Pues estos representan un objeto de importancia y por lo tanto el objetivo de la protección.

1.1.4 Amenaza

Se puede definir como amenaza a todo elemento o acción capaz de atentar contra la seguridad de la información.

Las amenazas surgen a partir de la existencia de vulnerabilidades, es decir que una amenaza sólo puede existir si existe una vulnerabilidad que pueda ser aprovechada, e independientemente de que se comprometa o no la seguridad de un sistema de información. (13)

Por lo que una amenaza es todo aquello que puede, intenta o pretende destruir, se encuentra latente y aun no se ha manifestado. Es la posibilidad de que ocurra un evento que comprometa la información o cualquier otro bien o activo de interés para la organización.

Existen muchas fuentes de amenaza a las que los usuarios se encuentran expuestos al encender su computadora o conectarse a una red y que pueden comprometer la información entendiendo a ésta como uno de los principales y más importantes bienes.

1.1.4.1 Tipos de amenazas

1.1.4.1.1 Amenaza humana

Se refiere a posibles amenazas que involucran directamente al factor humano, como por ejemplo:

- Un exempleado insatisfecho que conserva acceso y privilegios en la organización.
- Terrorismo.
- Curiosidad y experimentación fuera de un ambiente controlado y destinado a ello.
- Ingeniería social.
- Deficiente capacitación

Los anteriores ejemplos son sólo unas cuantas de las amenazas que pueden producirse a través del factor humano, el número total de ellas puede llegar a ser incalculable.

1.1.4.1.2 Amenaza de software (Malware)

Esta palabra nació de la unión de dos palabras de origen inglés: malicious software y cuya traducción literal es software malicioso. Engloba cualquier cantidad de programas desarrollados con el fin de dañar, sacar provecho y en general afectar a los sistemas informáticos (14). Cabe mencionar que los daños que estos códigos maliciosos pueden provocar, van desde hacer que una aplicación no funcione correctamente, robo de información, la falla del hardware, hasta acciones que representen pérdidas económicas para los usuarios o para la organización en general. Entre este tipo de amenazas encontramos:

- Virus
- Gusanos
- Puertas traseras
- Key logers
- Bots

- Exploits
- Rootkit
- Spyware
- Caballos de Troya

Día con día se generan nuevas técnicas y programas que los atacantes emplean para sus diversos fines y pueden ser tan variadas e innovadoras como lo sea la creatividad del perpetrador.

1.1.4.1.3 Amenazas de hardware

Son fallas en los dispositivos de hardware, errores físicos que pueden presentarse por cuestiones de diseño o por el mismo uso y desgaste del equipo. En algunos casos pueden estimarse si se considera el tiempo de vida que los fabricantes proporcionan, así como las condiciones de uso, pero es necesario recalcar que es solo una estimación y no siempre es posible realizarla. (15)

Estos errores puedes suscitarse por una variada cantidad de factores, desde cambios en el voltaje, variaciones en los valores de corriente eléctrica, desgaste en partes mecánicas, etc. A continuación se presentan algunos ejemplos de este tipo de amenazas:

- Falla de lectura o escritura en una unidad de disco duro u óptica
- Falla en los dispositivos periféricos
- Errores de fabricación

1.1.4.1.4 Amenaza de red

Las redes de cómputo son complejos sistemas de hardware y software que hacen posible la comunicación de dos o más computadoras. Cuanto mayor sean las redes estas se vuelven sistemas completamente caóticos. Por lo tanto siempre existe la posibilidad de registrar fallas o errores. Un ejemplo de ello son las colisiones entre los paquetes de información, fallas físicas o lógicas en los medios de comunicación, etc.

1.1.4.1.5 Amenazas naturales

Son amenazas que se generan por fenómenos naturales, también conocidos como actos de Dios, la mayoría son prácticamente impredecibles y sólo se puede intentar generar planes para minimizar los riesgos. (14) Por ejemplo:

- Incendios

- Terremotos
- Inundaciones
- Tornados
- Tormentas
- Erupciones volcánicas

Cabe resaltar que los fenómenos naturales pueden llegar a ser los menos controlables por su gran impacto, poca predicción y alto poder de destrucción, lo cual hace necesario que se ponga igual atención en los métodos de protección de la información y bienes en general, tanto de manera lógica como la protección física en el sitio. Así mismo generar planes de contingencia para reducir el impacto una vez que la amenaza se ha concretado.

1.1.5 Vulnerabilidad

Una vulnerabilidad es un hueco, una falla o una parte a lo que no se le dio suficiente atención o importancia. Es un punto débil de un sistema o equipo de cómputo, tanto de hardware como de software. Puede entenderse como una potencialidad o posibilidad de ocurrencia de la materialización de una amenaza sobre dicho activo. Las vulnerabilidades asociadas a los activos, incluyen las debilidades en el nivel físico sobre la organización, los procedimientos, el personal, la gestión, la administración, los equipos, el software o la información. Una vulnerabilidad por sí misma, no causa daño alguno, es simplemente una condición o conjunto de condiciones que puede permitir la materialización de una amenaza de intrusión. (10)

Día a día se publican un gran número de vulnerabilidades en aplicaciones, sistemas operativos o inclusive protocolos y servicios, fallos que pueden comprometer la información atentando contra cualquiera de los objetivos de la seguridad (disponibilidad, integridad y confidencialidad). Es importante comprender que por la complejidad de los sistemas, su extensa difusión y la amplia interacción con otros sistemas y personas, las vulnerabilidades siempre van a existir y hay que partir desde la máxima: “no hay sistema completamente seguro”. Y por tanto, hay que diseñar un plan de contingencia o plan de recuperación para cuando falle el sistema, no por si falla el sistema. (16)

A continuación se listan algunas vulnerabilidades consideradas generales y relativamente comunes que podemos encontrar en sistemas de cómputo, con el objetivo de ilustrar el concepto de vulnerabilidad. Sin embargo más adelante se describirá un listado de las diez vulnerabilidades más comunes en los sistemas web y sobre las cuales se describe el alcance de esta tesis:

- Desbordamiento de buffer (buffer overflow)
- Inyección de sentencias SQL (SQL injection)
- Heap-overflow
- Vulnerabilidad de Formato de Cadenas (FormatString)

En la figura SI.1 se describe estas relaciones que hay entre los diferentes elementos de la seguridad.

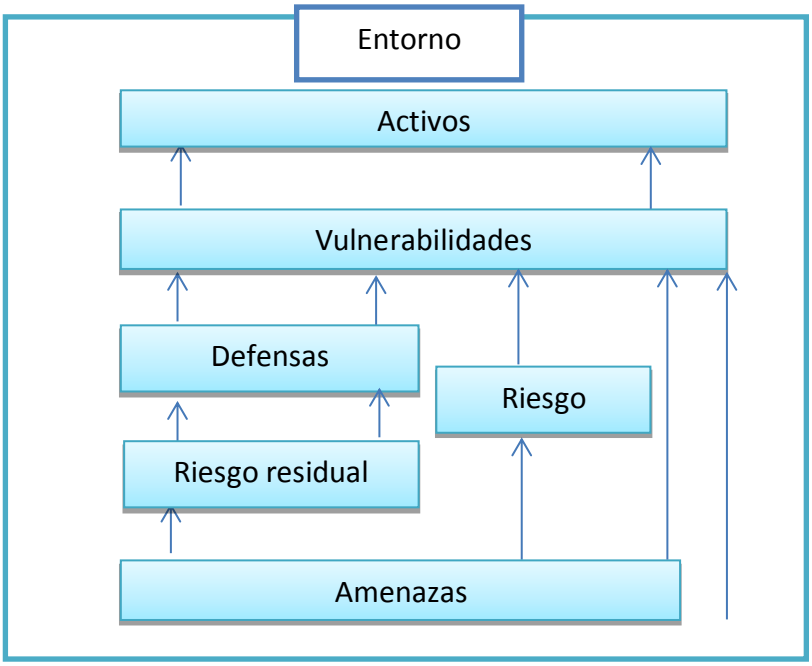


Figura SI. 1 Relación entre elementos de seguridad (10)

1.1.6 Atacante. Respuesta a la pregunta ¿de quién se quiere proteger?

Definimos como ***atacante*** a toda aquella persona que atenta de manera no autorizada en contra nuestros bienes y alguno de los objetivos de la seguridad. Se dedican a quebrantar sistemas de seguridad para poder acceder a partes prohibidas o conseguir información privilegiada. (17)

Los atacantes conforman la respuesta a la segunda pregunta ¿de quién se quiere proteger? siendo los objetivos de sus ataques muy variados, entre los que podríamos listar los siguientes:

- Obtener reputación como atacante
- Dañar la reputación de alguna persona u organización
- Obtener bienes lógicos o físicos
- Beneficios económicos
- Generar daños a bienes o servicios
- Robo o alteración de la información

Los atacantes o también llamados ***perpetradores*** están divididos en varias categorías según su experiencia, conocimiento y objetivos (17). A continuación se menciona sólo algunos de los grupos más comunes:

- **Hacker.** Persona que disfruta de la tecnología de computación y dedica tiempo a aprender y usar los sistemas de computación (18). Intenta tener acceso no autorizado o ilegal a sistemas de cómputo con el objetivo de evidenciar sus puntos débiles, para que estos sean corregidos, mejorados.
- **Cracker.** A diferencia del hacker, rompe sistemas y se aprovecha con fines ilícitos de la información obtenida (19).
- **Lámer.** En el medio es considerado un ignorante, en sus manos puede tener mucha información, pero no comparte nada con nadie. Cuando irrumpe en un sistema lo hace para fastidiar a los usuarios. Contrario a la mayoría de los demás atacantes (20).
- **Scripkidie.** Una especie de cracker, pero de baja categoría. Un individuo con experiencia técnica limitada. Sus herramientas son sencillas; muchas veces personalizadas, pero con ellas puede ejecutar actividades en contra de las redes (20).

1.1.7 Respuesta a la pregunta. ¿Cómo se va a proteger?

Es indispensable planificar una estrategia de seguridad. Se debe ser consciente de las herramientas y metodologías con las que se cuenta para la protección de los activos. Plantear una estrategia significa considerar el nivel de protección que requerimos y emplear de manera responsable los recursos con los que contamos.

La seguridad informática va desde generar políticas y reglas de uso sobre el equipo y las redes así como instalar seguridad por medios físicos y lógicos posterior a haber realizado el análisis del riesgo, respondiendo dos preguntas planteadas anteriormente: ¿qué se quiere proteger? y ¿de quién se quiere proteger?.

1.1.7.1 Seguridad en sitio

Es uno de los principales puntos donde hay que considerar la protección, cuidar el sitio o un equipo de cómputo de manera particular y personalizada resulta tan importante como proteger el perímetro de una red.

Existen muchos factores a considerar para brindar protección al sitio, entre ellas podemos enumerar la selección del sistema operativo manteniéndolo actualizado, la instalación de aplicaciones estrictamente necesarias y en sus últimas versiones, implementación de herramientas antimalware, IDS a nivel host, etc. De tal forma que nos ayudarán a identificar actividad poco frecuente, o un firewall local para controlar el tráfico que entra

o sale de nuestro equipo. Es importante también controlar la apertura de puertos rigurosamente indispensables y el levantamiento únicamente de servicios obligatorios, monitoreo frecuente del sistema, cumplimiento de las políticas de uso y principalmente un profundo conocimiento del sistema del cual se es usuario o administrador.

1.1.7.2 Seguridad perimetral

Definimos **seguridad perimetral** como la protección de los puntos donde nuestra red de confianza tiene contacto con otras redes que no controlamos, así pues son los métodos y dispositivos implementados para controlar y restringir el tráfico de entrada o salida de nuestra red. Estos dispositivos pueden ser Firewalls, IDS o IPS. (10)

1.1.7.2.1 Firewall

Son dispositivos de seguridad perimetral que controlan el tráfico de entrada y salida de una red, estos dispositivos leen la información de los paquetes y restringen o permiten su paso en cualquier dirección según la configuración dada a través de sus reglas. Existen tres tipos principales de firewalls que varían en su nivel de complejidad.

- **Filtrado de paquetes:** es el tipo más básico de firewall, lee las cabeceras de los paquetes y permite el paso o no según parámetros como la dirección *ip* de origen, destino, puerto origen o puerto destino o protocolo.

- **Estados:** es la evolución del firewall de filtrado de paquetes, donde además de leer los parámetros de las cabeceras también lee el estado de las conexión.

- **De aplicación o proxys:** es el tipo más complejo que podemos encontrar, donde además de leer las cabeceras de los paquetes también se lee el contenido, esto permite un mayor nivel de protección en contra de paquetes maliciosos que podrían resultar válidos en los dos tipos anteriores, la principal desventaja es la reducción en el performance o comportamiento de la red y su alto costo económico.

1.1.7.2.2 IDS

Sistema detector de intrusos o IDS por sus siglas en inglés. Su principal función es revisar cada paquete en la red y compararlo con una base de datos que contiene información de ataques maliciosos, en caso de que el paquete revisado concuerde con alguno de la base, entonces el IDS envía una alerta informando la posible existencia de código peligroso.

1.1.7.2.3 IPS

Sistema Preventor de Intrusos o IPS por sus siglas en ingles. Este dispositivo es una evolución de los IDS, donde además de enviar un alerta ante la presencia de código malicioso, toma de manera automática medidas para restringir el tráfico malicioso, podríamos decir que un IPS es la combinación de un IDS y un firewall.

1.1.8 Ataque informático

Los ataque informáticos son acciones deliberadas y generadas por una o más personas que causa un daño o problemas en sistemas de cómputo atentando en contra de uno o varios de los objetivos de la seguridad (confidencialidad, disponibilidad e integridad). (15)

1.1.9 Ataque pasivo

Un ataque pasivo se caracteriza por no alterar en ningún momento la información, lo que lo hace especialmente difícil de detectar. Por lo regular se le considera como el fisgoneo de datos, un ejemplo es el monitoreo de tráfico en la red. (15)

1.1.10 Ataque activo

A diferencia de un ataque pasivo, el ataque activo se caracteriza por realizar la modificación de la información, este ataque es relativamente más fácil de identificar. Por ejemplo el pharming un tipo de ataque que se basa en alterar los registros DNS con el objetivo de re-direccionar ilícitamente peticiones que hagan uso de este servicio. (15)

1.1.11 Análisis forense

El análisis forense es una técnica de estudio sobre equipos ya comprometidos o vulnerados, el objetivo es comprender el ataque, corregir o restablecer los bienes dañados, identificar responsables y preparar un método de protección para futuros ataque con características semejantes. Como una analogía, podríamos considerarlo como el equivalente a una autopsia humana. Definiéndose como el proceso mediante el cual se identifican, preservan, analizan y presentan las evidencias digitales de manera que sean aceptables en una vista judicial o administrativa. (10)

1.1.12 Ingeniería inversa

Es un método de análisis sobre elementos ya existentes de los cuales carecemos de información sobre su creación, el objetivos de la ingeniería inversa es comprender el funcionamiento de algún sistema (no sólo informático), partiendo en dirección cronológicamente opuesta, esto es que iniciamos con el sistema funcionando, completamente formado y poco a poco vamos descomponiendo y analizando cada una de

sus partes hasta llegar a los elementos más básicos y elementales de su creación siendo entonces capaces de reproducir total o parcialmente el diseño.

1.2 La seguridad en los sistemas web

Una aplicación web es un tipo especial de aplicaciones cliente/servidor, donde tanto el cliente como el servidor y el protocolo mediante el que se comunican (HTTP) están estandarizados y no han de ser creados por el programador de aplicaciones.

El protocolo HTTP forma parte de la familia de protocolos de comunicaciones TCP/IP, que son los empleados en Internet. Estos protocolos permiten la conexión de sistemas heterogéneos, lo que facilita el intercambio de información entre distintas computadoras. HTTP se sitúa en el nivel 7 (aplicación) del modelo OSI.

Las aplicaciones web se emplean en tres entornos informáticos muy similares que suelen confundirse entre sí: Internet, Intranet y Extranet. Una *Intranet* es una red corporativa o local a una organización que está estructurada en el conjunto de tecnologías que sustentan la Internet; una *Extranet* no es más que una intranet que una organización " abre" hacia la web de manera que parte de los recursos de la organización quedan accesibles para usuarios externos a la misma. (21)

Los desarrollos web conforman un conjunto amplio entre las aplicaciones de cómputo actuales. Este conjunto de tecnologías de software trabajan del lado del servidor y del cliente, involucran una combinación de procesos entre bases de datos y aplicaciones de terceros con el uso de un navegador a fin de realizar determinadas tareas o mostrar información.

Su amplia popularidad y exposición, las han convertido en un objetivo para los atacantes, tanto como una puerta de acceso a la infraestructura de las organizaciones, como una fuente de información para futuros ataques. Es de esta forma que una aplicación web puede ser el objetivo principal o un medio para que un perpetrador cumpla sus cometidos.

1.3 Las 10 vulnerabilidades más comunes en el desarrollo web

OWASP un proyecto global dedicado a la investigación y mejora de la seguridad en aplicaciones web, publica de manera periódica una lista con las diez vulnerabilidades más frecuentes, en donde se concentra la mayor parte del riesgo, estas vulnerabilidades son detectadas en todo tipo de desarrollos web sin importar el rubro o sector al que se dirijan.

La lista publicada por OWASP será el punto de partida para la cual se enfocarán los ejemplos en esta tesis, sin embargo la metodología que se propone puede aplicarse para cubrir un espectro mayor y a discreción de los involucrados en cada proyecto en particular.

Para cada elemento en la lista, los factores generales y las consecuencias de cada una se analizarán en los capítulos siguientes.

El objetivo de OWASP al publicar su lista de vulnerabilidades es educar a los desarrolladores, diseñadores, arquitectos, gerentes y organizaciones sobre las consecuencias de las debilidades de seguridad más comunes. Sin embargo, pueden plantearse las se consideren adecuadas según las condiciones y detalles del medio en particular para cada organización.

En la Tabla SS .1 se describen el listado de las 10 vulnerabilidades más comunes, publicado por el proyecto OWASP (22):

Vulnerabilidad	Descripción
1.Inyección de código	Las fallas de inyección tales como SQL, sistema operativo y la inyección LDAP, ocurren cuando los datos no confiables son enviados a un intérprete como parte de un comando o consulta. Datos hostiles del atacante puede engañar al intérprete para que ejecute comandos no deseados o de acceso a los datos no autorizadas.
2.Cross Site Scripting XSS	Las fallas de XSS ocurren cuando una aplicación toma los datos que no son de confianza y lo envía a un navegador web sin necesidad de una validación adecuada. XSS permite a los atacantes ejecutar scripts en el navegador de la víctima que pueden secuestrar sesiones de usuario, modificar sitios Web, o redirigir al usuario a sitios maliciosos.
3.Pérdida de autenticación y Gestión de sesiones	Ocurre cuando funciones de aplicación relacionada con la autenticación y administración de sesiones no se aplican correctamente, permitiendo a los atacantes comprometer contraseñas, claves, tokens de sesión, o explotar otras fallas de implementación para asumir la identidad de otros usuarios.
4.Referencia Directa Insegura a Objetos	Una referencia directa ocurre cuando un desarrollador expone una referencia a un objeto de implementación interna, tal como un archivo, directorio o base de datos de claves. Sin un control de accesos u otro tipo de protección, los atacantes pueden manipular esas referencias para acceder a datos no autorizados.

5.Falsificación de Peticiones en Sitios Cruzados(CSRF)	Un ataque CSRF fuerza una sesión iniciada en el navegador de la víctima para enviar una petición HTTP, incluyendo la cookie de sesión de la víctima y cualquier otro dato que incluye automáticamente la autenticación, a una aplicación web vulnerable. Esto permite al atacante forzar al navegador de la víctima para generar solicitudes de la aplicación vulnerable la cual asume que son peticiones legítimas de la víctima.
6.Defectuosa configuración de seguridad	Una buena seguridad requiere tener una configuración segura definida e implementada para la aplicación, los entornos de: servidor de aplicaciones, servidor web, servidor de base de datos, y la plataforma. Todos estos ajustes se deben definir, implementar y mantener. Esto incluye el mantener todo el software actualizado, incluyendo todas las bibliotecas de código que utiliza la aplicación.
7.Almacenamiento criptográfico inseguro	Muchas aplicaciones web no funcionan bien al proteger los datos confidenciales, tales como tarjetas de crédito, números de Seguro Social o las credenciales de autenticación con un cifrado apropiado o hash. Los atacantes pueden robar o modificar dichos datos débilmente protegidos para llevar a cabo el robo de identidad, fraudes u otros delitos.
8.Falla de restricción de Acceso a URL	Muchas aplicaciones web comprueban los derechos de acceso a las URL antes de emitir enlaces y botones. Sin embargo, las aplicaciones necesitan llevar a cabo el control de acceso similar al comprobar cada vez que se accede a estas páginas, o los atacantes serán capaces de formar las URL para acceder a las páginas ocultas.
9.Protección Insuficiente en la capa de Transporte	Las aplicaciones frecuentemente fallan para autenticar, cifrar y proteger la confidencialidad y la integridad del tráfico de red sensible. Cuando lo hacen, a veces, se apoyan en algoritmos débiles, certificados caducados o no válidos o no se usan correctamente.
10.Redirecciones y Reenvíos no validados	Las aplicaciones Web frecuentemente redirigen y envían a los usuarios a otras páginas y sitios web, utilizando datos no confiables para determinar las páginas de destino. Sin una correcta validación, los atacantes pueden redirigir a las víctimas de phishing o de sitios de malware, o usarlas para acceder a páginas no autorizadas.

Tabla SS. 1 Las 10 vulnerabilidades más frecuentes en los desarrollo WEB (22)

Capítulo 2 Las PYMES y el Proceso Unificado Ágil

- 2.1 Las PYMES definición y características
- 2.2 El Proceso Unificado Agil
- 2.3 Métricas de seguridad y calidad del software
- 2.4 Casos de uso 2.0

2. LAS PYMES Y EL PROCESO UNIFICADO ÁGIL

A lo largo de este capítulo se examinarán las características generales y la definición del concepto de PYMES. De igual forma se describirá el Proceso Unificado Ágil (PUA), base sobre la cual se desarrolla este trabajo de tesis.

2.1 Las PYMES definición y características

2.1.1 PYMES

Las pequeñas y medianas empresas (PYMES) constituyen la columna vertebral de la economía nacional por su alto impacto en la generación de empleos y en la producción nacional. Un conjunto aún más pequeño pero que coparte la importancia dentro de la economía nacional es el de las micro empresas (empresas con menos de 10 empleados y un monto de ventas anuales de hasta 4 millones de pesos (23)), que junto con las pequeñas y medianas empresas, se establece el concepto de las MIPYMES (Micro, pequeñas y medianas empresas)

De acuerdo con datos del Instituto Nacional de Estadística y Geografía INEGI, en México existen aproximadamente 4 millones 15 mil unidades empresariales, de las cuales 99.8% son MIPYMES que generan 52% del Producto Interno Bruto (PIB) y 72% del empleo en el país (24).

2.1.2 Definición

La palabra “PYMES” está formada por las primeras letras de los conceptos pequeñas y medianas empresas.

La clasificación del tamaño de las empresas no es la misma en todo el mundo, en la Tabla PPUA. 1, se exhibe una comparación entre las diferencias que hay en la definición del tamaño de las empresas, según el número de trabajadores que laboren en ella, tomando como referencia las siguientes instituciones: el Instituto Nacional De Estadística Y Estudios Económicos en Francia (INSEE), la *Small Business Administrations* de Estados Unidos (SBA), la Comisión Económica Para América Latina (SEPAL), la revista mexicana de Ejecutivos De Finanzas (EDF), y finalmente la Secretaria de Economía de México (SE).

Institución	Tamaño de la empresa	No. De trabajadores
Instituto Nacional De Estadística Y Estudios Económicos (Francia)	Pequeña	50 a 250

	Mediana	250 a 1000
Small Business Administrations (Estados Unidos de América)	Pequeña	Hasta 250
	Mediana	250 a 500
Comisión Económica Para América Latina	Pequeña	5 a 49
	Mediana	50 a 250
Revista mexicana de Ejecutivos De Finanzas (México)	Pequeña	Menos de 25
	Mediana	50 a 250
Secretaria de Economía(México)	Pequeña	16 a 100
	Mediana	101 a 250

Tabla PPUA. 1 Clasificación de PYMES por número de trabajadores (25)

En la Tabla PPUA. 2 se describe un criterio de la Secretaria de Economía (26), donde estratifica a las empresas según su actividad productiva.

Actividad / Tamaño de la empresa	**Industriales**	**Comerciales**	**Servicios**
Pequeña	25 a 100 empleados	25 o menos empleados	21 a 50 empleados
Mediana	101 a 500 empleados	21 a 100 empleados	51 a 100 empleados

Tabla PPUA. 2 Clasificación de las empresas según su actividad económica (26)

2.1.3 Características de las Pymes

Hay puntos que caracterizan el comportamiento de las PYMES, algunos de los cuales se citan a continuación (25):

- El capital es proporcionado por una o dos personas que establecen una sociedad.
- Los propios dueños dirigen la marcha de la empresa.
- La administración se hace de manera empírica.
- Dominan y abastecen un mercado más amplio, aunque no necesariamente tiene que ser local o regional, ya que muchas veces llegan a producir para el mercado nacional e incluso para el mercado internacional.

- Están en proceso de crecimiento, las pequeñas empresas tienden a ser medianas y aspiran a ser grandes.
- Obtienen algunas ventajas fiscales por parte del Estado que algunas veces las considera causantes menores dependiendo de sus utilidades.
- Su tamaño es pequeño o mediano en relación con las otras empresas que operan en el ramo. De manera importante existen tres rasgos que merecen ser contemplados:

    1. Un capital social repartido entre unos cuantos socios
    2. Una autonomía real del financiamiento y gestión
    3. Un nivel de desarrollo dimensional

Por la importancia de las PYMES es necesario instrumentar acciones para mejorar el entorno económico y apoyar directamente a las empresas, con el propósito de crear las condiciones que contribuyan a su establecimiento, desarrollo y consolidación. Entonces se hace necesario exponer las tendencias que produce un mercado tan competitivo. Prácticamente todas las empresas se están renovando y las PYMES no son la excepción. Las empresas que buscan al menos mantenerse rentables, necesitan llevar a cabo esta renovación. Y un buen camino para lograrlo es a través de proyectos de transformación, de reingeniería de procesos, de certificación de calidad (ISO's), de adquisición e implantación de nuevas tecnologías, etc. Y en el caso de las empresas productoras de software se incluye el adoptar una metodología o proceso de desarrollo (27).

Hoy en día todas las empresas requieren llevar a cabo más y más proyectos además de continuar con la operación diaria lo más eficientemente posible. El reto para la pequeña y mediana empresa es una modernización inteligente que la lleve a elevar su eficiencia y su productividad. Una manera de lograrlo es mediante la selección y ejecución inteligente de proyectos de modernización: Todo gasto o inversión debe ser altamente productivo, debe generar valor y ese valor debe poder hacerse visible para la empresa (27).

2.2 Proceso Unificado Ágil (PUA)

El Proceso Unificado Ágil (PUA) es un derivado del Proceso Unificado de Rational (PUR). El PUA fue desarrollado por Scott Ambler entre 2002 y 2006 (28) y combina algunos de los flujos centrales de PUR. Las fases son parecidas a las del PUR y los principios del PUA son muy similares a los principios de otros enfoques ágiles, como agilidad, simplicidad, primero los requerimientos prioritarios y confianza en el equipo.

Scott Ambler ha tomado el concepto de desarrollo dirigido por el modelado (MDD) y lo trajo a un nivel ágil. El modelado es una pieza importante de PUA, donde éste es desarrollado en forma iterativa.

El desarrollo dirigido por el modelado ágil (MADD) es un concepto tomado del enfoque dado por el desarrollo dirigido por pruebas lo que se vuelve modelado dirigido por pruebas. Las estimaciones del equipo inician con un modelado muy limitado pero lo suficientemente bueno para que se puedan realizar cálculos aproximados de tiempos de desarrollo. En las iteraciones se implementa el modelo en el código, se mejora el modelo y así sucesivamente. (29)

2.2.1 Principios del PUA

A continuación se describe una serie de principios regidores del PUA motivados en la filosofía de los procesos ágiles (30):

a) *El personal necesita saber lo que está haciendo*. La gente no va a leer la documentación del proceso en detalle, sino que quieren una orientación de alto nivel y/o formación de vez en cuando.

b) Simplicidad. Todo se describe concisamente usando unas páginas, no miles de páginas.

c) *Agilidad*. El PUA se ajusta a los valores y principios de la Alianza Ágil.

d) *Centrarse en las actividades importantes*. La atención se centra en las actividades que realmente cuentan.

e) *Independencia de las herramientas*. Es recomendable utilizar herramientas que mejor se adapten para el trabajo, que a menudo son herramientas simples o incluso herramientas de código abierto.

f) Querer adaptar este producto para satisfacer sus propias necesidades. *Seguir los principios que propone el PUA que mejor se ajusten a cada equipo y proyecto.*

El modelado ágil define un número de buenas prácticas (29):

- **Activa participación de los involucrados.** El cliente o el dueño del producto necesita estar fuertemente involucrado en el proyecto para proporcionar la información y los detalles que el equipo desarrollador necesite.

- **Identificación de requerimientos.** En el comienzo del proyecto, el equipo, con los dueños del producto debe conocer de los requerimientos en orden de prioridad.

- **Priorizar los requerimientos.** Construir una lista de priorización de requerimientos, e iniciar con la implementación de los requerimientos que encabecen la lista.

- **Visión de la arquitectura**. Cuando el equipo inicia, necesita ponerse descuerdo con los involucrados en una arquitectura de alto nivel.

- **Documentación final.** La documentación es importante y necesita hacerse, pero debe construirse en un punto donde esta documentación será sólida y asegurándose de que sufrirá pocos cambios.

- **Especificaciones ejecutables.** Define las especificaciones con las historias de usuario que permitan una fácil traducción en los casos de prueba

- **Modelado iterativo.** Si una iteración cumple sus metas, publicando una nueva versión del producto que implemente ciertos casos de uso, el desarrollo continúa con la siguiente. Cuando no las cumple, se deben revisar las decisiones previas y probar un nuevo enfoque.

- **Sólo los artefactos necesarios.** El modelado debe ser exactamente en el nivel que se necesita para la siguiente iteración.

- **Modelar una parte adelante.** Al tiempo que se revisa la lista de prioridades para ver cuál será la siguiente iteración se debe de asegurar que el modelo permite una fácil adición de esos requerimientos.

- **Múltiples modelos.** Elegir los tipos de modelos que necesite el proyecto.

- **Revisar el modelo.** Antes de implementar la siguiente pieza de código, revisar el modelo y mejorarlo a medida que se avanza.

- **Diseño dirigido por pruebas.** Desarrollar los casos de prueba antes de iniciar la codificación funcional.

2.2.2 Ciclo de vida del PUA

El ciclo de vida del PUA es secuencial en lo grande e iterativo en lo pequeño, liberando incrementos de los entregables en cada iteración. La Figura PPUA. 1 describe el ciclo de vida del PUA incluyendo sus fases y flujos de trabajo, así como la relación de intensidad.

El PUA combina los flujos de modelado del negocio, requerimientos, análisis y diseño, dentro de un flujo de modelado. La razón de mover la administración de cambios dentro del modelado es que en los proyectos ágiles, los requerimientos son revisados en el comienzo de cada iteración y nuevos requerimientos son agregados o se pueden modificar. (30).

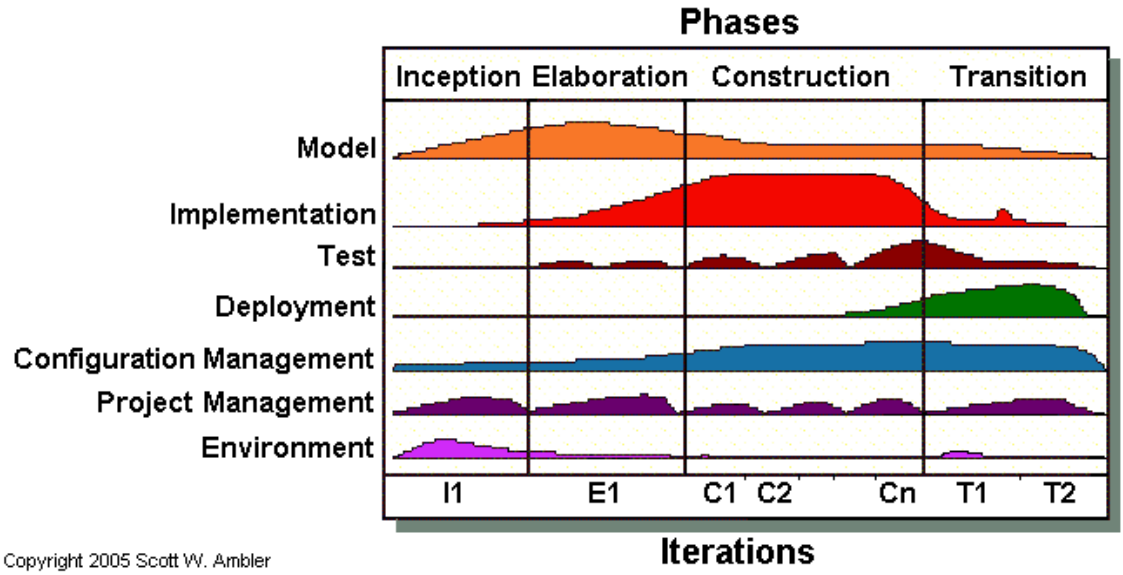


Figura PPUA. 1 Fases y flujos del PUA (28)

Las disciplinas son ejecutadas de una forma iterativa, definiendo las actividades que el equipo de desarrollo ejecuta para construir, validar y liberar software funcional, el cual cumple con las necesidades de los involucrados (28).

Al igual que PUR, AUP tiene las 4 fases clásicas consecutivas:

- **Inicio (Concepción):** El objetivo de esta fase es obtener una comprensión común entre el cliente y el equipo de desarrollo, del alcance del nuevo sistema y definir una o varias arquitecturas candidatas para el mismo.

- **Elaboración:** El objetivo es que el equipo de desarrollo profundice en la comprensión de los requisitos del sistema y en validar la arquitectura.

- **Construcción:** Durante la fase de construcción el sistema es desarrollado y probado por completo en el ambiente de desarrollo.

- **Transición:** el sistema se lleva a los entornos de preproducción donde se somete a pruebas de validación y aceptación y finalmente se despliega en los sistemas de producción.

A continuación se describe las disciplinas de desarrollo consideradas por el PUA:

- **Modelado:** Se busca entender el negocio de la organización, el problema de dominio que se abordan en el proyecto, y determinar una solución viable.

- **Implementación:** Consiste en transformar los modelos o modelo en código ejecutable y realizar un nivel básico de las pruebas.

- **Pruebas:** Se busca realizar una evaluación objetiva para garantizar la calidad. Esto incluye la búsqueda de defectos, validar que el sistema funciona tal como está establecido, y verificando que se cumplan los requisitos.

- **Despliegue:** Consiste en la elaboración de un plan para la entrega del sistema y ejecutar el plan para que el sistema esté a disposición de los usuarios finales.

Disciplinas administrativas que se van llevando paralelamente al desarrollo:

- **Gestión de Configuración:** Consiste en administrar el acceso a los artefactos del proyecto. Esto incluye no sólo el seguimiento de las versiones de los artefactos en el tiempo, sino también el control y la gestión de los cambios.

- **Administración del Proyecto:** Consiste en dirigir las actividades que se llevan a cabo dentro del proyecto. Esto incluye la gestión de riesgos, la dirección de personas (la asignación de tareas, el seguimiento de los progresos, etc.), y de coordinación con el personal y los sistemas fuera del alcance del proyecto para asegurarse de que el software final sea entregado a tiempo y dentro del presupuesto.

- **Entorno:** Es un soporte para el resto de los esfuerzos para garantizar un proceso adecuado, orientación (normas y directrices), y herramientas (hardware, software, etc.) estén disponibles para el equipo según sea necesario

En comparativa con PUR, el PUA condensa los flujos de trabajo expresados en la Tabla PPUA. 3

Proceso Unificado Ágil (PUA)	Proceso Unificado de Rational (PUR)
Modelado	Modelado de negocio Requerimientos Análisis y diseño
Implementación	Implementación
Pruebas	Pruebas
Despliegue	Despliegue
Administración del proyecto	Administración del proyecto
Administración de la configuración	Administración de la configuración y cambios
Ambiente	Ambiente

Tabla PPUA. 3 Comparativa entre los flujos de trabajo del RUP y del PUA

PUA plantea una liberación del producto en partes, por ejemplo, versión 1, luego versión 2 del software en producción y así sucesivamente hasta tener el producto completo. De acuerdo a este planteamiento PUA distingue dos tipos de iteraciones:

- Versión de desarrollo, cuyo resultado está desplegado en entorno QA (Aseguramiento de la calidad) o Demo. Estas versiones deben ser rápidas de ser posible desarrolladas en una o tres semanas. (Rombo azul)

- Versión de producción, cuyo resultado es un despliegue en producción (rombo verde).

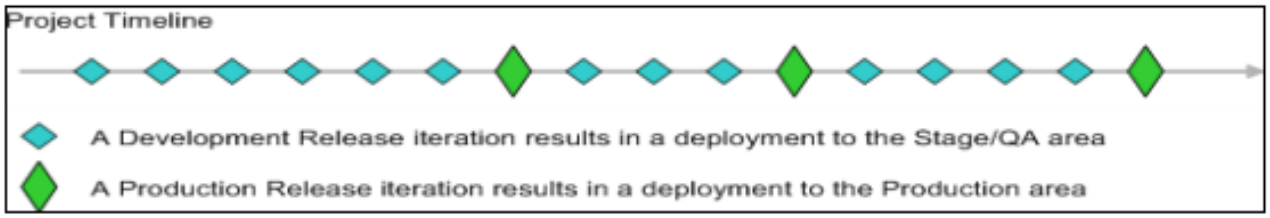


Figura PPUA. 2 Iteraciones PUA

2.2.3 Incorporación del requerimiento de seguridad en el PUA

El presente documento de tesis aborda principalmente las tres disciplinas del PUA que se encargan del desarrollo: modelado, implementación y pruebas.

En cada una de ellas se incorporan actividades relacionadas para el requerimiento de seguridad, comenzando con la fase de inicio y terminando con la de construcción. La fase de transición no se aborda ya que las actividades a seguir son las mismas que en el PUA.

El objetivo es indicar en cada una de las disciplinas nuevas las actividades, productos y roles que el equipo de desarrollo debe realizar para incorporar desde el inicio el requerimiento no funcional de la seguridad, por ser un aspecto particularmente importante, del cual no todos los clientes y desarrolladores conocen, ni reconocen como crítico en vista de los activos que pueden ponerse en riesgo, más allá que la pura funcionalidad del producto.

Es importante indicar que todos los participantes en el proyecto deben de ser íntegros y con buenos antecedentes, además de estar conscientes de los conceptos de seguridad, los riesgos, y el valor de que el equipo de programación conozca las buenas prácticas de desarrollo de software seguro, las madure y actualice. Así como adoctrinar tanto a los miembros de equipo como al cliente de la importancia de la seguridad y lo mucho que se arriesga si no se considera.

Es necesario recalcar la necesidad de que los desarrolladores mantengan una ética que los contenga de incorporar código malisioso y la importancia de hacer una revisión entre colegas para detectar oportunamente estos códigos en caso de existir.

En esta propuesta se incorpora un rol, del cual se describirá su perfil más adelante, llamado: "*Asesor de seguridad*". Quien apoya al equipo de proyecto durante cada uno de los flujos de trabajo y disciplinas, también se incorporan nuevos productos y se enriquecen algunos otros (cada uno descrito en los siguientes capítulos). Sin embargo es importante aclarar, que los agregados al PUA no buscan incrementar de manera considerable el volumen de documentación ni crecer desproporcionadamente el proyecto, sino al contrario, se busca centrar esfuerzos con un mínimo de recursos con lo que se respetan los principios del manifiesto ágil sobre los cuales fue construido el PUA.

2.3 Métricas de seguridad y la calidad del software

El poder medir un proceso lo hace más visible y por lo mismo más entendible y controlable. Se necesita medir las características y los parámetros de calidad de los

desarrollos, ya que para mejorar la calidad de los productos de software, se deben seguir procesos, y dentro de los procesos necesitamos elegir y hacer uso de métodos específicos, empleando herramientas y tecnologías adecuadas.

Una **métrica** es el *planteamiento o descripción de lo que se quiere medir*, tiene asociado un proceso de medición y como consecuencia una medida. El objetivo de las métricas dentro del software es documentar las características de proyectos buenos y malos para analizar tendencias y tomar acciones preventivas y correctivas, permitiendo entender, controlar y mejorar los procesos, el producto y por consecuencia a la organización.

Para poder mejorar la variable de la seguridad de algún método, primero se debe medir a fin de cuantificar su evolución.

En principio las mismas métricas empleadas al cuantificar la calidad del software pueden ser utilizadas para medir la calidad en relación a la seguridad.

El número de vulnerabilidades encontrados entre el tamaño del producto puede proporcionar una línea base a la que llamaremos **Densidad de Defectos**, desde la cual se parte para obtener mayor información sobre la evolución de fallos. Como base se pueden adoptar las siguientes medidas:

Densidad de defectos (DD)

DD(severidad) = # Vulnerabilidades Potenciales / KLOC.
DD(severidad) = # Vulnerabilidades Potenciales / FP.

Las métricas sobre vulnerabilidades deben desglosarse por severidad del defecto, dar un peso mayor a fallos que puedan comprometer una cantidad de activos superior o impactarlos de manera más agresiva. Esta severidad puede basarse en la medición del riesgo propuesta en los temas siguientes, de esta manera podemos definir una densidad de defectos críticos, graves, moderados, de bajo impacto o notas. Ejemplo:

DD(críticos)= # Vulnerabilidades Potenciales de nivel crítico / KLOC.
DD(críticos) = # Vulnerabilidades Potenciales de nivel crítico / FP.

En cada ciclo se contabiliza la densidad de defectos separándolos en base a una clasificación predefinida e ir registrando el tiempo promedio de solución. El conocimiento de esta información permitirá hacer una mejor estimación del posible número de errores y tiempo para corregirlos en proyectos futuros, adaptando la planeación de las siguientes iteraciones.

Promedio de tiempo de solución (PTS)

PTS(severidad)=$\frac{\sum Tiempos\ de\ solución(severidad)}{\#\ de\ vulnerabilidades\ críticas}$

Ejemplo:

PTS(críticos)=$\frac{\sum Tiempos\ de\ solución\ de\ vulnerades\ críticas}{\#\ de\ vulnerabilidades\ críticas}$

El registro histórico de estas métricas puede apoyar al conocimiento del equipo de desarrollo, para comprender su evolución respecto a la seguridad e identificar en qué etapa se detecta o se provoca la mayor densidad de fallas y subsecuentemente tomar decisiones sobre mejorar las actividades relacionadas para reducirlas y estimar el tiempo de solución.

2.4 Casos de uso 2.0

Los casos de Uso 2.0 (31) son una técnica para capturar requerimientos. El punto central de esta técnica es el caso de uso enriquecido con las historias de usuario, descomponiendo a los casos de uso en rebanadas incorporando requerimientos no funcionales, flujos alternativos o la especialización para algún actor en particular.

Ivar Jacobson, junto a Ian Spence y a Kurt Bittner, recientemente publicaron esta técnica en su libro electrónico: "Use Case 2.0" (31).

Las rebanadas evolucionan en rebanadas de análisis, diseño, implementación y pruebas. En palabras de los autores, “una porción de caso de uso no necesariamente contiene un flujo completo” y mucho menos un caso de uso completo, es más bien una parte de funcionalidad con valor para el usuario y que puede servir para la versión o incremento actual del software.

Por las características que brinda este enfoque de modelado, se empleará en el capítulo de Modelado en esta tesis, ejemplificando principalmente el aspecto de la seguridad, describiendo detalladamente el uso de esta técnica.

Capítulo 3 Modificación del Proceso Unificado Ágil con aspectos de seguridad (Flujo de administración del proyecto)

- 3.1 Administración del proyecto

3 MODIFICACIÓN DEL PROCESO UNIFICADO ÁGIL CON ASPECTOS DE SEGURIDAD (FLUJO DE ADMINISTRACIÓN DEL PROYECTO)

Durante la etapa de inicio, se establecen las reglas del negocio relevantes para el sistema y se delimita el alcance del proyecto. Para lograrlo se necesita identificar todas las entidades con las cuales el sistema interactuará (actores) y define la naturaleza de su interacción en un alto nivel. Esto incluye identificar todos los casos de uso y describir un poco de cada uno de ellos. Las reglas del negocio incluyen un criterio de éxito, análisis de riesgo, y las estimaciones necesarias, así como un plan de fases que muestra las fechas en que se alcanzarán los principales objetivos (32).

A continuación se describen los objetivos, productos, roles y actividades que se proponen incluir para fortalecer el requerimiento de seguridad dentro del flujo de trabajo de la Administración del proyecto. Se detallan únicamente los productos, roles y actividades nuevas a incorporar.

3.1 Administración del proyecto

Al flujo de la *administración del proyecto,* así como los flujos subsecuentes, se les agregan nuevos objetivos, roles, productos y actividades a las propuestas por el PUA, enfocadas a la seguridad.

Como primera aparición en el proceso, durante la administración del proyecto, se define el rol de *asesor de seguridad*, el cual estará presente en los demás flujos y actividades propuestos. Por otra parte, se definen nuevos objetivos, dos productos de seguridad y las actividades que los generan, de manera particular a este flujo de trabajo.

3.1.1 Objetivos

- Mantener el proyecto de desarrollo de software planeado, monitoreado y bajo control.
- Administrar los riesgos.
- Establecer las necesidades de seguridad
- Definir el plan de iteraciones incluyendo las actividades de desarrollo de software seguro planteadas en esta tesis.

3.1.2 Productos

- Plan de desarrollo de software.
- Plan de iteraciones.
- Lista de riesgos.
- Mediciones del plan.

- Informe del análisis de riesgos de seguridad (producto de seguridad)
- Contrato de software seguro(producto de seguridad)

3.1.3 Actividades

- Crear el plan del proyecto.
- Crear el plan de iteraciones.
- Identificar y evaluar los riesgos.
- Monitorear el plan del proyecto.
- Analizar los riesgos de seguridad (actividad de seguridad)
- Definir y acordar un contrato de software seguro (actividad de seguridad)

3.1.4 Roles

El rol definido en el PUA es el *administrador de proyecto* es principal participante en la fase de inicio. En esta tesis se propone además el rol de *asesor de seguridad*, encargado de concientizar y dirigir tanto al equipo del proyecto como al cliente, sobre temas de seguridad. A continuación se describe su perfil y función en general.

3.1.4.1 Role: asesor de seguridad

El *asesor de seguridad* es el profesional que se encarga de apoyar y asesorar sobre los temas de seguridad, brindando un voto importante sobre la toma de decisiones referentes a su campo, debe de contar con una profunda experiencia en su campo, así como valores de ética, discreción y responsabilidad, considerando que tiene acceso a información privilegiada. En relación al campo técnico, el asesor de seguridad debe ser un experto en el campo de la seguridad y tecnologías de la información, así como en prácticas de desarrollo de software seguro. También debe tener una actitud conciliadora y concientizadora, para hacer frente a desacuerdos que pueden resultar de algunas prácticas y medidas de seguridad.

Está involucrado en todas las actividades del proyecto que tengan que ver con su campo, acompañando a todos los demás roles en sus actividades. Puede existir más de un asesor de seguridad según el tamaño del proyecto y los recursos con los que se cuenten.

Su misión es evaluar y proporcionar su punto de vista referente al requerimiento de seguridad, ofreciendo la mejor solución al proyecto.

3.1.5 Actividad: definir y acordar un contrato de software seguro

En esta actividad se acuerda junto con el cliente, el alcance del desarrollo en cuestión de la seguridad, se definen la responsabilidad del equipo de proyecto sobre los fallos que se pueden detectar y su corrección, considerando el impacto, costo, tiempo y esfuerzo que se requiere para corregirlos. Durante esta actividad el asesor de seguridad aporta su punto de vista sobre los cuales se basan y dictan las directrices del documento resultante, el cual es el contrato de software seguro.

3.1.5.1 Producto: Contrato de software seguro

La sección a continuación descrita, está basada en el proyecto legislativo de OWASP (22), particularmente en su apartado de contrato de Software Seguro, y para su análisis debe participar el asesor de seguridad junto con el administrador del proyecto y los stakeholders que éste último considere necesarios.

Este contrato pretende ayudar a los proveedores de software y sus clientes a negociar y capturar importantes términos y condiciones contractuales relacionadas a la seguridad en el software a ser desarrollado o entregado. La razón es que la mayoría de los contratos no mencionan estos temas, y las partes frecuentemente tienen puntos de vista dramáticamente diferentes a lo que en realidad fue acordado. Articular claramente estos términos es la mejor manera de asegurarse que ambas partes pueden hacer decisiones informadas sobre cómo proceder.

El contrato busca expresar qué actividades deberían hacerse por ambas partes para minimizar costos, y cuándo deberían ocurrir, y pretende facilitar la discusión sobre quién tomaría el riesgo por las vulnerabilidades de seguridad en el software. En un extremo del espectro de la seguridad, el cliente podría tomar todo el riesgo y el proveedor podría entregar código con muchas vulnerabilidades. Por el otro extremo, el proveedor podría tomar todo el riesgo y asumir la responsabilidad de entregar código perfecto. Ambos extremos son irrealizables. En este convenio se propone que el proveedor tome el riesgo por problemas que sean descubiertos en los requerimientos o deben ser cubiertos por esfuerzos razonables en pruebas. Pero del remedio a "nuevos" problemas de seguridad tendría que ser pagado por el cliente en un equilibrio funcional, ya que limita el riesgo adquirido por el proveedor y promueve obtener requerimientos de seguridad por anticipado. Esto es un ejemplo de conciliación de intereses, pero no es la única distribución que puede hacerse.

También es importante contemplar la responsabilidad por código de terceros, la cual podría ser llevada mejor por el proveedor, aunque ellos podrían no tener toda la capacidad para garantizar la seguridad por ellos mismos. Sin embargo, la seguridad puede ser parte de la decisión de "construir o comprar" y esta parece ser la mejor manera de

promoverla. El proveedor, por supuesto, tiene la opción de transferir la responsabilidad a un proveedor externo o puede también analizar el código de terceros por él mismo o contratar expertos en seguridad para analizarlo por él.

Hay muchos beneficios al trabajar con este convenio. El principal es que pondrá claras las expectativas entre las partes involucradas. En algunos casos ayudará a evitar demandas cuando problemas de seguridad difíciles aparecen en el software. También, estas son las mismas actividades requeridas por muchas razones legales y de conformidad con estándares o leyes.

El objetivo de esta documentación es simplemente asegurar, en cada etapa del desarrollo de software, que se ha puesto la atención apropiada en la seguridad. Un beneficio adicional es que esta documentación puede ser recolectada en conjunto para formar un "paquete de certificación" que básicamente esboce el argumento de porqué deberíamos confiar que el software hará lo que dice hacer.

En el Apéndice 6.1, se incluye un contrato de software seguro propuesto por el proyecto OWASP el cual puede ser empleado como base para el contrato particular de cada proyecto.

3.1.6 Actividad: analizar los riesgos de seguridad

El descubrimiento de vulnerabilidades es importante, pero igual de importante es ser capaz de estimar el riesgo asociado a la empresa. Siguiendo este enfoque, se debe ser capaz de estimar no solo los riesgos sino también la gravedad de los mismos. Con el objetivo de tomar una decisión informada sobre qué hacer con ellos. Un sistema de calificación permite ahorrar tiempo. Este sistema de calificación debe ser particular para cada organización, ya que no puede existir una calificación universal para todas las organizaciones, esto es porque una vulnerabilidad y su impacto puede resultar fundamental para una empresa y no muy importante para otras organizaciones. Es por ello que aquí solo se expresa un marco básico que debe de personalizarse.

Durante esta actividad, el asesor de seguridad expone el riesgo que una vulnerabilidad puede representar para la organización, y el cliente indica el impacto según su negocio y entorno. De manera conjunta se evalúa cuantitativamente el nivel de riesgo y priorización, dando como resultado un informe de análisis de seguridad.

A continuación se describe la metodología sugerida para evaluar los riesgos basada en la propuesta ofrecida por el proyecto OWASP (22) en su apartado de análisis de riegos.

Paso 1. Identificación del riesgo

Recopilar información sobre los agentes de amenaza, los diferentes ataques que se pueden ejecutar, las vulnerabilidades que pueden explotar, y el impacto para el negocio de un ataque exitoso.

Paso 2. Factores para la estimación de riesgo

Después de identificar el riesgo es necesario saber que tan grave es, para ello se calcula la probabilidad de que la vulnerabilidad sea descubierta y explotada por un atacante. Puede bastar con indicar si la probabilidad es alta, media o baja. Sin precisar, según lo afirma la metodología de OWASP.

Para resolver esto, nos pueden ayudar los siguientes factores divididos en conjuntos de opciones, cada uno en una escala del 0 al 9, valor indicado entre paréntesis y ubicado frente a cada opción, con cual podemos calcular la probabilidad total:

Factor: Agente de amenaza

Se refiere a la probabilidad de una amenaza según las condiciones que debe contar para atentar en contra del producto o la organización.

Nivel de habilidad

a) Sin conocimientos técnicos (1)
b) Algunos conocimientos técnicos (3)
c) Usuario avanzado de informática (4)
d) Habilidades de programación y conocimiento de redes (6)
e) Habilidades de penetración de seguridad (9)

Motivo

a) Disminución o ausencia de recompensa (1)
b) Posible recompensa (4)
c) Alta recompensa (9)

Oportunidad

a) Acceso completo o costosos recursos necesarios (0)
b) Acceso especial o recursos necesarios (4)
c) Algunos accesos o recursos necesarios (7)
d) Falta de acceso o recursos necesarios (9)

Tamaño

a) Comprende sólo a los desarrolladores (2)
b) Abarca sólo a los Administradores de sistemas (2)
c) Usuarios de la intranet (4)
d) Socios (5)
e) Usuarios autenticados (6)

f) Usuarios anónimos de internet (9)

Factor: Vulnerabilidad

Este factor indica las características de un fallo presente en el producto y evalúa la probabilidad de explotación del mismo independientemente de la amenaza.

Facilidad de descubrimiento

a) Prácticamente imposible (1)
b) Difícil (3)
c) Fácil (7)
d) Detectable por herramientas automatizadas (9)

Facilidad de explotación

a) Teórico (1)
b) Difícil (3)
c) Fácil (5)
d) Explotable por herramientas automatizadas (9)

Conciencia

a) Desconocida (1)
b) Oculto (4)
c) Obvio (6)
d) Del conocimiento público (9)

Detección de intrusos

a) Detección activa en la aplicación (1)
b) Registrados y revisados (3)
c) Registrados sin la revisión (8)
d) No se registran (9)

Paso 3. Estimación del impacto

Se refiere al daño que un ataque exitoso puede provocar. Existen dos tipos de impacto: el "**impacto técnico"** el cual afecta en la solicitud del servicio, los datos que utiliza y las funciones que ofrece el sistema. El otro es el "**impacto en el negocio**" de la empresa operadora de la aplicación.

Factor de impacto técnico:

Pérdida de confidencialidad

a) Mínimo de datos sensibles revelados (2)
b) Mínimo de datos críticos dados a conocer (6)
c) Extensos datos no confidenciales revelados (6)
d) Gran cantidad de datos críticos (7)
e) Todos los datos divulgados (9)

Pérdida de integridad

a) Mínimo de datos ligeramente corruptos (1)
b) Mínimo de datos seriamente corruptos (3)
c) Gran cantidad de datos poco corruptos (5)
d) Gran cantidad de datos corruptos seriamente en serio (7)
e) Todos los datos totalmente corruptos (9)

Pérdida de disponibilidad

a) Mínimos servicios secundarios interrumpidos (1)
b) Mínimo de servicios primarios interrumpidos (5)
c) Amplios servicios secundarios interrumpidos (5)
d) Amplios servicios primarios interrumpidos (7)
e) Todos los servicios completamente perdido (9)

Perdida rastreo

a) Trazabilidad completa (1)
b) Posiblemente trazable (7)
c) Totalmente anónimo (9)

Factor de impacto al negocio

Daño financiero

a) Menor que el costo para corregir la vulnerabilidad (1)
b) Menor efecto en el beneficio anual (3)
c) Un efecto significativo sobre la ganancia anual (7)
d) La quiebra (9)

Daño a su reputación

a) Daños mínimos (1)
b) Pérdida de grandes cuentas (4)
c) Pérdida de buena voluntad (5)
d) Daños a la marca (9)

Incumplimiento

a) Violación de menor importancia (2)
b) Una clara violación (5)
c) La violación de alto perfil (7)

Violación de privacidad

a) Un individuo (3)
b) Cientos de personas (5)
c) Miles de personas (7)
d) Millones de personas (9)

Paso 4. Determinación de la gravedad del riesgo

En este paso se determina la estimación de la probabilidad y el impacto para el cálculo general de la gravedad del riesgo. Según la suma del puntaje seleccionado, se determina si la probabilidad es alta, media o baja.

Para medir el impacto o la probabilidad, empleamos la siguiente escala.

Los niveles de probabilidad e impacto	
0 a <3	BAJA
3 a <6	MEDIO
6 a 9	ALTA

Con base en la probabilidad e impacto y de acuerdo a la siguiente tabla, podemos estimar la gravedad del riesgo global:

La gravedad del riesgo global				
Impacto	ALTA	Medio	Alto	Crítico
	MEDIO	Bajo	Medio	Alto
	BAJA	Nota	Bajo	Medio
		BAJA	MEDIO	ALTA
	Probabilidad			

Paso 5. Decidir la prioridad

Después de clasificar los riesgos, es necesario priorizar. Como regla general se deben fijar los riesgos más graves incluso si los riesgos menos importantes son más fáciles o baratos arreglar.

No todos los riesgos deberán neutralizarse, esto dependerá de cada organización y disposición para aceptar el riesgo. Por ejemplo: si le cuesta $ 100.000 implementar controles para frenar un fraude de $ 2.000 por año, se necesitarían 50 años de retorno de la inversión, pero es necesario considerar que puede haber un daño en la reputación que podría costar a la organización mucho más.

Paso 6. Personalización del modelo de calificación de riesgos

Un modelo personalizado es capaz de ofrecer mayores beneficios, coincidiendo de mejor manera con las percepciones de la gente sobre lo que es un riesgo grave. Sin embargo este modelo puede apoyar para hacer una aproximación o como punto de partida para agregar o quitar factores, opciones o modificar la ponderación para ajustar el modelo según las necesidades. Por ejemplo: una aplicación militar podría añadir factores de impacto relacionados con la pérdida de vidas humanas o información clasificada. También

podría añadir factores de probabilidad, como la ventana de oportunidad de un atacante o la fuerza de un algoritmo de cifrado.

3.1.7 Producto: informe del análisis de riesgos de seguridad

Este producto tiene como objetivo la identificación y priorización de las vulnerabilidades posibles del producto, se debe considerar su entorno de implementación, tipo de usuarios, información y activos relacionados al mismo ya sea directa o indirectamente.

Para cada una de los riesgos, OWASP (22) ofrece información genérica acerca de la probabilidad y el impacto técnico a través del siguiente esquema.

Amenaza agente	Vector de ataque	Debilidad Prevalencia	Debilidad Detectabilidad	Impacto técnico	Impacto en el Negocio
?	Fácil	Extendido	Fácil	Grave	?
?	Promedio	Común	Promedio	Moderado	?
?	Difícil	Poco común	Difícil	Menor	?

3.1.7.1 Riesgo relacionado a las 10 vulnerabilidades más frecuentes en la web

A continuación se describe la calificación de riesgo dada a las 10 vulnerabilidades más frecuentes de los sistemas web, en base a la metodología de análisis de riesgo que propone OWASP (22). Como ejemplo de un informe de análisis de riesgos de seguridad:

3.1.7.1.1 Inyección de código

Las fallas de inyección, tales como SQL, OS, y LDAP, ocurren cuando datos no confiables son enviados a un intérprete como parte de un comando o consulta. Los datos hostiles del atacante pueden engañar al intérprete en ejecutar comandos no intencionados o acceder datos no autorizados.

Agente de riesgo	Vector de ataque	Deficiencias de seguridad		Impacto técnico	Impacto de negocio
———	**Explotación Fácil**	**Prevalencia Común**	**Detección Promedio**	**Impacto Grave**	———
Se considere la posibilidad de cualquier persona que puede enviar	El atacante envía sencillos ataques basados en texto que se aprovechan	Las fallas de inyección se producen cuando una aplicación envía datos no confiables a un intérprete. Las fallas de inyección son muy		La inyección puede resultar en la pérdida de datos o la corrupción, o la	Se considera el valor comercial de los datos afectados y la plataforma

datos al sistema, incluidos los usuarios externos, usuarios internos, y los administradores.	de la sintaxis del intérprete. Casi cualquier fuente de datos puede ser un vector de inyección, incluyendo fuentes internas	frecuentes, sobre todo en el código heredado, a menudo se encuentran en las consultas SQL, consultas LDAP, consultas XPath, comandos del sistema operativo, los argumentos del programa, etc. Son fáciles de descubrir al examinar el código, pero es más difícil a través de las pruebas. Escáneres y fuzzers pueden ayudar a los atacantes para detectarlos.	denegación. La Inyección a veces puede llevar a comprometer completamente el servidor que resguarda la aplicación.	donde se ejecuta el intérprete. Todos los datos pueden ser robados, modificados o eliminados.

3.1.7.1.2 Cross Site Scripting XSS

Las fallas XSS ocurren cada vez que una aplicación toma datos no confiables y los envía al navegador web sin una validación y codificación apropiada. XSS permite a los atacantes ejecutar secuencia de comandos en el navegador de la víctima los cuales pueden secuestrar las sesiones de usuario, destruir sitios web, o dirigir al usuario hacia un sitio malicioso.

Agente de riesgo	Vector de ataque	Deficiencias de seguridad		Impacto técnico	Impacto de negocio
———	**Explotación Media**	**Prevalencia Muy difundida**	**Detección Fácil**	**Impacto Moderado**	———
Considerar cualquier persona que pueda enviar datos no confiables al sistema, incluyendo usuarios externos, internos y administradores.	El atacante envía simples cadenas de texto que explotan la sintaxis del intérprete atacado. Casi cualquier fuente de datos puede ser un vector de inyección, incluyendo fuentes internas tales como datos de la base de datos.	XSS es la falla de seguridad más prevalente en aplicaciones web. Las fallas XSS ocurren cuando una aplicación incluye datos suministrados por el usuario en una página enviada al navegador sin ser el contenido apropiadamente validado o escapado. Existen tres tipos conocidos de fallas XSS: 1) Almacenados 2) Reflejados 3) XSS basado en DOM La detección de la mayoría de las fallas XSS es relativamente fácil a través de pruebas análisis de código.		Los atacantes pueden ejecutar secuencias de comandos en el navegador de una víctima para secuestrar las sesiones de usuario, destruir sitios web, insertar código hostil, redirigir usuarios, instalar código malicioso en el navegador de la víctima, etc.	Considerar el valor en el negocio de los datos afectados o funciones de la aplicación o el impacto en el negocio de la exposición pública de la vulnerabilidad.

3.1.7.1.3 Pérdida de Autenticación y Gestión de Sesiones

Las funciones de la aplicación relacionadas a autenticación y gestión de sesiones son frecuentemente implementadas incorrectamente, permitiendo a los atacantes comprometer contraseñas, llaves, token de sesiones, o explotar otras fallas de implementación para asumir la identidad de otros usuarios.

Agente de riesgo	Vector de ataque	Deficiencias de seguridad		Impacto técnico	Impacto de negocio
———	**Explotación Media**	**Prevalencia Común**	**Detección Media**	**Impacto Severo**	———
Atacantes anónimos externos, además de usuarios con sus propias cuentas, que podrían intentar robar cuentas de otros, también a trabajadores que quieran enmascarar sus acciones.	El atacante utiliza filtraciones o vulnerabilidades en las funciones de autenticación o gestión de las sesiones (por ejemplo cuentas expuestas, contraseñas, identificadores de sesión) para hacerse pasar por otros usuarios.	Los desarrolladores a menudo crean esquemas propios de autenticación o gestión de las sesiones, pero conseguir que sean correctos es complicado. Por ello, a menudo estos esquemas propios contienen vulnerabilidades en las secciones de cierre de sesión, gestión de contraseñas, tiempo de desconexión, función de recordar contraseña, pregunta secreta, actualización de cuenta, etc. Encontrar estas vulnerabilidades puede ser difícil por la particularidad de la implementación.		Estas vulnerabilidades podrían permitir que algunas o todas las cuentas sean atacadas. Una vez el ataque resulte satisfactorio, el atacante podría realizar cualquier acción que la víctima pudiese. Las cuentas privilegiadas son los objetivos prioritarios.	Considerar el valor de negocio de los datos afectados o funciones de la aplicación. Considerar el impacto en el negocio de la exposición pública de la vulnerabilidad.

3.1.7.1.4 Referencia Directa Insegura a Objetos

Una referencia directa a objetos ocurre cuando un desarrollador expone una referencia a un objeto de implementación interno, tal como un archivo, directorio, o base de datos. Sin un chequeo de control de acceso u otra protección, los atacantes pueden manipular estas referencias para acceder datos no autorizados.

Agente de riesgo	Vector de ataque	Deficiencias de seguridad		Impacto técnico	Impacto de negocio
———	**Explotación Fácil**	**Prevalencia Común**	**Detección Fácil**	**Impacto Severo**	———
Considerar los tipos de usuarios en su sistema. ¿Existen usuarios que tengan únicamente acceso parcial a determinados tipos de datos del sistema?	Un atacante, como usuario autorizado en el sistema, simplemente modifica el valor de un parámetro que se refiere directamente a un objeto del sistema a otro objeto para el que el usuario no se encuentra autorizado. ¿Se concede el acceso?	Normalmente, las aplicaciones utilizan el nombre o clave actual de un objeto cuando se generan las páginas web. Las aplicaciones no siempre verifican que el usuario tiene autorización sobre el objetivo. Esto resulta en una vulnerabilidad de referencia de objetos directos inseguros. Los auditores pueden manipular fácilmente los valores del parámetro para detectar estas vulnerabilidades y un análisis de código mostraría rápidamente si la autorización se verifica correctamente.		Dichas vulnerabilidades pueden comprometer toda la información que pueda ser referida por parámetros. A menos que el espacio de nombres resulte escaso, para un atacante resulta sencillo acceder a todos los datos disponibles de ese tipo.	Considerar el valor de negocio de los datos afectados. También considere el impacto en el negocio de la exposición pública de la vulnerabilidad

3.1.7.1.5 Falsificación de Peticiones en Sitios Cruzados (CSRF)

Un ataque CSRF obliga al navegador de una víctima autenticada a enviar una petición HTTP falsificada, incluyendo la sesión del usuario y cualquier otra información de autenticación incluida automáticamente, a una aplicación web vulnerable. Esto permite al atacante forzar al navegador de la víctima para generar peticiones que la aplicación vulnerable considera legítimas provenientes de la víctima.

Agente de riesgo	Vector de ataque	Deficiencias de seguridad		Impacto técnico	Impacto de negocio
———	**Explotación Media**	**Prevalencia Muy común**	**Detección Fácil**	**Impacto Moderado**	———
Cualquiera que pueda suplantar a usuarios al momento de enviar peticiones a un sitio web. Cualquier sitio web,	Los atacantes crean peticiones HTTP falsas. Engañan a la víctima al enviarlas a través de etiquetas de imágenes, XSS, o muchas otras técnicas. Si el usuario	La CSRF aprovecha aplicaciones web que permiten a los atacantes predecir todos los detalles de un acción en particular. Cuando los navegadores envían credenciales de autenticación automáticamente, como en el caso de las		Los atacantes pueden cambiar cualquier dato que la víctima esté autorizado a cambiar, o acceder a	Considerar el valor de negocio asociado a los datos o funciones afectados. Tener en cuenta lo que representa no estar seguro si los

u otros canales HTML, a los cuales accedan los usuarios de un determinado sitio web.	está autenticado entonces el ataque será exitoso.	cookies de sesión, los atacantes pueden crear páginas web maliciosas las cuales generan peticiones falsas que son indistinguibles de las auténticas. Los fallos debidos a CSRF son fácilmente detectables a través de código, o pruebas de penetración.	cualquier funcionalidad que la víctima esté autorizada a utilizar.	usuarios en realidad desean realizar dichas acciones. Considerar el impacto que tiene en la reputación del negocio..

3.1.7.1.6 Defectuosa configuración de seguridad

Una buena seguridad requiere tener definida e implementada una configuración segura para la aplicación, marcos de trabajo, servidor de aplicación, servidor web, base de datos, y plataforma. Todas estas configuraciones deben ser definidas, implementadas, y mantenidas ya que por lo general no son seguras por defecto. Esto incluye mantener todo el software actualizado, incluidas las librerías de código utilizadas por la aplicación.

Agente de riesgo	Vector de ataque	Deficiencias de seguridad		Impacto técnico	Impacto de negocio
———	**Explotación Fácil**	**Prevalencia Común**	**Detección Fácil**	**Impacto Moderado**	———
Atacantes externos anónimos, así como usuarios con contraseñas auténticas que puedan ser utilizadas para comprometer el sistema. También se incluye a empleados con información y acceso privilegiado que quieran ocultar sus acciones.	Para obtener acceso, o conocimiento, no autorizado al sistema, el atacante puede utilizar cuentas predeterminadas, páginas no utilizadas, defectos en software no actualizado o no parchados, archivos o directorios no protegidos, etc.	Una mala configuración de seguridad puede ocurrir en cualquier capa de la aplicación, incluyendo la plataforma, el servidor web, el servidor de aplicaciones, el ambiente de trabajo, y el código personalizado. Los desarrolladores de software y los administradores de la red necesitan trabajar de forma conjunta para asegurar que todos los niveles de la pila de la aplicación estén correctamente configurados. Los escaneos automatizados son útiles para detectar actualizaciones pendientes, configuraciones defectuosas, cuentas predeterminadas activas, servicios activos innecesarios, etc.		Los defectos frecuentemente permiten a los atacantes obtener acceso no autorizado a datos o funcionalidad del sistema. De forma ocasional, tales defectos resultan en un riesgo para todo el sistema.	El sistema puede estar en riesgo sin que se pueda tener conocimiento de este hecho. Los datos pueden ser robados o modificados. Los costos de recuperación pueden ser altos.

3.1.7.1.7 Almacenamiento criptográfico inseguro

Muchas aplicaciones web no protegen adecuadamente los datos sensibles, tales como tarjetas de crédito, datos personales y credenciales de autenticación con mecanismos de cifrado o hashing. Atacantes pueden modificar o robar tales datos protegidos

inadecuadamente para conducir robos de identidad, fraudes de tarjeta de crédito u otros crímenes.

Agente de riesgo	Vector de ataque	Deficiencias de seguridad		Impacto técnico	Impacto de negocio
———	**Explotación Difícil**	**Prevalencia Poco común**	**Detección Difícil**	**Impacto Severo**	———
Considere a los usuarios de su sistema. ¿Estarían interesados en obtener acceso a datos protegidos para los cuales no tienen autorización? ¿Ha considerado a sus administradores de sistemas internos?	Los atacantes normalmente no rompen el sistema criptográfico. Rompen alguna otra cosa, por ejemplo, encontrando claves, copias de datos no cifradas o accediendo por canales que automáticamente descifran la información.	El error más común en este área es simplemente no cifrar datos que deberían ser cifrados. Cuando se cifra la información, son comunes la generación y almacenamiento inseguros de claves, no rotación de claves y el uso de algoritmos débiles. También es común el uso de hashes inseguros y sin sal para la protección de contraseñas. Los atacantes externos tendrán dificultades para identificar este tipo de vulnerabilidades debido al acceso limitado que disponen. Normalmente es necesario explotar alguna otra vulnerabilidad primero con el objetivo de obtener el nivel de acceso necesario.		Esta vulnerabilidad normalmente compromete todos los datos que deberían haber estado cifrados. Típicamente esta información incluye datos sensibles tales como datos médicos, cuentas de usuario, datos personales, tarjetas de crédito, etc.	Considere el valor para su negocio de los datos perdidos y el impacto a su reputación. ¿Cuál es su responsabilidad legal si esos datos son expuestos? Además considere los daños a su reputación.

3.1.7.1.8 Falla de Restricción de Acceso a URL

Muchas aplicaciones web verifican los privilegios de acceso a URLs antes de generar enlaces o botones protegidos. Sin embargo, las aplicaciones necesitan realizar controles similares cada vez que estas páginas son accedidas, o los atacantes podrán falsificar URLs para acceder a estas páginas.

Agente de riesgo	Vector de ataque	Deficiencias de seguridad		Impacto técnico	Impacto de negocio
———	**Explotación Fácil**	**Prevalencia Poco común**	**Detección Media**	**Impacto Moderado**	———
Cualquiera con acceso a la red puede enviar una petición a su aplicación. ¿Podría un usuario anónimo acceder a una página privada o un usuario normal acceder a una página que requiera privilegios?	El atacante, que es un usuario legítimo en el sistema, simplemente cambia la URL a una página con privilegios. ¿Se le concede acceso? Usuarios anónimos podrían acceder páginas privadas que no están protegidas.	Las aplicaciones no siempre protegen las páginas adecuadamente. En ocasiones la protección a URLs se administra por medio de una configuración, y en sistema está mal configurado. Otras veces los programadores deben incluir el código adecuado que verifique el acceso y se olvidan. La detección de este tipo de fallo es sencilla. La parte más compleja es identificar qué páginas (URLs) existen para el ataque.		Estas vulnerabilidades permiten a los atacantes el acceso no autorizado a funciones del sistema. Las funciones administrativas con un objetivo clave de este tipo de ataques.	Considere el valor para su negocio de las funciones que quedan expuestas y los datos que éstas procesan. Además, considere el impacto a su reputación si esta vulnerabilidad se hiciera pública.

3.1.7.1.9 Protección Insuficiente en la capa de Transporte

Las aplicaciones frecuentemente fallan al autenticar, cifrar, y proteger la confidencialidad e integridad de tráfico de red sensible. Cuando esto ocurre, es debido a la utilización de algoritmos débiles, certificados expirados, inválidos, o sencillamente no utilizados correctamente.

Agente de riesgo	Vector de ataque	Deficiencias de seguridad		Impacto técnico	Impacto de negocio
———	**Explotación Difícil**	**Prevalencia Común**	**Detección Fácil**	**Impacto Moderado**	———
Considere la probabilidad de que alguien pueda capturar el tráfico de red de sus usuarios. Si la aplicación se encuentra en Internet, debe considerar quién conoce cómo sus usuarios pueden acceder a esta aplicación. Por otro lado, no olvide las conexiones a sistemas finales (back-end).	Aunque generalmente es difícil capturar el tráfico de red de los usuarios, en ocasiones puede resultar fácil. La principal dificultad radica en capturar el tráfico de red adecuado mientras los usuarios están accediendo al sitio vulnerable.	Con frecuencia, las aplicaciones no protegen el tráfico de red. Si utilizan SSL/TLS durante la autenticación, pero no en otros lugares, posibilitan que datos sensibles e identificadores de sesión puedan ser interceptados. A menudo, también se utilizan certificados expirados o configurados incorrectamente. Detectar problemas básicos es fácil, basta con observar el tráfico de red. Otras deficiencias más sutiles requieren analizar el diseño de la aplicación y la configuración del servidor.		Estos problemas exponen información asociada a los usuarios y pueden derivar en un robo de cuentas. Si una cuenta de administración es comprometida, podría verse expuesta toda la aplicación. Configuraciones SSL deficientes también pueden facilitar los ataques de phishing y MITM.	Considere el valor de negocio de la información expuesta en los canales de comunicación, en cuanto a sus necesidades de confidencialidad e integridad, así como la necesidad de autenticar a ambos extremos.

3.1.7.1.10 Redirecciones y reenvíos no validados

Las aplicaciones web frecuentemente redirigen y reenvían a los usuarios hacia otras páginas o sitios web, y utilizan datos no confiables para determinar la página de destino. Sin una validación apropiada, los atacantes pueden redirigir a las víctimas hacia sitios de phishing o malware o utilizar reenvíos para acceder páginas no autorizadas.

Agente de riesgo	Vector de ataque	Deficiencias de seguridad		Impacto técnico	Impacto de negocio
———	**Explotación Media**	**Prevalencia Poco común**	**Detección Fácil**	**Impacto Moderado**	———
Considere la probabilidad de que alguien pueda engañar a los usuarios a enviar una petición a su aplicación web. Cualquier aplicación o código HTML al que acceden sus usuarios podría realizar este engaño.	Un atacante crea enlaces a redirecciones no validadas y engaña a las víctimas para que hagan clic en dichos enlaces. Las víctimas son más propensas a hacer clic sobre ellos ya que el enlace lleva a una aplicación en la que se confía. El atacante tiene como objetivo los destinos inseguros para evadir los controles de seguridad.	Con frecuencia, las aplicaciones redirigen a los usuarios a otras páginas, o utilizan destinos internos de forma similar. Algunas veces la página de destino se especifica en un parámetro no validado, permitiendo a los atacantes elegir dicha página.	Detectar redirecciones sin validar es fácil. Se trata de buscar redirecciones donde el usuario puede establecer la dirección URL completa. Verificar reenvíos sin validar resulta más complicado ya que apuntan a páginas privadas.	Estas redirecciones pueden intentar instalar código malicioso o engañar a las víctimas para que revelen contraseñas u otra información sensible. El uso de reenvíos inseguros puede permitir evadir el control de acceso.	Considere el valor de negocio de conservar la confianza de sus usuarios. ¿Qué pasaría si sus usuarios son infectados con código malicioso? ¿Qué ocurriría si los atacantes pudieran acceder a operativas que sólo debieran estar disponibles de forma interna?

Capítulo 4 Modificación del Proceso Unificado Ágil con aspectos de seguridad (Modelado y arquitectura)

- 4.1 Modelado
- 4.2 Establecer una arquitectura de seguridad posible.

4 MODIFICACIÓN DEL PROCESO UNIFICADO ÁGIL CON ASPECTOS DE SEGURIDAD (MODELADO Y ARQUITECTURA)

4.1 Modelado

En este flujo se obtiene un marco de referencia sobre lo que se va a generar como producto final y se definen todos los puntos que se deban considerar, asegurándose de que se ha entendido la necesidad del cliente. Se establecen y analizan los requerimientos y en base a ellos, se esbozan las arquitecturas candidatas.

En este flujo se explica la técnica de *casos de uso 2.0,* en la cual se aprovecha la definición de rebanada para incorporar las diferentes vulnerabilidades modeladas como rebanadas. Para cada caso de uso, se identifican sus vulnerabilidades y se define de manera concreta los puntos en donde puede existir un fallo y en los cuales hay que poner atención, de igual forma, se definen los casos de prueba tanto de la funcionalidad como de la seguridad.

4.1.1 Objetivo

- Esbozar los casos de uso y los requerimientos clave que dirigirán las decisiones.
- Esbozar las arquitecturas candidatas.

4.1.2 Actividades

- Análisis de los requerimientos
- Establecer una arquitectura posible.

4.1.3 Análisis de requerimientos incluyendo la seguridad

Durante el proceso del análisis de requerimientos es necesario profundizar en el requerimiento no funcional de la seguridad, en este momento se aterrizan las vulnerabilidades analizadas en los puntos anteriores dentro de cada uno de los casos de uso.

El asesor de seguridad es el encargado de diseñar junto con los analistas de requerimientos para diseñar cada caso de uso, considerando la seguridad, la funcionalidad y los activos valiosos que están involucrados.

4.1.4 Casos de Uso 2.0 para la seguridad

El modelado de requerimientos contemplando la seguridad, se enfoca en describir la funcionalidad y exponer las consideraciones de seguridad.

A continuación se describen algunos de los casos de uso comúnmente empleados en los sistemas, con la descripción de las vulnerabilidades que pueden afectarlos según el entorno que se describe, con el objetivo de ejemplificar el empleo de Casos de Uso 2.0 incorporando seguridad. Esto nos permite modelar problemáticas ajenas a la funcionalidad específica, a través de rebanadas ortogonales que definen requisitos y condiciones especiales o en este caso potenciales vulnerabilidades.

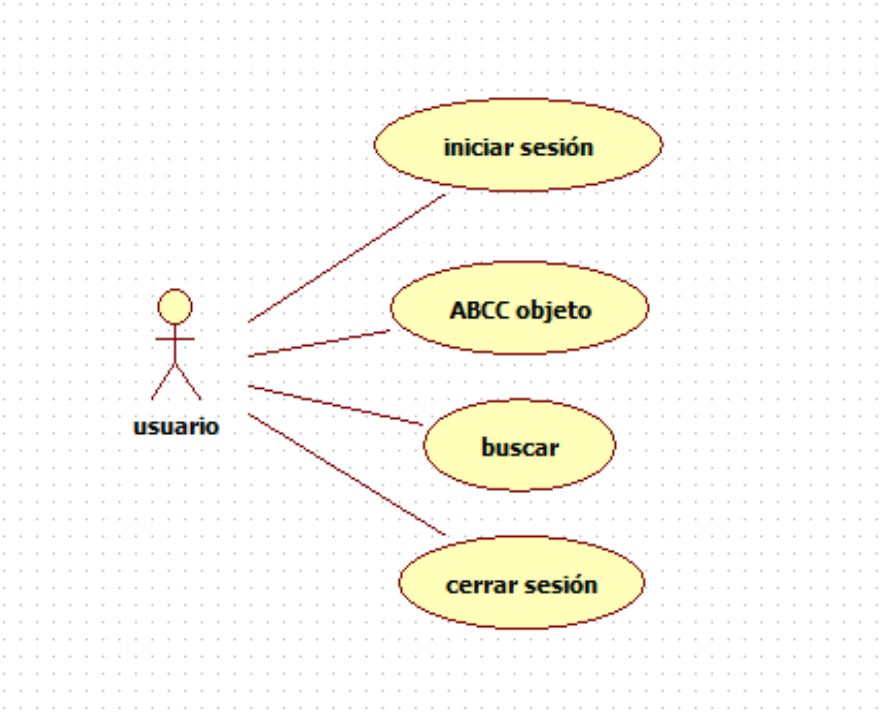
Las actividades necesarias para el modelado de requerimientos a través de los *casos de uso 2.0* son:

1) Encontrar los actores y casos de uso
2) Crear el modelo de casos de uso
3) Rebanar los casos de uso en:
    a. Flujo básico
    b. Flujos alternativos
    c. Aseguramiento de algún requerimiento no funcional
4) Preparar una rebanada
    a. Definir sus casos de prueba
    b. Analizar la rebanada

5) Regresar a revisar el modelo de casos de uso, adaptar los cambios y volver a empezar.

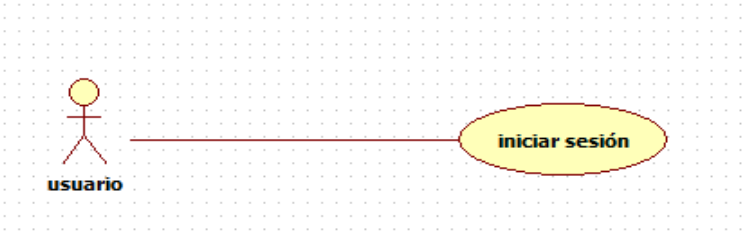
Es importante aclarar que estas vulnerabilidades y la descripción de los casos de uso, tienen el objetivo de ejemplificar y pueden cambiar según las condiciones y entorno particular de cada proyecto. Es de esta forma que las vulnerabilidades detectables en cada funcionalidad de un sistema real pueden ser diferentes, además de que sólo se plantean bajo el entorno de las 10 vulnerabilidades más comunes en los desarrollos web y por lo tanto debe contemplarse el universo total de los tipos de vulnerabilidades y filtrar las que apliquen en cada caso particular.

4.1.4.1 Diagrama General de Casos de Uso genérico



4.1.4.2 Casos de Uso desarrollados

4.1.4.2.1 **Caso de uso 1:** Iniciar sesión.



Descripción: El usuario a través de la tupla (nombre de usuario, contraseña) se identifica en el sistema, el cual lo autentica a través de una base de datos y le permite el acceso al sistema, iniciando una sesión en el sistema.

Nivel: el nivel de este ciclo es alto ya que es una precondición para hacer uso de los módulos funcionales del sistema.

Actores principales: usuario.

Inicio del caso de uso: cuando el usuario requiere ingresar al sistema.

Precondiciones: el usuario debe de estar registrado en el sistema.

Post-condiciones: el sistema efectúa cada acción en nombre del usuario que inicio sesión.

Excepciones no controladas: No se puede iniciar sesión debido a una falla en la conexión con la base de datos.

Consecuencias de fallo: de no poder iniciar sesión, el sistema no realiza ninguna acción que requiera la autenticación, ni expone ningún tipo de información, sólo presenta un mensaje de error de credenciales y regresa al módulo de inicio de sesión.

Garantías mínimas: se garantiza que de no poder iniciar sesión, el sistema no es afectado en su desempeño (disponibilidad del servicio), no expone información restringida (confidencialidad) y no se modifica ningún tipo de información que no deba ser alterada (integridad).

Frecuencia: alta, debido a que es una precondición de varios casos de uso posteriores.

Supuestos: se asume que el usuario ya se encuentra registrado en el sistema.

Activos relacionados:

- Información almacenada en la base de datos y su estructura.
- Información sobre la configuración del servidor de base de datos.
- La identidad del usuario que inicia sesión y hace uso del sistema.

Amenazas relacionadas:

a) Inyección de código
b) Pérdida de Autenticación y Gestión de Sesiones
c) Almacenamiento Criptográfico Inseguro
d) Falla de Restricción de Acceso a URL
e) Redirecciones y Reenvíos no validados

Definición de rebanadas:

- *1.1 Error al ingresar el nombre de usuario o contraseña*

  Historia: el usuario ingresa una contraseña o nombre de usuario no registrados en el sistema.
  Flujo: el sistema envía un mensaje de error de credenciales y regresa a la página de inicio de sesión.
  Condiciones de prueba: ingresar un nombre de usuario o contraseña inválida

- *1.2 Inyección de código SQL*

  Historia: el usuario ingresa código SQL que complete la sentencia de verificación sin que las credenciales sean necesariamente validas

Flujo: envía un mensaje de error de credenciales y regresa a la página de inicio de sesión.
Condiciones de prueba: ingresar una cadena de texto que complete la sentencia SQL sintácticamente correcta.

- *1.3 Traslado del identificador de inicio de sesión*

  Historia: el usuario ingresa al sistema y copia la URL que contiene un identificador de sesión o la obtiene del historial de navegación, accede desde otra instancia al sistema con la URL copiada sin ingresar las credenciales de acceso.
  Flujo: el sistema envía a la página de inicio de sesión.
  Condiciones de prueba: ingresa la URL con el identificador de una sesión validada.

- *1.4 Sesión perpetua*

  Historia: el usuario ingresa al sistema y después de un periodo de tiempo de inactividad superior al periodo acordado, con el acceso vigente.
  Flujo: el sistema cierra la sesión después de un periodo de tiempo establecido.
  Condiciones de prueba: ingresa la URL después de un tiempo de inactividad superior al establecido.

- *1.5 Bloqueo de inicio de sesión*

  Historia: el usuario mal intencionado busca a través de intentos consecutivos y automatizados el ingreso al sistema.
  Flujo: el sistema bloquea la cuenta después de un número de intentos consecutivos fallidos.
  Condiciones de prueba: se realiza un número de intentos fallidos igual al establecido para el bloqueo de la cuenta

- *1.6 Falla de Restricción de Acceso a URL*

  Historia: el usuario mal intencionado busca llegar a una página o módulo restringido a través de la URL.
  Flujo: el sistema declara que el sitio está restringido para el usuario.
  Condiciones de prueba: se ingresa a través de la URL la dirección de los módulos restringidos.

- *1.7 Redirecciones y reenvíos no validos*

  Historia: el usuario mal intencionado busca dirigir a otros usuarios a un sitio malicioso que el controla, a través de la redirección desde un parámetros editables por el usuario.
  Flujo: el sistema declara que el sitio a donde se busca redireccionar es inválido y cancela la acción.
  Condiciones de prueba: se manipula el parámetro de destino a un sitio fuera del sistema.

- *1.8 Ingreso de usuario y contraseña correcto*

  Historia: el usuario ingresa sus credenciales correctas.
  Flujo: despliega una página de bienvenida.
  Condiciones de prueba: contraseña y nombre de usuario correctos.

Flujo optimo:

Paso	Entrada	Respuesta del sistema
1	El usuario presiona en el botón de "Iniciar sesión" en el menú principal.	El sistema despliega un formulario con los campos de texto de inicio de sesión y contraseña.
2	El usuario se identifica ingresando en el campo de texto el nombre de un usuario registrado, en otro campo su correspondiente contraseña y adicionalmente ingresa una clave dinámica u OTP y presiona el botón "Ingresar".	El sistema autentica al usuario y le inicio sesión desplegando una página de bienvenida con las nuevas acciones que puede realizar según su perfil.
3	El usuario ingresa la URL del sistema después de un periodo de inactividad ya habiendo iniciado sesión exitosamente	Después de un periodo de inactividad el sistema cierra la sesión.

Flujos alternativos:

Paso	Entrada	Respuesta del sistema
4.1	El usuario ingresa credenciales invalidas	El sistema registra el fallo en una bitácora, envía un mensaje de error en las credenciales y redirige a la página de inicio de sesión para repetir el proceso.
4.2	El usuario ingresa sentencias SQL en alguno o varios de los campos de credenciales	El sistema registra el fallo en una bitácora, envía un mensaje de error en las credenciales y redirige a la página de inicio de sesión para repetir el proceso.
4.3	El usuario ingresa un identificador de sesión invalido	El sistema registra el fallo en una bitácora, y redirige al usuario a la página de inicio de sesión para iniciar el proceso.
4.4	El usuario intenta realizar una actividad en el sistema después de un tiempo establecido de sesión.	El sistema cierra la sesión, registra el evento en una bitácora, envía un mensaje de sesión caducada y redirige a la página de inicio de sesión para repetir el proceso de autenticación.
4.5	El usuario intenta iniciar sesión con credenciales invalidas, un número de veces consecutivas igual al establecido para el bloqueo de la cuenta	El sistema bloquea la cuenta por un periodo de tiempo establecido o hasta su reactivación manual

4.6	El usuario intenta acceder a un módulo restringido para su perfil , a través de la URL	El sistema envía un mensaje de error por falta de privilegios y restringe el acceso al módulo y registra el fallo en una bitácora.
4.7	El usuario intenta modificar los parámetros de destino para el redireccionamiento	El sistema envía un mensaje de error en el destino de redireccionamiento y cancela la acción registrando el fallo en una bitácora.

La forma de construcción de los demás casos de uso es similar, por lo que no se incluyen en este capítulo, sin embargo una descripción de algunos otros junto con sus amenazas relacionadas, se encuentran en el apéndice 7.2.

4.2 Establecer una arquitectura de seguridad posible.

La arquitectura de software es una vista de alto nivel que establece la descomposición del sistema en componentes y sus relaciones, es la distribución física del sistema y sus componentes.

La arquitectura del sistema enfocada al desarrollo seguro de aplicaciones de software, nos permite comprender la forma en la que se integran los mecanismos de seguridad, cómo se implementa la confianza y cómo se comporta el sistema bajo un ataque.

La arquitectura de seguridad debe definirse sobre todas las capas involucradas en los sistemas: base de datos, red, sistema operativo y la capa de aplicación, siendo flexible para apoyar la integración de las nuevas tecnologías. Proporcionando un enfoque modular para la autenticación y autorización, asignar niveles de seguridad de forma consistente, otorgando el nivel más bajo de acceso requeridos por el individuo (la seguridad más estricta).

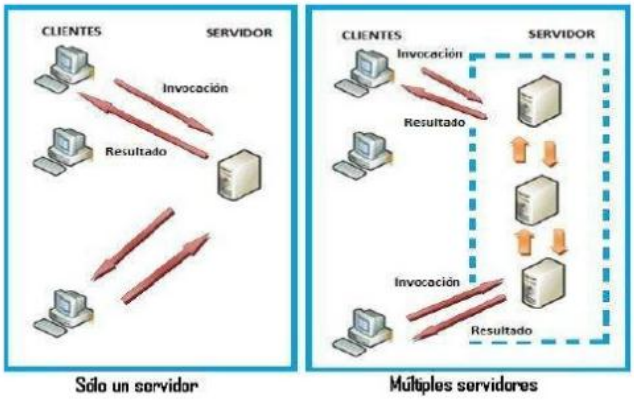
En primer lugar antes de definir una arquitectura de seguridad, es necesario realizar un análisis de riesgos arquitecturales enfocado en los artefactos de especificación, diseño y búsqueda de defectos en la autenticación, la seguridad de los componentes, la seguridad de los nodos de ejecución y problemas de protección de los datos.

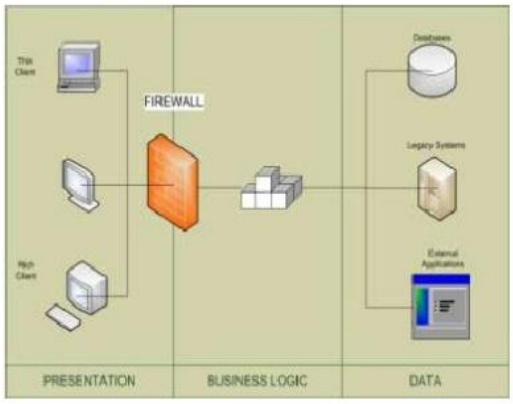
Debe especificarse un modelo a alto nivel de las amenazas y referencias a los documentos que describen los procedimientos de seguridad, también deberán enumerarse las medidas de seguridad a implementar, pero teniendo en cuenta que las tecnologías en sí no brindan seguridad.

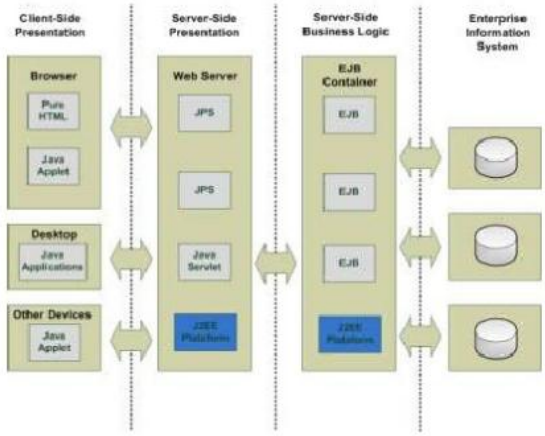
Identificar los puntos vulnerables, los componentes de diseño y reutilización común y probado, consolidar el almacenamiento de datos sensibles y garantizar el registro

auditable de las actividades sensibles de un usuario dentro del sistema, como los accesos fallidos y autorizados a la aplicación, bajas, altas o cambios en la información realizados desde el desarrollo, etc.

La arquitectura de seguridad también debe de ser capaz de documentar cómo los datos son utilizados por cada componente, a dónde van, sus transmisiones y los métodos de intercambio de información privada (cifrado). Documentar cómo se autentica una aplicación o componente a otro servicio, por ejemplo: contraseñas, certificados PKI, restricción por IP, etc.







A continuación se ejemplifica una arquitectura de seguridad multicapa, a considerar para la protección de un sistema de software y su entorno:

	Políticas, estándares, procedimientos, técnicas referentes a la arquitectura. Aprovechamiento de herramientas y el ciclo de vida Aprobación de excepciones Revisiones reguladas	

USUARIO	**RED**	**APLICACIOÓN**	**DATOS**	**OPERACIÓN**
Manejo de identificación, autenticación y educación	Administración, Firewall, cifrado	Autorización, auditoria a herramientas de prueba	Autorización, cifrado, inventario	Backups, auditoria, recuperación ante desastres

La arquitectura es un aspecto tan importante dentro del desarrollo, que es conveniente realizar actividades de verificación de forma temprana, con el fin de identificar problemas que podría resultar muy costoso eliminar posteriormente. La evaluación de la arquitectura de software permite justamente realizar la verificación del diseño y cubren el conjunto de aspectos relacionados con el desarrollo de arquitectura de software.

Los atributos de calidad son aquellas características medibles, tales como el desempeño o disponibilidad, que permiten expresar la calidad del sistema de un punto de vista del cliente y de la organización de desarrollo. Un buen diseño arquitectónico es clave para poder satisfacer este tipo de requerimientos.

Al terminar de realizar el diseño de una arquitectura, quisiéramos estar seguros que el diseño arquitectural propuesto satisface realmente requerimientos como los anteriores. El riesgo de no tener seguridad al respecto de ello de forma temprana en el desarrollo puede tener consecuencias muy serias en etapas posteriores del desarrollo y muy particularmente, si se descubren problemas relacionados con la arquitectura en etapas tardías tales como la implantación del sistema.

La evaluación de la arquitectura de software es una herramienta que ayuda a mitigar riesgos como el descrito anteriormente. Para lograrlo, la evaluación busca esencialmente responder a la pregunta siguiente: ¿el diseño de la arquitectura satisface a los requerimientos que influyen a la arquitectura y, principalmente, a los atributos de calidad? (33)

Capítulo 5 Modificación del Proceso Unificado Ágil con aspectos de seguridad (Implementación)

- 5.1 Implementación
- 5.2 Implementación de los criterios de seguridad, control y auditoría del sistema
- Codificación segura de programas

5 MODIFICACIÓN DEL PROCESO UNIFICADO ÁGIL CON ASPECTOS DE SEGURIDAD (IMPLEMENTACIÓN)

5.1 Implementación

Este es uno de los flujos más técnicos del sistema, con una alta interacción y comunicación entre el equipo de analistas, diseñadores y el equipo de programadores. En este momento es donde se hace uso de las prácticas de codificación segura.

La implementación tiene su mayor carga de trabajo durante la fase de construcción, la cual asume como propósito, el completar la funcionalidad del sistema de software

Referente al requerimiento de seguridad, es aquí en donde se plasma de manera concreta todas las medidas de protección contempladas en los capítulos anteriores.

El objetivo de la fase de construcción es clarificar los requerimientos faltantes y completar el desarrollo del sistema basados en la arquitectura base.

Las herramientas proporcionadas por UML, se orientan a mantener la comunicación y un diálogo entre los analistas, diseñadores y los programadores, correspondiendo a las señaladas en los flujos de análisis y diseño. Vista de cierta forma esta fase es un proceso de manufactura.

5.1.1 Objetivos

- Distribuir el sistema a través de un mapeo de componentes ejecutables dentro del modelo de desarrollo
- Implementar el diseño de las clases y subsistemas encontrados durante el modelado. En particular, el diseño de clases es implementado como un archivo de componentes que contiene el código fuente.

5.1.2 Actividades

Este flujo de trabajo está compuesto por las siguientes actividades:

- Asignación de los programas especificados a los programadores.
- Definición de los estándares de codificación.
- Codificación segura de programas (actividad de seguridad).
- Implementación de la base de datos del módulo o sistema.

- Implementación de los criterios de seguridad, control y auditoría del sistema (actividad de seguridad).

5.1.3 Productos

- Diagrama de clases
- Diagrama de Secuencia
- Diagrama de componentes
- Diagrama de estados
- Diagrama de distribución
- Modelo entidad relación incorporando registros de auditoría (producto de seguridad)
- Código seguro (producto de seguridad)

Todos los productos anteriores se obtienen de la manera tradicional, por lo que en este capítulo, sólo se describe el correspondiente al diagrama entidad relación incluyendo registro de auditoría y la codificación segura.

5.2 Implementación de los criterios de seguridad, control y auditoría del sistema

Todo sistema debe contar con registros de seguridad, control y auditoria que proporcionen una traza de las condiciones y uso del sistema, puntualmente, sobre datos presente en ciertos eventos estratégicos. Por ejemplo:

- Inicio de sesión
- Alta, baja o modificación de usuarios
- Alta, baja o modificación de objetos
- Errores
- Manejo de permisos
- Cierre de sesión
- Tiempo de inactividad

La definición de los eventos depende principalmente del cliente según sean sus objetivos, sin embargo, este último puede apoyarse del asesor de seguridad para identificar aquellos en los que se pueda tener mayor valor, así como la granularidad de la información recopilada, con el objetivo de potencializar esa fuente de datos, no sólo para fines estadísticos, sino como mecanismo de alerta durante algún ataque, o como evidencia para deslindar responsabilidades sobre un sistema comprometido.

Esta información proporciona un panorama en tiempo real del uso de las aplicaciones, errores tanto de programación como de la arquitectura, intentos de acceso no autorizado, trazabilidad de las actividades de un usuario, etc.

Estos datos deben ser considerados al momento de diseñar el diagrama entidad relación, con el fin de definir un espacio para la permanencia de los mismos, así como el incremento de trabajo y recursos que significa recaudar, almacenar y procesar esa información.

Esta actividad genera el diagrama de entidad relación incorporando registros de seguridad. Para el cual también se debe de contemplar las clases y objetos, que se emplearán para generarla, módulos de seguridad que pueden llegar ser tan complejos como los funcionales y para los cuales hay que destinar recursos y considerar que incrementan por sí mismos, el tamaño del proyecto.

Un ejemplo de los registros a considerar puede ser durante el caso de uso de inicio de sesión, donde la información valiosa (para fines demostrativos) puede ser: la dirección IP del usuario, el nombre de usuario, el password con el que intento acceder, la fecha y hora del intento, el resultado del evento (satisfactorio o no), el estado del usuario en el sistema (en línea o desconectado).

Esta información nos puede indicar cuando estamos siendo objeto de un ataque de fuerza bruta, desde donde se está ejecutando el ataque, si se trata de un sistema automatizado o de una persona real, si se emplea algún diccionario o son pruebas al azar, si la cuenta ya fue comprometida (el atacante ingreso satisfactoriamente) y desde cuándo, la frecuencia de sus accesos y si se encuentra dentro del sistema actualmente.

Según los resultados del análisis de la información podemos tomar medidas preventivas, mitigantes, correctivas o tratarla como evidencia. Continuando con el escenario anterior, un ejemplo de medida preventiva es la de bloquear el acceso al sistema previa a una autenticación exitosa del atacante. Como medida mitigante, es la de cerrar la sesión y bloquear la cuenta si el atacante ya logro algún acceso. Correctiva, si se identificó algún error en la programación que permite el acceso, corregirlo inmediatamente y como evidencia, conservar la traza de intentos de acceso.

5.3 Codificación segura de programas

A continuación, describimos algunos escenarios y sus recomendaciones de protección, en base a las 10 vulnerabilidades más comunes, descritas en el Capítulo 2.

5.3.1 Inyección de código

5.3.1.1 Escenario:

El caso de uso, recibe las credenciales habituales de autenticación (usuario y password), desde un formulario y realiza la construcción de la siguiente consulta SQL vulnerable.

```
$consulta = "SELECT * FROM cuentas WHERE usuario='" + $_GET["usuario"] +"' AND password= '"
+ $_GET["password"] +"'";
```

El atacante puede modificar el parámetro "usuario" dentro del navegador para enviar la cadena: *usuario_existente* ' --. Esto produce que la consulta regrese la información del usuario sin importar la contraseña que se introduzca. Así que la petición enviada por el atacante es:

http://ejemplo.com/verCuenta.php?usuario=*usuario_existente* ' --&password=no_importa

La consulta a ejecutar en el servidor de base de datos resultante es:
SELECT * FROM cuentas WHERE usuario='*usuario_existente* ' --' AND password= 'no_importa'

Los caracteres -- convierten a la cadena siguiente en un comentario dentro del código SQL por lo que de manera efectiva se ejecuta:

SELECT * FROM cuentas WHERE usuario='*usuario_existente* '

Esto permite el acceso sin importar la contraseña. En el peor de los casos el atacante utiliza esta debilidad para invocar procedimientos almacenados especiales en la base de datos, lo que permite que el servicio de base de datos se vea completamente comprometido.

5.3.1.2 Método de protección

Prevenir la inyección requiere mantener los datos no confiables separados de comandos y consultas.

1. La opción más común es utilizar un API segura que provea una interfase parametrizada, sin embargo es importante ser cuidadoso con APIs, tales como los procedimientos almacenados, que aunque son parametrizados, pueden permitir la inyección de manera implícita.

2. Si una API parametrizada no se encuentra disponible, se deben de escapar los caracteres especiales utilizando una sintaxis de escape especial para dicho intérprete.

3. Una validación positiva de entradas con una apropiada selección de caracteres permitidos es también recomendado, pero no es una defensa completa ya que muchas aplicaciones requieren caracteres especiales en sus entradas.

5.3.2 Cross Site Scripting XSS

5.3.2.1 Escenario:

La aplicación utiliza datos no confiables en la construcción del siguiente código HTML sin validar o escapar los datos:

$pagina += "<input name='creditcard' type='TEXT‘ value='" + $_GET["CC"] + "'>";

El atacante modifica el parámetro ‘CC’ en el navegador:

'><script>document.location=
'http://www.attacker.com/cgi-bin/cookie.cgi?
foo='+document.cookie</script>'.

Esto causa que el identificador de sesión de la víctima sea enviado al sitio web del atacante, permitiendo al atacante secuestrar la sesión actual del usuario. Notar que los atacantes pueden también utilizar XSS para anular cualquier defensa CSRF que la aplicación pueda utilizar.

5.3.2.2 Método de protección

Prevenir XSS requiere mantener los datos no confiables separados del contenido activo del navegador.

1. La opción preferida es escapar todos los datos no confiables basados en el contexto HTML (cuerpo, atributo, JavaScript, CSS, o URL) donde los mismos serán ubicados.

2. Una validación de entradas positiva o “whitelist” con apropiada canonicalización y decodificación es también recomendable ya que ayuda a proteger contra XSS, pero no es una defensa completa ya que muchas aplicaciones requieren caracteres especiales en sus entradas. Tal validación debería, tanto como sea posible,

decodificar cualquier entrada codificada, y luego validar la longitud, caracteres, formato, y cualquier regla de negocio en dichos datos antes de aceptar la entrada.

5.3.3 Pérdida de Autenticación y Gestión de Sesiones

5.3.3.1 Escenario:

Escenario #1: Aplicación que soporta la reescritura de direcciones URL poniendo los identificadores de sesión en la propia dirección:

http://sitio.com/sale/saleitems;jsessionid=2P0OC2JDPXM0OQSNDLPSKHCJUN2JV?objeto=objeto1

Un usuario autenticado en el sitio quiere mostrar un objeto del sitio a otras personas. Envía por correo electrónico el enlace anterior, sin ser consciente de que está proporcionando su identificador de sesión. Cuando sus amigos utilicen el anterior enlace utilizarán su sesión.

Escenario #2: No se establecen correctamente los tiempos de desconexión en la aplicación. Un usuario utiliza una computadora pública para acceder al sitio. En lugar de utilizar la función de "Cerrar sesión", cierra la pestaña del navegador o cambia la URL a otro sitio y se marcha. Un atacante utiliza el mismo navegador al cabo de una hora, y ese navegador todavía se encuentra autenticado.

Escenario #3: Un atacante dentro de la organización, o externo, consigue acceder a la base de datos de contraseñas del sistema. Las contraseñas de los usuarios no se encuentran cifradas, comprometiendo todas las cuentas.

Escenario #4: Un perpetrador a través de un ataque de fuerza bruta intenta de manera automática el acceso al sistema hasta que logra el acceso.

5.3.3.2 Metodología de protección

La recomendación principal para una organización es facilitar a los desarrolladores:

1. Un único y fuerte conjunto de controles de autenticación y gestión de sesiones. Dichos controles deberán conseguir:

   a) Reunir todos los requisitos de administración de sesiones y autenticación
   b) Tener un interfaz simple para los desarrolladores.

2. Se recomienda limitar las sesiones a un periodo de tiempo de inactividad, aunque esto puede afectar en algunos casos la comodidad del usuario, previene un acceso no autorizado.

3. Generar un formulario con un campo oculto dinámico que es generado por cada solicitud. El contenido del campo debe ser generado aleatoriamente utilizando un buen sistema de PRNG (Pseudo Random Number Generator) para que no pueda ser adivinado mediante técnicas estadísticas. Este campo se guardará en el servidor y será eliminado apenas el formulario sea utilizado ya que una nueva solicitud del formulario deberá ser acompañada por un nuevo contenido en el campo. Si se recibe un formulario con un contenido no válido, simplemente se rechaza la solicitud. La técnica puede enriquecerse asignando un nombre aleatorio al campo mediante el mismo concepto. Esta técnica también mitiga la vulnerabilidad de XSRF y evita que un usuario introduzca dos veces la misma transacción (lo que puede suceder fácilmente si el usuario presiona el botón de enviar dos veces porque la conexión es lenta y no ha recibido la respuesta esperada). Además este sistema evita también que el formulario sea escaneado automáticamente en busca de vulnerabilidades, ya que la gran mayoría de los "scanners" de vulnerabilidades, cargarán el formulario una sola vez para luego proceder a modificar con el contenido de los diferentes campos.

4. Otra técnica que evita el inicio de sesión aun cuando se comprometan las contraseñas de acceso es a través de un OTP (One Time Passwords) o dispositivo que genera claves dinámicas de acceso de un solo uso, es una solución definitiva para muchos otros problemas, sin embargo su costo es el principal factor a tomar en cuenta. Este tipo de solución podría ser sobre dimensionada dependiendo del tipo de aplicación y la cantidad de usuarios de la misma.

5. La incorporación de un *captcha* (una prueba de turing) podría evitar los intentos automáticos de inicio de sesión, frenando un ataque por fuerza bruta sin embargo existen técnicas que pueden reconocer los *captchas* en casos particulares, pero su implementación es muy económica.

6. Bloqueo de sesión por una serie consecutiva de intentos fallidos. Esta técnica entorpece y retrasa el éxito de un ataque de fuerza bruta hasta convertirlo en inviable para un atacante.

7. Se debe hacer especial hincapié en evitar vulnerabilidades de XSS que podrían ser utilizadas para robar identificadores de sesión, más adelante se describe cómo evitar este tipo de fallos.

5.3.4 Referencia Directa Insegura a Objetos

5.3.4.1 Escenario:

La aplicación utiliza datos no verificados en una llamada SQL que accede a información sobre la cuenta:

```
$query = "SELECT * FROM accts WHERE account = ?";
PreparedStatementpstmt = connection.prepareStatement(query , … );
pstmt.setString( 1, request.getparameter("acct"));
ResultSetresults = pstmt.executeQuery( );
```

El atacante simplemente modificaría el parámetro “acct” en su navegador para enviar cualquier número de cuenta que quiera. Si esta acción no se verifica, el atacante podría acceder a cualquier cuenta de usuario, en vez de a su cuenta de cliente correspondiente.
http://example.com/app/accountInfo?acct=notmyacct

5.3.4.2 Metodología de protección

Prevenir referencias inseguras a objetos directos requiere seleccionar una manera de proteger los objetos accesibles por cada usuario (por ejemplo, identificadores de objeto, nombres de archivos):

1. Utilizar referencias indirectas por usuario o sesión. Esto evitaría que los atacantes accedan directamente a recursos no autorizados. Por ejemplo, en vez de utilizar la clave del recurso de base de datos, se podría utilizar una lista de 6 recursos para utilizar los números del 1 al 6 e indicar cuál es el valor elegido por el usuario. La aplicación tendría que realizar la correlación entre la referencia indirecta con la clave de la base de datos correspondiente en el servidor.

2. Comprobar el acceso. Cada uso de una referencia directa a un objeto de una fuente que no es de confianza debe incluir una comprobación de control de acceso para asegurar que el usuario está autorizado a acceder al objeto solicitado.

5.3.5 Falsificación de Peticiones en Sitios Cruzados (CSRF)

5.3.5.1 Escenario:

La aplicación permite que los usuarios envíen peticiones de cambio de estado que no incluyen nada secreto. Por ejemplo:

http://example.com/app/transferFunds?amount=1500&destinationAccount=4673243243

El atacante puede construir una petición que transfiera dinero desde la cuenta de la víctima a su propia cuenta. Podrá insertar su ataque dentro de una etiqueta de imagen en un sitio web, o iframe, que esté bajo su control y al que la víctima se podrá dirigir.

<imgsrc="http://example.com/app/transferFunds?
amount=1500&destinationAccount=attackersAcct#“
width="0" height="0" />

Cuando la víctima visite el sitio, en lugar de cargarse la imagen, se realizará la petición HTTP falsificada. Si la víctima previamente había adquirido privilegios entonces el ataque será exitoso.

5.3.5.2 Metodología de protección

Para prevenir la CSFR se necesita incluir un testigo no predecible en el cuerpo, o URL, de cada petición HTTP. Dicho testigo debe ser, como mínimo, único por cada sesión de usuario.

1. La opción preferida es incluir el testigo en un campo oculto. Esto genera que el valor sea enviado en el cuerpo de la petición HTTP evitando su inclusión en la URL, lo cual está sujeto a una mayor exposición.

2. El testigo único también puede ser incluido en la URL misma, o en un parámetro de la URL. Sin embargo, este enfoque presenta el riesgo que la URL sea expuesta a un atacante, y por lo tanto exponiendo al testigo.

5.3.6 Defectuosa configuración de seguridad

5.3.6.1 Escenario:

El trabajo para evitar esta vulnerabilidad es una actividad que no se trata directamente en esta tesis, ya que el PUA originalmente establece que la administración de la configuración debe ser consistente y constante, tratando las vulnerabilidades como cualquier fallo y dando seguimiento a sus correcciones de la misma manera. Sin embargo, a continuación se plantean algunos escenarios y las prácticas de mitigación con el objetivo de ejemplificar esta vulnerabilidad.

Escenario #1: La aplicación está basada en un ambiente de trabajo como Struts o Spring. Se han presentado defectos de XSS en algunos de los componentes que utiliza la

aplicación. Se ha liberado una actualización que sirve para corregir esos defectos. Hasta que no se realicen dichas actualizaciones, los atacantes podrán encontrar y explotar los fallos, ahora conocidos, de la aplicación.

Escenario #2: La consola de administración del servidor de aplicaciones está instalada y no ha sido removida. Las cuentas predeterminadas no han sido cambiadas. Los atacantes descubren que las páginas de administración están activas, se registran con las claves predeterminadas y toman posesión de los servicios.

Escenario #3: El listado del contenido de los directorios no está deshabilitado en el servidor. Los atacantes descubren que pueden encontrar cualquier archivo simplemente consultando el listado de los directorios. Los atacantes encuentran y descargan las clases java compiladas. Dichas clases son desensambladas por ingeniería inversa para obtener su código. A partir de un análisis del código se pueden detectar defectos en el control de acceso de la aplicación.

Escenario #4. La configuración del servidor de aplicaciones permite que los mensajes de la pila sean retornados a los usuarios. Eso potencialmente expone defectos en la aplicación. La información de error que dichos mensajes proveen es un recurso valioso para los atacantes.

5.3.6.2 Metodología de protección

Las principales recomendaciones se enfocan en establecer lo siguiente:

1. Un proceso repetible que permita configurar, rápida y fácilmente, entornos asegurados. Los entornos de desarrollo, pruebas y producción deben estar configurados de la misma forma. Este proceso debe ser automatizado para minimizar el esfuerzo requerido en la configuración de un nuevo entorno.

2. Un proceso para mantener y desplegar todas actualizaciones y parches de software de manera oportuna. Este proceso debe seguirse en cada uno de los ambientes de trabajo. Es necesario que se incluya las actualizaciones de todas las bibliotecas de código.

3. Una arquitectura robusta de la aplicación que provea una buena separación y seguridad entre los componentes.

4. Considerar la realización periódica de exploraciones y auditorias para ayudar a detectar fallos en la configuración o parches faltantes.

5.3.7 Almacenamiento Criptográfico Inseguro

5.3.7.1 Escenario:

Escenario #1: Una aplicación cifra las tarjetas de crédito en la base de datos para prevenir que sean expuestos a los usuarios finales. Sin embargo, la base de datos descifra automáticamente las columnas de las tarjetas de crédito, permitiendo que una vulnerabilidad de inyección de SQL pueda extraer las tarjetas de crédito en texto plano. El sistema debería haberse configurado de manera que solo las aplicaciones del back-end pudieran descifrar los datos, no la capa frontal de la aplicación web.

Escenario #2: Una cinta de backup almacena datos médicos cifrados pero la clave de cifrado se encuentra en el mismo backup. La cinta nunca llega al centro de copias de seguridad.

Escenario #3: La base de datos de contraseñas usa hashes sin sal para almacenar las contraseñas de todos los usuarios. Una vulnerabilidad en la subida de archivos permite a un atacante obtener el archivo de contraseñas. Todos los hashes sin sal se pueden romper en 4 semanas, mientras que los hashes con sal llevarían más de 3000 años dependiendo de la longitud de la sal.

5.3.7.2 Metodología de protección

El listado de todos los peligros del cifrado inseguro está fuera del alcance de este documento. Sin embargo, para todos los datos sensibles que requieran cifrado, es necesario hacer como mínimo lo siguiente:

1. Considerar las amenazas que afecten a los datos y de las cuales se quiere proteger (por ejemplo, ataques internos, usuarios externos) y asegúrese de que todos los datos están cifrados de manera que se defienda de las amenazas.

2. Asegurar que las copias de seguridad almacenadas externamente están cifradas, pero las claves son gestionadas y almacenadas de forma separada.

3. Asegurar el uso adecuado de algoritmos estándares robustos, que las claves usadas son fuertes y que existe una gestión de claves adecuada.

4. Asegurar que sus contraseñas se almacenan en forma de hash con un algoritmo estándar robusto y con sal.

5. Asegurar que todas las claves y contraseñas son protegidas contra acceso no autorizado.

5.3.8 Falla de Restricción de Acceso a URL

5.3.8.1 Escenario:

El atacante simplemente navega forzosamente en la URL objetivo. Considere las siguientes URLs las cuales se supone que requieren autenticación. Para acceder a la página "admin_getappInfo" se necesitan permisos de administrador.

http://ejemplo.com/app/getappInfo
http://ejemplo.com/app/admin_getappInfo

Si un atacante no autenticado puede acceder a cualquiera de estas páginas entonces se ha permitido acceso no autorizado. Si un usuario autorizado, no administrador, puede acceder a la página "admin_getappInfo", esto es un fallo, y puede llevar al atacante a otras páginas de administración que no están debidamente protegidas.

Este tipo de vulnerabilidades se encuentran con frecuencia cuando links y botones simplemente se ocultan a usuario no autorizados, pero la aplicación no protege adecuadamente las páginas de destino.

5.3.8.2 Metodología de protección

Prevenir el acceso no autorizado a URLs requiere planificar un método que requiera autenticación y autorización adecuadas para cada página. Frecuentemente, dicha protección viene dada por uno o más componentes externos al código de la aplicación. Con independencia del mecanismo o mecanismos se recomienda:

1. Que la autenticación y autorización estén basadas en roles, para minimizar el esfuerzo necesario para mantener estas políticas.

2. Las políticas deberían ser configurables, para minimizar cualquier aspecto embebido en la política.

3. La implementación del mecanismo debería negar todo acceso por defecto, requiriendo el establecimiento explícito de permisos a usuarios y roles específicos por cada página.

4. Si la página forma parte de un proceso de varios pasos, verifique que las condiciones de la misma se encuentren en el estado apropiado para permitir el acceso.

5.3.9 Protección Insuficiente en la capa de Transporte

5.3.9.1 Escenario:

Al igual que otros fallos de seguridad, la protección insuficiente en la capa de transporte puede no pertenecer a un problema de desarrollo, sino de configuración del servidor o de elección de los métodos de transporte de datos, sin embargo a continuación se describen algunos escenarios para ejemplificar y corregir este fallo.

Escenario #1: Una aplicación no utiliza SSL para todas las páginas que requieren autenticación. El atacante simplemente captura el tráfico de red (por ejemplo, a través de una red inalámbrica abierta o un sistema vecino de su red cableada), y observa la cookie de sesión de una víctima autenticada.

Escenario #2: Una aplicación utiliza un certificado SSL configurado incorrectamente, lo que provoca que el navegador muestre advertencias a sus usuarios. Los usuarios tienen que aceptar dichas advertencias y continuar para poder acceder a la aplicación. Esto hace que los usuarios se acostumbren a estos avisos. Un ataque de phishing contra la aplicación atrae los clientes a otra aplicación de apariencia similar a la original que no dispone de un certificado válido, lo que genera advertencias similares en el navegador. Como las víctimas se encuentran acostumbradas a dichas advertencias, proceden a acceder al sitio de phishing facilitando contraseñas u otra información sensible.

Escenario #3: Una aplicación simplemente utiliza ODBC/JDBC para la conexión con la base de datos, sin darse cuenta de que todo el tráfico se transmite en claro.

5.3.9.2 Metodología de protección

Proporcionar una protección adecuada a la capa de transporte puede afectar al diseño de la aplicación. De esta forma, resulta más fácil requerir SSL para la aplicación completa. Por razones de rendimiento, algunas aplicaciones utilizan SSL únicamente para acceder a páginas privadas. Otras, utilizan SSL sólo en páginas “críticas”, pero esto puede exponer identificadores de sesión y otra información sensible. Como mínimo, se debería aplicar lo siguiente:

1. Requerir SSL para todas las páginas sensibles. Las peticiones sin SSL a estas páginas deben ser redirigidas a las páginas con SSL.

2. Establecer el atributo “secure” en todas las cookies sensibles.

3. Configurar el servidor SSL para que acepte únicamente algoritmos considerados fuertes (por ejemplo, que cumpla FIPS 140-2).

4. Verificar que el certificado sea válido, no se encuentre expirado o revocado y que se ajuste a todos los dominios utilizados por la aplicación.

5. Conexiones a sistemas finales (back-end) y otros sistemas también deben utilizar SSL u otras tecnologías de cifrado.

5.3.10 Redirecciones y Reenvíos no validados

5.3.10.1 Escenario:

Escenario #1: La aplicación tiene una página llamada “redirect.jsp” que recibe un único parámetro llamado “url”. El atacante compone una URL maliciosa que redirige a los usuarios a una aplicación que realiza el phishing e instala código malicioso.

http://www.example.com/redirect.jsp?url=evil.com

Escenario #2: La aplicación utiliza destinos para redirigir las peticiones entre distintas partes de la aplicación. Para facilitar esto, algunas páginas utilizan un parámetro para indicar dónde será dirigido el usuario si la transacción es correcta. En este caso, el atacante compone una URL que evadirá el control de acceso de la aplicación y llevará al atacante a una función de administración a la que en una situación habitual no debería tener acceso.

http://www.example.com/boring.jsp?fwd=admin.jsp

5.3.10.2 Metodología de protección

Puede realizarse un uso seguro de redirecciones y re-envíos de varias maneras:

1. Simplemente, evitando el uso de redirecciones y reenvíos.

2. Si se utiliza, no involucrar parámetros manipulables por el usuario para definir el destino.

3. Si los parámetros de destino no pueden evitarse, asegúrese de que el valor facilitado es válido y autorizado para el usuario. Se recomienda que el valor de cualquier parámetro de destino sea un valor de mapeo, en lugar de la dirección, o parte de la dirección, de la URL y en el código del servidor traducir dicho valor a la dirección URL de destino. Las aplicaciones pueden utilizar ESAPI para sobrescribir el método “sendRedirect()” y asegurarse de que todos los destinos redirigidos son seguros.

Evitar estos problemas resulta extremadamente importante ya que son un blanco preferido por los phishers que intentan ganarse la confianza de los usuarios.

Capítulo 6 Modificación del Proceso Unificado Ágil con aspectos de seguridad (Pruebas)

- 6.1 Pruebas
- 6.2 Pruebas de seguridad
- 6.3 Pruebas de integración considerando la seguridad

6 MODIFICACIÓN DEL PROCESO UNIFICADO ÁGIL CON ASPECTOS DE SEGURIDAD (PRUEBAS)

6.1 Pruebas

El flujo de pruebas, asegura la correcta funcionalidad del sistema, en cuestión de la seguridad, es aquí en donde se revisa la correcta implementación de las medidas de protección, así mismo se busca detectar fallos o errores que se hayan pasado por alto desde la fase de inicio, para lo cual es necesario que sean correctamente planificadas, ejecutadas y documentadas con el objetivo de que los resultados aporten la mayor cantidad de información valiosa.

6.1.1 Objetivos

- Diseñar e implementar las pruebas a través de los casos de prueba especificados, creando un procedimiento de pruebas que especifique cómo funcionan las mismas, automatizando las mismas, si es posible
- Manejar sistémicamente los resultados de las pruebas y en lo posible regresar a los flujos de trabajo, en donde se generan los fallos, para poder corregirlos

6.1.2 Actividades

Este flujo de trabajo está compuesto por las siguientes actividades:

- Planificar las Pruebas
- Diseñar las Pruebas
- Implementar las Pruebas
- Ejecutar las Pruebas en la etapa de Integración de pruebas
- Ejecutar las Pruebas en la etapa de pruebas del sistema
    - Ejecutar las pruebas de penetración (actividad de seguridad)
    - Ejecutar las pruebas de análisis estático (actividad de seguridad)
- Evaluar las Pruebas

6.1.3 Productos

- Plan de pruebas
- Casos de prueba
- Procedimientos de prueba
- Resultado de las pruebas de integración

- Resultado de las pruebas del sistema
    - Resultado de pruebas de penetración (producto de seguridad)
    - Resultado de pruebas de análisis estático (producto de seguridad)
- Evaluación de pruebas y requerimientos de cambio

Si se desea tener un programa de pruebas, es necesario saber cuáles son los objetivos de las pruebas. Éstos objetivos son especificados por los requisitos de seguridad. Uno de los objetivos de las pruebas de seguridad es validar que las funciones de control de seguridad funcionan como se esperaba. Esto está documentado a través de los requisitos de seguridad que describen la funcionalidad del control de seguridad para este las técnicas que se plantean en este documento, el plan de pruebas se encuentra implícito en la descripción de rebanadas de los casos de uso 2.0, sin embargo es importante aclarar que entre más granular se hayan descrito dichas rebanadas respecto a la seguridad, será menos probable la existencia de estas, ya que el desarrollo se enfoca en resolverlas. En un alto nivel, esto significa probar la confidencialidad, integridad y disponibilidad de los datos, así como el servicio. El otro objetivo es validar que los controles de seguridad estén implementados con pocos o sin vulnerabilidades (22).

6.2 Pruebas de seguridad

Las pruebas de seguridad representa la primera oportunidad de los programadores para garantizar que los componentes de software que han desarrollado, hayan sido probados a nivel de seguridad antes de que sean integrados con otros componentes e incluidos en la aplicación. Los componentes de software podrían consistir en elementos de software, tales como funciones, métodos y clases, así como interfases de programación, bibliotecas y ejecutables. Para probar la seguridad, los programadores pueden confiar en los resultados del análisis de código fuente para verificar estáticamente que el código fuente desarrollado no incluye potenciales vulnerabilidades y que es compatible con las normas de codificación segura.

Las pruebas de seguridad pueden verificar dinámicamente (es decir, en tiempo de ejecución) que los componentes funcionan como se esperaba. Antes de la integrar los nuevos y existentes cambios en el código en la construcción de la aplicación, los resultados de análisis estático y dinámico deben ser revisados y validados.

La validación de código fuente antes de la integración en la aplicación es, por lo general, responsabilidad del programador en combinación con el asesor de seguridad. Su función es liderar la revisión segura de código y tomar decisiones sobre si aceptará o no el código que se integrará en la construcción de la aplicación o requerir más cambios y pruebas. Este flujo de revisión segura del código puede ser realizado a través de la aceptación formal, también como una aceptación dentro de una herramienta de gestión de flujo de trabajo. Por ejemplo, asumiendo que la típica gestión de defectos usado para errores

funcionales, los errores de seguridad que han sido solucionados por un programador pueden ser informados sobre el sistema de administración de cambios. El integrador principal puede ver los resultados de las pruebas reportadas por los programadores y dar autorización de integrar el cambio en la aplicación (22).

Una buena práctica para los programadores es construir casos de prueba de seguridad como un conjunto de pruebas generales de seguridad que forma parte del actual framework de pruebas unitarias, para probar la seguridad de funciones, métodos y clases. Un conjunto de pruebas generales de seguridad podrían incluir casos de prueba de seguridad para validar tanto requerimientos positivos como negativos para los controles de seguridad tales como:

- Control de Acceso y Autenticación
- Codificación y Validación de entrada
- Cifrado
- Gestión de Sesiones y de Usuarios
- Gestión de Errores y Excepciones
- Auditoría y Registro

Los programadores armados con una herramienta de análisis de código fuente integrada en su IDE, normas de codificación de seguridad, y un framework de pruebas de seguridad unitarias pueden evaluar y verificar la seguridad de los componentes de software que están siendo desarrolladas. Casos de prueba de seguridad pueden ser ejecutadas para detectar posibles problemas de seguridad que tienen causas profundas en el código fuente: además de validación de entrada y salida de los parámetros que entran y salen de los componentes, estas cuestiones incluyen los controles de autenticación y autorización realizada por el componente, la protección de los datos dentro del componente, excepciones de seguridad y manejo de errores, y auditoría y registro. Frameworks de pruebas unitarias tales como JUnit, Nunit, Cunit. Pueden ser adaptados para verificar los requerimientos de pruebas de seguridad. En el caso de las pruebas funcionales de seguridad, el nivel de pruebas unitarias puede probar la funcionalidad de los controles de seguridad a nivel de componentes del software, tales como funciones, métodos o clases. Por ejemplo, un caso de prueba podría validar la entrada y la salida, y los límites de los controles de las variables preguntando por la funcionalidad del componente.

Los escenarios de amenaza identificados con los casos de uso 2.0, pueden ser usados para documentar los procedimientos para pruebas de componentes de software. En el caso de los componentes de autenticación, por ejemplo, pruebas unitarias de seguridad puede validar la funcionalidad de bloqueo de una cuenta, así como el hecho de que los parámetros de entrada de usuario no puede ser objeto de abuso para evitar el bloqueo de cuenta (por ejemplo, mediante el establecimiento de un contador para cuenta bloqueada a un número negativo). A nivel de componente, pruebas unitarias de seguridad pueden validar afirmaciones positivas como afirmaciones negativas, tales como los errores y

manejo de excepciones. Las excepciones deben ser capturadas sin abandonar el sistema en un estado inseguro, también como una denegación de servicio causada por los recursos que no han sido liberados (por ejemplo, manejo de conexiones no cerradas dentro de un bloque de declaración final), así como una elevación de privilegios (por ejemplo, el aumento de privilegios adquiridos antes de que la excepción sea lanzada y no volver a restablecerse el previo nivel antes de salir de la función). El manejo seguro de errores puede evitar la divulgación de información a través de mensajes de error y registros.

6.2.1 Análisis estático para la seguridad (pruebas de caja blanca)

Las pruebas de caja blanca tienen la peculiaridad de que se cuenta con el código fuente de la aplicación, el cual se analiza sin ejecutar el software.

Las ventajas de este tipo de análisis son:

- Consistencia. La herramienta ve lo que ve, sin ideas preconcebidas (que normalmente tienen los desarrolladores o revisores).
- Apuntan a la causa raíz, no a los síntomas. Una prueba de penetración puede establecer que hay un problema, pero no su causa final ni cómo corregirlo.
- Detección precoz. La aplicación no tiene que estar integrada ni necesita ejecutarse. Su ejecución es barata. Un sistema puede re-analizarse cuando se aplican cambios, o cuando se descubre una nueva vulnerabilidad.

Los inconvenientes del análisis estático son:

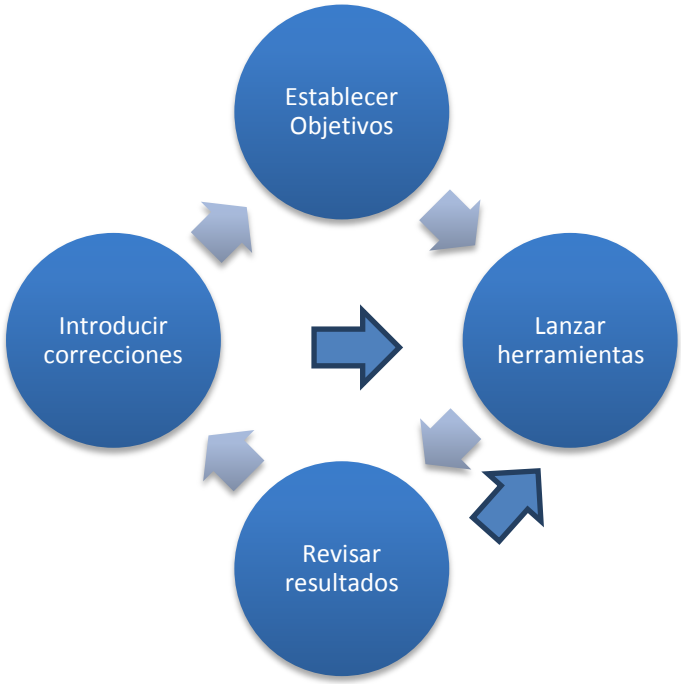
- Falsos positivos. Impacto (costo) crece al tener que evaluar cada positivo.
- Falsos negativos. Suelen ser incapaces de detectar vulnerabilidades de seguridad achacables al diseño, o específicas del contexto propio de la aplicación (se centran en vulnerabilidades genéricas, de codificación).

Las pruebas de caja blanca, pueden ser ejecutadas a nivel de componentes unitarios, y ya sea el desarrollador o los ingenieros de pruebas o inclusive ambos, apoyados del asesor de seguridad, las efectúan haciendo uso de herramientas que ayudan a esta labor, sin embargo, no basta con las pruebas automáticas, sino que hay que hacer un reprocesamiento manual. Para el análisis estático, se sugiere el siguiente ciclo de revisión:

1. Establecer objetivos
    - Priorizar qué partes a analizar
    - Entender el software (a alto nivel)
    - Comprender riesgos
2. Lanzar herramientas

- Introducir reglas específicas
- Compilar el código; lanzar herramienta

3. Revisar resultados
   - Revisión manual a partir de problemas potenciales
   - Si falso positivo: Reconfigurar; Si falso negativo: Añadir regla
   - Registro de problemas (informe formal, gestor de defectos…)
4. Introducir correcciones
   - Revisar (manual/automáticamente) cambios
   - Actualizar "Buenas prácticas", objetivos alcanzados, y reglas



6.3 Pruebas de integración considerando la seguridad

Después que los componentes y cambios en el código sean probados por los programadores y verificados en la construcción de la aplicación por el asesor de seguridad, el siguiente paso más probable en el proceso de desarrollo de software es realizar pruebas sobre la aplicación como un todo. Este nivel de pruebas es generalmente denominado pruebas de integración. Cuando las pruebas de seguridad son parte de estas actividades, ellas pueden ser utilizadas para validar la seguridad de la funcionalidad en la aplicación como un todo, así como la exposición del nivel de vulnerabilidades en la aplicación. Estas pruebas de seguridad sobre la aplicación incluyen tanto pruebas de caja blanca, tales como análisis estático de código fuente, y pruebas de caja negra, tales como pruebas de

penetración. Las pruebas de caja gris son similares a las pruebas de caja negra. En las pruebas de caja gris podemos asumir que tenemos algún conocimiento parcial de la gestión de sesiones de nuestra aplicación, y que debería ayudarnos a entender que las funciones de desconexión estén debidamente aseguradas.

El objetivo de las pruebas de seguridad es el sistema completo que es el objeto que será potencialmente atacado e incluye todo el código fuente y el ejecutable. Una peculiaridad de las pruebas de seguridad en esta etapa es que es posible, para los que ejecutarán las pruebas de seguridad, determinar si las vulnerabilidades pueden ser explotadas y exponer la aplicación a riesgos reales. Estos incluyen vulnerabilidades comunes de aplicaciones Web, así como problemas de seguridad que han sido identificados antes con otras actividades como el modelado de amenazas, análisis de código fuente y revisiones de seguridad en código.

Por lo general, los ingenieros de pruebas, en lugar de los programadores de software, efectúan la validación de seguridad, cuando la aplicación está en la etapa de pruebas de integración. Los ingenieros de pruebas de seguridad, tienen conocimientos de las vulnerabilidades y su explotación en las aplicaciones Web y se apoyan fuertemente del asesor de seguridad, en técnicas de pruebas de seguridad y de la propia validación de los requerimientos de seguridad. Para la realización de tales pruebas, es un requisito previo que los casos de prueba de seguridad se documenten, para que funcionen como directrices de los procedimientos.

6.3.1 Pruebas de penetración (pruebas de caja negra)

Las pruebas de penetración, comúnmente llamadas hacking ético, se ejecutan en una etapa tardía del ciclo de desarrollo, estas pruebas se realizan al tener completo un incremento o inclusive todo el proyecto, y son guiadas por los casos de abuso, los cuales en esta tesis fueron definidos como las rebanadas de seguridad en los Casos de Uso 2.0, sin embargo, la creatividad y habilidad del probador puede llevar a detectar más fallos de los contemplados. Estas pruebas permiten conoce el comportamiento del sistema bajo un ataque, probándolo directamente en su ambiente de ejecución, razón por la cual el probador tiene un amplia libertad para desarrollar los vectores de ataque que considere apropiados.

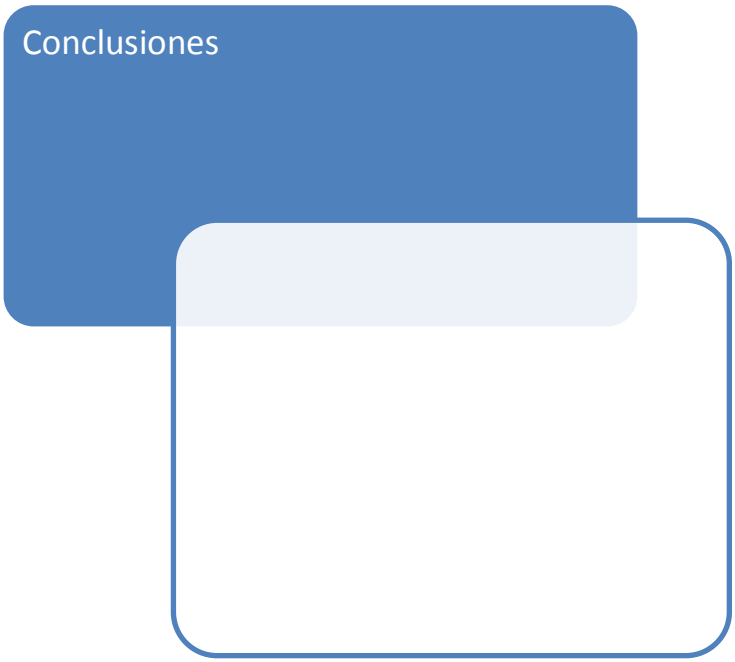
Lo ideal de las pruebas de penetración es ejecutarlas en un entorno lo más parecido al de producción, pero sin tratarse de este, o sin compartir ambiente con sistemas productivos, ya que la misma naturaleza e incertidumbre de los resultados, pueden corromper la información, comprometer activos o afectar la estabilidad de los ambientes involucrados.

En un principio estas pruebas buscan mostrar el alcance que lograría un verdadero atacante en condiciones reales, por lo cual se requiere de un probador altamente creativo

y capacitado en el área de seguridad, que trabaje conjuntamente con el asesor de seguridad para definir y ejecutar la mayor cantidad de vectores de ataque posibles, esto permitirá identificar la mayor cantidad de debilidades y asegurarlas, antes de que el sistema sea productivo.

Ya que se busca emular un escenario de ataque real, usualmente el probador cuanta con la misma información con la cual contaría un atacante en cualquier parte del mundo, usualmente se trata de una dirección IP o una URL, a partir de la cual se obtiene información para posteriormente ejecutar el ataque, sin embargo, el probador puede hacer uso de cierta información adicional que agilice el desarrollo de las pruebas y le aporte una ventaja en comparación con un atacante real.

El uso de escaners automatizados para la búsqueda de vulnerabilidades, es un mecanismo a través del cual se puede ahorrar un periodo de tiempo importante. Herramientas como Nmap, Nessus, Openvas, Acunetix o Metasploit, permiten ejecutar pruebas particulares en cada etapa del pentesting, desde el reconocimiento do hasta la explotación, sin embargo el marco de las pruebas de penetración no debe quedar en la ejecución de estas herramientas, sino que también debe realizarse una exploración manual, a través de la manipulación de parámetros o directamente del código fuente que se ejecuta en el explorador del cliente. Aunque la mayor parte de este trabajo es realizado de manera paso a paso, hay otro conjunto de herramientas que pueden apoyar en la realización de esta tarea, tal es el caso del Fuzzing, donde se pueden ingresar cadenas peculiarmente formadas, aleatorias y de longitud variable en las entradas del sistema. La técnica de Fuzzing puede develar la posibilidad de la existencia de un fallo, pero está limitada a la detección de una posibilidad de error, el identificar el fallo, confirmarlo y explotarlo, es tarea del probador.


Conclusiones

CONCLUSIONES

Los desarrollos web han cobrado un gran auge, es por ello que con mucha frecuencia, las empresas productoras de aplicaciones vean en ellas un nicho creciente y rentable de negocio, y muchas pequeñas organizaciones se involucren en la generación de este tipo de aplicaciones. Es por ello que este trabajo de tesis puede apoyarlas a generar productos de mayor valor y calidad, enfocándose en el requerimiento no funcional de la seguridad. Se agregaron al PUA, actividades, productos y un rol para ocuparse de la seguridad de forma explícita a lo largo de todo el proceso de desarrollo de aplicaciones web.

La seguridad debe ser considerada en todas las capas de un sistema. No solo en los sistemas operativos o la red. Si bien la seguridad perimetral, configuraciones, protocolos, políticas y estándares, son puntos necesarios, es igual de significativo considerar al mismo sistema e inclusive a los propios usuarios.

Las estadísticas demuestran que la mayoría de los ataques se producen en la capa de aplicación, lo que hace necesario el invertir más en la protección del software, y adoptar una estrategia de seguridad para el desarrollo, involucrando personas, procesos y tecnología, entendiendo que el software seguro es el resultado de múltiples actividades y que mejorar la seguridad implica un cambio cultural en la organización, usuarios y clientes.

Como trabajo en esta tesis, a cada flujo de trabajo del PUA, se agregaron productos, roles y actividades que enriquecen el aspecto de seguridad. Un elemento importante es la aparición del rol del "*asesor de seguridad"* quien orquesta el proyecto desde el punto de vista de este requerimiento no funcional. Así mismo se incluyen medidas y recomendaciones a considerar para la administración, el modelado, el diseño, la arquitectura, la construcción y las pruebas. Indicando en qué momento se genera cada uno de los productos y las actividades que los conciben, atendiendo el aspecto de seguridad de forma iterativa en cada bloque de software desarrollado.

Además del PUA, los casos de uso 2.0 son una herramienta eficaz y capaz de ser adaptable como modelos para incorporar aspectos de seguridad desde la toma de requerimientos, cimentando la seguridad del software para las futuras etapas de desarrollo.

El proceso que aquí se abordó, busca exponer los fallos más recurrentes a los que las aplicaciones web son susceptibles (las 10 vulnerabilidades más comunes), así como las medidas de protección que se deben de considerar, de igual forma, se busca exponer que la seguridad debe ser contemplada y abordada desde el inicio, y durante todo el proceso de desarrollo.

Un trabajo futuro es la actualización de las actividades a ejecutar, según los nuevos vectores de ataque que se presenten, la inclusión de un espectro más amplio de

vulnerabilidades a considerar o la incorporación del aspecto de seguridad en otras metodologías, así como la extensión a otros tipos de aplicaciones, considerando sus particularidades. Por ejemplo: el software desarrollado para dispositivos portátiles.

7 APÉNDICE

7.1 Contrato de software seguro

1. INTRODUCCIÓN

Este anexo está hecho para _____ ("convenio") entre el Cliente y el proveedor. Tanto el Cliente como el proveedor acuerdan maximizar la seguridad del software de acuerdo a los siguientes términos.

2. FILOSOFÍA

Este anexo pretende aclarar los derechos y obligaciones relacionados a seguridad de todas las partes en una relación de desarrollo de software. Al mayor nivel, las partes acuerdan que:

a) Las decisiones de seguridad estarán basadas en el riesgo

Las decisiones sobre la seguridad serán tomadas en conjunto por el cliente y el proveedor basándose en un firme entendimiento del riesgo involucrado.

b) Las actividades de seguridad estarán balanceadas

El esfuerzo de seguridad será distribuido equitativamente a través de todo el ciclo de desarrollo de software.

c) Las actividades de seguridad estarán integradas

Todas las actividades y documentación discutidas aquí pueden y deben ser integradas dentro del ciclo de desarrollo de software del proveedor y no mantenerse separadas del resto del proyecto. Nada en este anexo implica algún proceso de desarrollo de software en particular.

d) Se esperan vulnerabilidades

Todo software contiene defectos, y algunos de estos crearan problemas de seguridad. Ambos, cliente y proveedor se esforzarán para identificar las vulnerabilidades tan pronto como sea posible en el ciclo de desarrollo.

e) La información sobre seguridad será comunicada por completo

Toda la información relevante para la seguridad se compartirá entre el cliente y el proveedor inmediatamente y completamente.

f) Se requerirá solo documentación de seguridad necesaria

La documentación de seguridad no necesita ser demasiado amplia para describir claramente el diseño de seguridad, análisis de riesgo y problemas.

3. ACTIVIDADES EN EL CICLO DE DESARROLLO

a) Entendimiento del Riesgo

El proveedor y el cliente acuerdan trabajar juntos para entender y documentar los riesgos que enfrenta la aplicación. Este esfuerzo debe identificar los riesgos clave para los activos y funciones importantes que provee la aplicación. Cada uno de los temas listados en la sección de requerimientos debe ser considerado.

b) Requerimientos

Basado en los riesgos, el proveedor y cliente acuerda trabajar juntos para crear requerimientos de seguridad detallados como parte de la especificación del software a ser desarrollado. Cada uno de los temas listados en la sección de requerimientos de este anexo debe ser discutido y evaluados por ambos, cliente y proveedor. Estos requerimientos pueden ser satisfechos por software creado a la medida, software de terceros, o la plataforma.

c) Diseño

El desarrollador acuerda proveer documentación que explique claramente el diseño para adquirir cada uno de los requerimientos de seguridad. En muchos de los casos, esta documentación describirá los mecanismos de seguridad, donde se incluirán dentro de la arquitectura, todos los patrones de diseño relevantes para asegurar su uso adecuado. El diseño debe especificar claramente si tales mecanismos vienen de software a la medida, software de terceros o de la plataforma.

d) Implementación

El proveedor acuerda entregar y seguir un conjunto de lineamientos de codificación segura. Estos lineamientos indicarán como se le debe dar formato, estructurar y comentar el código fuente. Todo código relacionado a seguridad debe ser comentado profundamente. Guías sobre cómo evitar vulnerabilidades se seguridad comunes debe ser incluida. También, todo el código debe ser revisado por al menos otro desarrollador basándose en los requerimientos de seguridad y los lineamientos de codificación antes de ser considerado listo para pruebas unitarias.

e) Pruebas y análisis de seguridad

El proveedor acuerda entregar y seguir un plan de pruebas de seguridad que defina cómo se realizarán las pruebas o bien establecer como cada uno de los requerimientos de seguridad va a ser cumplido. El nivel de rigor de esta actividad debe ser considerado y detallado en un plan. El proveedor ejecutará el plan de prueba y proveerá los resultados al cliente.

f) Publicación segura

El proveedor acuerda entregar lineamientos de configuración segura que describan completamente todas las opciones de configuración relacionadas con seguridad y sus implicaciones para la seguridad del software en su conjunto. Los lineamientos deben incluir una descripción completa de las dependencias en la plataforma y como deben ser configuradas de manera segura, incluidas las del sistema operativo, servidor Web y servidor de aplicación. La configuración predeterminada del software debe ser segura.

4. ÁREAS CON REQUERIMIENTOS DE SEGURIDAD

Los siguientes temas/áreas deben ser considerados durante las actividades de entendimiento de riesgo y definición de requerimientos. Este esfuerzo debe producir un conjunto de requerimientos ajustados, específicos y probables. Ambos, proveedor y cliente deben ser involucrados en este proceso y deben ponerse de acuerdo sobre el conjunto final de requerimientos.

a) Validación de entradas y codificación

Los requerimientos deben especificar las reglas para canonicalización, validación y codificación de cada dato de entrada a la aplicación, ya sea de usuarios, sistemas de archivos, bases de datos o sistemas externos. La regla predeterminada debe ser que todas las entradas sean validadas a menos que cumplan con una especificación detallada de que está permitido. Además, los requerimientos deben especificar las acciones a tomar cuando se reciben entradas no válidas. Específicamente, la aplicación no debe ser susceptible a inyecciones, desbordamientos, manipulación y otros ataques con entradas de usuario corruptas.

b) Autentificación y manejo de sesiones

Los requerimientos deben especificar como se protegerán las credenciales para autentificación y los identificadores de sesión a través del ciclo de desarrollo de software. Los requerimientos para todas las funciones relacionadas deben ser agregados incluyendo recuperar contraseñas, cambio de contraseñas, recordar contraseñas, desconexión y conexión múltiple.

c) Control de Acceso

Los requerimientos deben incluir una descripción detallada de todos los roles (grupos, privilegios, autorizaciones) usadas en la aplicación. Los requerimientos deben indicar todos los activos y funciones que provee la aplicación. Los requerimientos deben especificar detallada y exactamente los derechos de acceso para cualquier activo y función de cada rol. Se sugiere utilizar un formato de matriz de control de acceso para documentar estas reglas.

d) Manejo de Errores

Los requerimientos deben detallar como se van a manejar los errores que ocurran dentro del procesamiento. Algunas aplicaciones deberían hacer lo mejor posible en caso de un error, mientras que otras deberían terminar su procesamiento inmediatamente.

e) Historial

Los requerimientos deben especificar que eventos son relevantes para la seguridad y necesitan ser registrados, como ataques detectados, intentos de conexión fallidos e intentos de exceder la autorización. Los requerimientos deben especificar también que información registrar con cada evento, incluyendo hora y fecha, descripción del evento, detalles de aplicación, y otra información útil en esfuerzos forenses.

f) Conexiones a sistemas externos

Los requerimientos deben específica como la autentificación y cifrado será manejado para todos los sistemas externos, tales como bases de datos, directorios y servicios Web. Todas las credenciales requeridas para la comunicación con sistemas externos deben almacenarse cifradas y fuera del código dentro de archivos de configuración.

g) Cifrado

Los requerimientos deben especificar qué datos deben ser cifrados, cómo serán cifrados y cómo todos los certificados y otras credenciales deben ser manejados. Las aplicaciones deben usar algoritmos estándar implementados en librerías de cifrado que hayan sido usadas y probadas ampliamente.

h) Disponibilidad

Los requerimientos deben especificar como protegerse de ataques de negación de servicio. Todos los posibles ataques en la aplicación deben ser considerados, incluyendo bloqueos de autentificación, agotamiento de conexiones y otros ataques de agotamiento de recursos.

i) Configuración segura

Los requerimientos deben especificar que los valores predeterminados para todas las configuraciones relacionadas a la seguridad deben ser seguras. Para propósitos de auditoría, el software debería ser capaz de producir un reporte sencillo de leer que muestre los detalles de todas las configuraciones relacionadas con seguridad.

j) Vulnerabilidades especificas

Los requerimientos deben incluir un conjunto de vulnerabilidades específicas que no deben estar presentes en el software. A menos que sea especificado de otra manera, el software no debe incluir ninguna de las fallas descritas en el la "Lista de OWASP sobre las 10 vulnerabilidades más críticas en aplicaciones Web".

2. PERSONAL Y ORGANIZACIÓN

a) Arquitecto en seguridad

El desarrollador asignará la responsabilidad por la seguridad a un solo recurso técnico experimentado, para ser conocido como el arquitecto de seguridad del proyecto. El arquitecto de seguridad certificará la seguridad de cada entregable.

b) Entrenamiento en Seguridad

El proveedor será responsable de verificar que todos los miembros del equipo de desarrollo han sido entrenados en técnicas de programación segura.

c) Desarrolladores dignos de confianza

El proveedor acuerda realizar las investigaciones de antecedentes, apropiadas a todos los miembros del equipo de desarrollo.

3. AMBIENTE DE DESARROLLO

a) Codificación segura

El proveedor debe mencionar que herramientas se usarán en el ambiente de desarrollo de software para promover la codificación segura.

b) Administración de configuración

El proveedor debe usar un sistema de control de versiones que autentifique y registre el miembro del equipo asociado con todos los cambios en el código base, todos los archivos de configuración y compilación.

c) Distribución

El proveedor debe usar un proceso de compilación que construya confiablemente un archivo de distribución completo desde el código fuente. Este proceso debe incluir un método para verificar la integridad del software entregado al cliente.

4. LIBRERIAS, MARCOS DE DESARROLLO Y PRODUCTOS

a) Apertura

El proveedor debe mencionar todas las aplicaciones de terceros usadas en el software, incluyendo todas las librerías, marcos de desarrollo, componentes y otros productos, ya sean comerciales, gratuitos, de código abierto o código cerrado.

b) Evaluación

El proveedor debe hacer esfuerzos razonables para asegurar que el software de terceros cumple con todos los términos de este contrato y es tan seguro como el código a la medida desarrollado bajo este contrato.

5. REVISIONES DE SEGURIDAD

a) Derecho de revisión

El cliente tiene el derecho a mandar a revisar el software para buscar fallas de seguridad, esto a sus expensas y en cualquier momento dentro de los 60 días después de la entrega. El desarrollador acepta proveer apoyo razonable al equipo de revisión proveyendo el código fuente y acceso a los ambientes de pruebas.

b) Cobertura de la revisión

Las revisiones de seguridad deben cubrir todos los aspectos del software entregado, incluyendo código a la medida, componentes, productos y configuración de sistema.

c) Alcance de la revisión

Al menos, la revisión debe cubrir todos los requerimientos de seguridad y debería buscar otras vulnerabilidades comunes. La revisión puede incluir una combinación de búsqueda automatizada de vulnerabilidades, pruebas de intrusión, análisis estático de código fuente y revisión de código por expertos.

d) Problemas encontrados

Los problemas de seguridad descubiertos serán reportados al cliente y proveedor por igual. A todos los problemas se les dará seguimiento y serán reparados como se especifique en la sección de seguimiento de problemas de seguridad de este anexo.

6. ADMINISTRACIÓN DE PROBLEMAS DE SEGURIDAD

a) Identificación

El proveedor dará seguimiento a todos los problemas de seguridad descubiertos en todo el ciclo de desarrollo, ya sea un problema en los requerimientos, diseño, implementación, pruebas, publicación u operación. El riesgo asociado con cada problema de seguridad será evaluado, documentado y reportado el cliente tan pronto como sea posible.

b) Protección

El proveedor protegerá apropiadamente la información relacionada a problemas de seguridad y la documentación relacionada a ellos, esto, para ayudar a disminuir la probabilidad de las vulnerabilidades en el software operacional del cliente sean expuestas.

c) Reparación

Los problemas de seguridad que sean identificados antes de la entrega deben ser reparados por el proveedor. Los problemas de seguridad descubiertos después de la entrega deben ser manejados en la misma manera que otros problemas funcionales según lo especificado en este contrato.

7. CERTEZA

a) Certeza

El proveedor proveerá un "paquete de certificación" consistente en la documentación de seguridad creada a través del proceso de desarrollo. El paquete deberá establecer que los requerimientos de seguridad, diseño, implementación y resultados de pruebas fueron completados apropiadamente y todos los problemas de seguridad fueron resueltos apropiadamente.

b) Auto-Certificación

El arquitecto de seguridad certificará que el software cumple con los requerimientos de seguridad, que todas las actividades de seguridad se realizaron, y que todos los problemas de seguridad identificados han sido documentados y resueltos. Cualquier excepción al estado de la certificación debe estar totalmente documentada al momento de la entrega.

c) No código malicioso

El proveedor garantiza que el software no debe contener ningún código que no se alinee a algún requerimiento del software y debilite la seguridad de la aplicación, incluyendo virus, gusanos, bombas de tiempo, puertas traseras, caballos de Troya y cualquier otra forma de código malicioso.

8. CONCENTIMIENTO Y MANTENIMIENTO DE LA SEGURIDAD

a) Consentimiento

El software no debe ser considerado como aceptado hasta que el paquete de certificación este completo y todos los problemas de seguridad han sido resueltos.

b) Investigación de problemas de seguridad

Después del consentimiento, si se sospecha (razonablemente) o descubren problemas de seguridad, el proveedor deberá asistir al cliente en realizar una investigación para determinar la naturaleza del problema. El problema debe ser considerado "nuevo" si no es cubierto por los requerirnos de seguridad y está fuera del alcance (razonable) de las pruebas de seguridad.

c) Problemas de seguridad nuevos

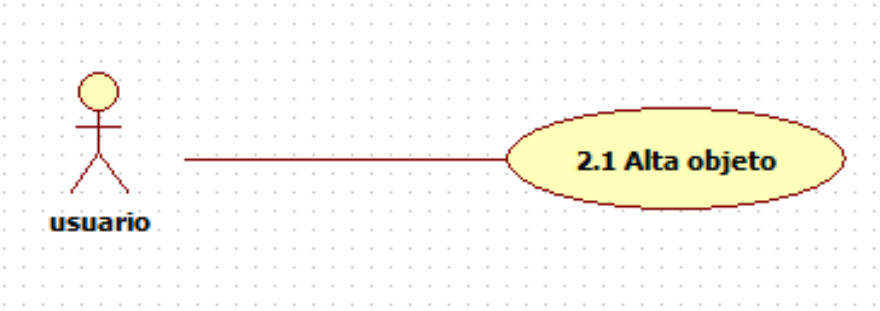
El proveedor y el cliente consientes en medir el alcance y esfuerzo requerido para resolver problemas de seguridad nuevos, y negociar de buena fe para alcanzar un acuerdo y realizar el trabajo requerido para solucionarlos.

d) Otros problemas de seguridad

El proveedor debe usar todos los esfuerzos comercialmente razonables consistentes con prácticas de desarrollo comunes en la industria, tomando en cuenta la severidad del riesgo, para resolver todos los problemas de seguridad considerados nuevos tan pronto como sea posible.

7.2 Casos de uso genéricos

7.2.1 Caso de uso 2.1: Alta objeto.



Descripción: el usuario registra un nuevo elemento dentro del sistema, el cual puede ser otro usuario, alguna mercancía, venta, elemento o información, para ello llena los campos que solicitan la descripción del objeto, identificador, nombre y cualquier otro dato que resulte relevante. Esta información es almacenada en una base de datos para su futura referencia.

Nivel: el nivel de este ciclo es alto ya que es una precondición de las acciones de buscar, consultar y dar de baja algún objeto.

Actor principal: usuario

Inicio del caso de uso: cuando el usuario requiere dar de alta un nuevo objeto al sistema.

Precondiciones: el usuario debe haber iniciado sesión en el sistema.

Post-condiciones: al hacer la consulta o búsqueda de objetos, se puede constatar que el nuevo elemento ya se encuentra disponible.

Excepciones no controladas: no se puede registrar el objeto debido a una falla en la conexión con la base de datos.

Consecuencias de fallo: de existir un fallo al momento de registrar un nuevo objeto, los cambios no se registran en la base de datos y el inventario permanece sin alteraciones.

Garantías mínimas: se garantiza que de no poder dar de alta a un nuevo objeto, esto no afectará el desempeño del sistema, por lo que basta con repetir el procedimiento para poder concluirlo, la disponibilidad del servicio se mantiene y no se modifica ningún tipo de información que no deba ser alterada (integridad).

Frecuencia: alta, debido a que es una precondición de varios casos de uso posteriores.

Supuestos: se asume que el usuario ya inicio sesión y tiene los privilegios suficientes para dar de alta nuevos objetos en el sistema.

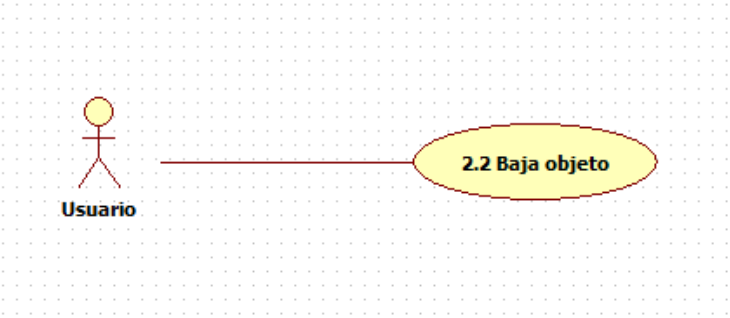
Activos relacionados:

- Información almacenada en la base de datos y su estructura.
- Información sobre la configuración del servidor de base de datos.

Amenazas relacionadas:

a) Inyección de código
b) Falla de Restricción de Acceso a URL
c) Redirecciones y Reenvíos no validados
d) Cross Site Scripting XSS
e) Pérdida de Autenticación y Gestión de Sesiones

7.2.2 Caso de uso 2.2: Baja de objeto.



Descripción: El usuario da de baja un objeto dentro del sistema, el cual puede ser otro usuario, alguna mercancía, venta, elemento o información, para ello selecciona el o los elementos que desea dar de baja y ejecuta la acción, una confirmación se le solicitará antes de hacer efectivo el cambio en la base de datos.

Nivel: el nivel de este caso de uso es baja ya que no es para otra funcionalidad.

Actor principal: usuario

Inicio del caso de uso: cuando el usuario requiere dar de baja un conjunto de objetos del sistema.

Precondiciones: el usuario debe haber iniciado sesión en el sistema y deben existir objetos dentro del mismo.

Post-condiciones: al ir a la consulta o búsqueda de objetos, se puede constatar que el elemento ya no se encuentra disponible.

Excepciones no controladas: No se puede dar de baja el objeto debido a una falla en la conexión con la base de datos.

Consecuencias de fallo: de existir un fallo al momento de dar de baja un objeto, los cambios no se registran en la base de datos y el inventario permanece sin alteraciones.

Garantías mínimas: se garantiza que de no poder dar de baja a un objeto, esto no afectará el desempeño del sistema, por lo que basta con repetir el procedimiento para poder concluirlo, la disponibilidad del servicio se mantiene y no se modifica ningún tipo de información que no deba ser alterada (integridad).

Frecuencia: alta, debido a que no es ninguna precondición de algún otro caso de uso posterior, pero forma parte de las transacciones frecuentes del negocio.

Supuestos: se asume que el usuario ya inicio sesión y tiene los privilegios suficientes para dar de baja objetos del sistema, además de que existen objetos previamente registrados.

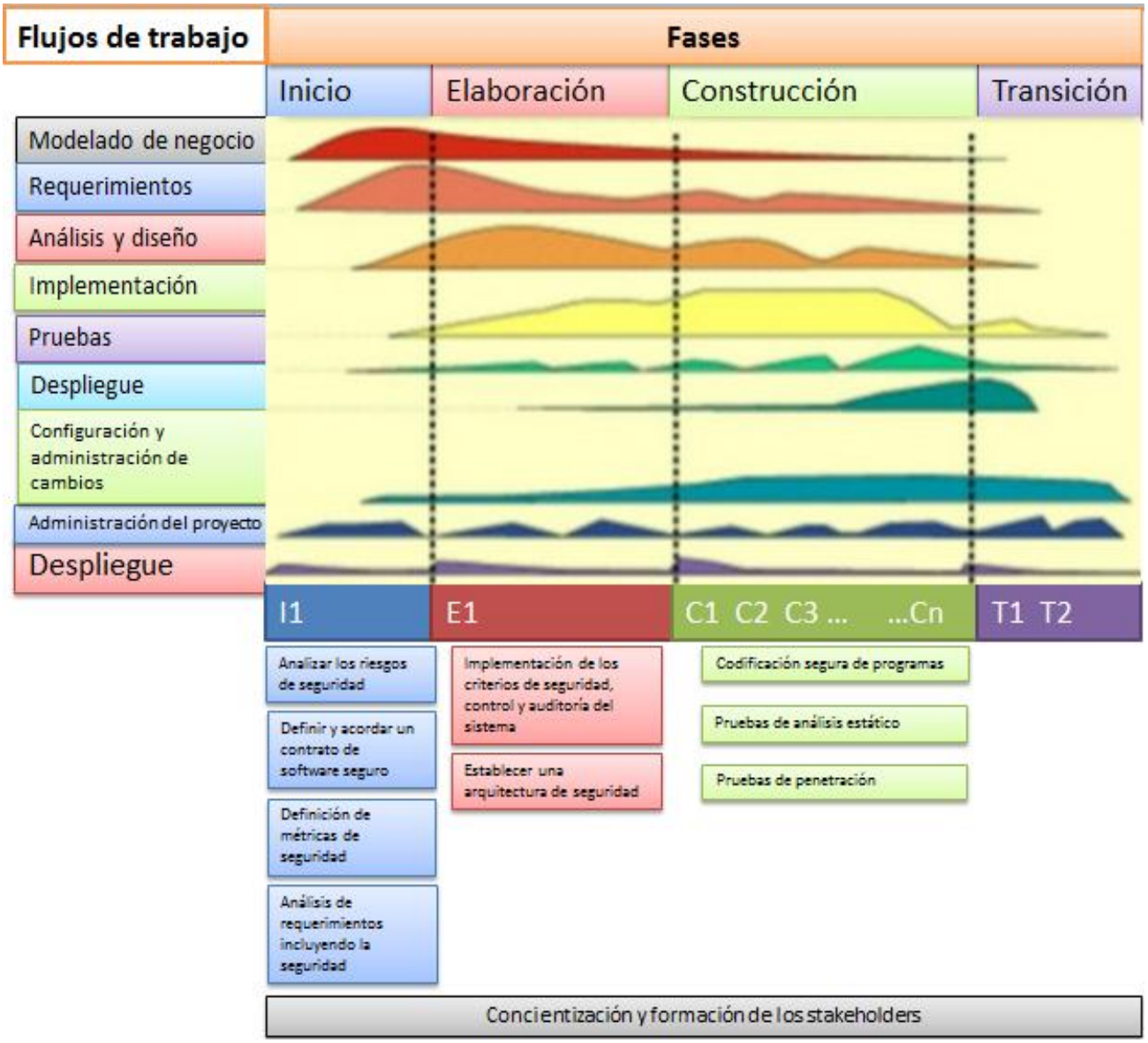
Activos relacionados:

- Información almacenada en la base de datos y su estructura.
- Información sobre la configuración del servidor de base de datos.

Amenazas relacionadas:

a) Inyección de código
b) Falla de Restricción de Acceso a URL
c) Pérdida de Autenticación y Gestión de Sesiones
d) Falsificación de Peticiones en Sitios Cruzados (CSRF)

7.3 Proceso Unificado Ágil incorporando actividades de seguridad



BIBLIOGRAFÍA Y MESOGRAFÍA

1. Web Application Security Consortium. [En línea] 2005. [Citado el: 05 de Julio de 2012.] http://www.webappsec.org/.
2. **Lopez Hernandez Ardieta, Jorge.** *Ingeniería de Software Seguro.* [Artículo] Madrid España : Indra company, 2012.
3. **Asociación Española de Normalización y Certificación.** *CEI/IEC 61508-6: Seguridad funcional de los sistemas eléctricos/electrónicos/electrónicos programables relacionados con la seguridad. Parte 6, Directrices para la aplicación de las normas CEI 61508-2 y CEI 61508-3.* s.l. : AENOR, 2003.
4. **Sommerville, Ian.** *Ingeniería del software.* s.l. : Pearson Educación, 2005. ISBN 9788478290741.
5. El portal de ISO 27001. [En línea] ISO/IEC, 2005. [Citado el: 17 de Enero de 2012.] http://www.iso27000.es.
6. SSE-CMM. *Systems Security Engineering Capability Maturity Model.* [En línea] ISSEA, 2003. http://www.sse-cmm.org.
7. *Desarrollo de productos de software seguros en sintonía con los modelos SSE-CMM, COBIT e ITIL.* **Tovar, Edmundo, y otros, y otros.** 2, Madrid, España : Revista Procesos y Métricas - AEMES, 2006, Vol. 3. ISSN: 1698-2029.
8. **McGraw, Gary.** *Software Security: Building Security In.* s.l. : Addison-Wesley Professional, 2006. ISBN-13: 978-0-321-35670-3.
9. **Ramos Alvarez, Benjamín y Ribagorda Garnacho, Arturo.** *Avances en criptología y seguridad de la información.* Madrid, España : Ediciones Díaz de Santos, 2004. ISBN:9788479786502.
10. **Areitio Bertolín, Javier.** *Seguridad de la información. Redes, Informática y Sistemas de información.* Madrid, España : Editorial Paraninfo, 2008. ISBN:9788497325028.
11. **Desongles Corrales, Juan y Moya Arribas, Marcial.** *Conocimientos Basicos de Informatica Ebook.* Sevilla España : MAD-Eduforma, 2006. ISBN:9788466567206.
12. **Real Academia Española.** Diccionario de la Real Academia Española. [En línea] Real Academia Española. [Citado el: 20 de Noviembre de 2011.] http://www.rae.es.
13. **Universidad Nacional de Luján.** Departamento de Seguridad Informática de la Universidad Nacional de Luján. [En línea] Universidad Nacional de Luján. [Citado el: 10 de Enero de 2012.] http://www.seguridadinformatica.unlu.edu.ar.
14. **Garcia-Cerevignon Hurtado, Alfonso y Alegre, Maria del Pilar.** *Seguridad informática.* Madrid España : Editorial Paraninfo, 2011. ISBN:9788497328128.
15. **Facultad de Ingeniería UNAM.** Tutorial de seguridad informática. [En línea] Facultad de Ingeniería UNAM. [Citado el: 20 de Febrero de 2012.] http://redyseguridad.fi-p.unam.mx/proyectos/tsi/index.html.
16. **Sánchez Garreta, José Salvador, Chalmeta Rosalen, Ricardo y Colltell Símon, Óscar.** *Ingeniería de Proyectos Informáticos: Actividades y Procedimientos.* Castellón de la Plana España : Universitat Jaume I, 2003. ISBN:9788480214087.

17. **Corrales Hermoso, Alberto Luís, Beltrán Pardo, Marta y Guzmán Sacristán, Antonio.** *Diseño e implantación de arquitecturas informáticas seguras: Una aproximación práctica.* Madrid España : Librería-Editorial Dykinson, 2006. ISBN:9788497728843.
18. **M. Stair, Ralph y Walter Reynolds, George.** *Principios de Sistemas de Informacion: Enfoque Administrativo.* s.l. : Cengage Learning Editores, 2000. ISBN: 9789687529974.
19. **Villacorta Michelena, Alberto.** *Enredados. El mundo de la Internet.* Lima Perú : Alberto Villacorta. ISBN: 9789972251900.
20. **Hackers, Bandidos y.** *Patiño Builes,Albeiro.* Medellín Colombia : Universidad de Antioquia, 2007. ISBN: 9789587140286.
21. **Corrales Desongles, Juan, Ponce Cifredo, Eduardo Antonio y Grazón Villar, Ma. Luisa.** *Tecnicos de Soporte Informatico.* Sevilla España : MAD-Eduforma, 2005. ISBN:9788466551045.
22. **OWASP.** The Open Web Application Security Project (OWASP). [En línea] OWASP, 20 de Julio de 2012. [Citado el: 18 de Noviembre de 2011.] http://www.owasp.org/.
23. *Clasificación de empresas.* Ciudad de México : Diario Oficial de la Federación, 2009.
24. **Instituto Pyme.** Instituto Pyme. [En línea] Consejo Mexicano para el Desarrollo Economico y Social, A.C. y SME Institute, 2010. [Citado el: 20 de Febrero de 2012.] http://www.institutopyme.org.
25. **Rodríguez Valencia, Joaquin.** *Administración de Pequeñas y medianas empresas.* 5. s.l. : Cengage Learning Editores, 2002. ISBN: 9706862420, 9789706862426.
26. **SECRETARÍA DE ECONOMÍA, MÉXICO .** *SECRETARÍA DE ECONOMÍA, MÉXICO .* [En línea] SECRETARÍA DE ECONOMÍA, MÉXICO , 2010. [Citado el: 20 de Febrero de 2012.] http://www.economia.gob.mx/.
27. **Hernández R., Regalado y otros.** *Las MIPYMES en Latinoamérica. Estudios e Investigaciones en la Organización Latinoamericana de Administración.* [ed.] Juan Carlos Martínez Coll. 2007. ISBN: 9788469065860.
28. **Ambler, Scott W.** The Agile Unified Process (AUP). [En línea] Ambysoft, 2005. [Citado el: 26 de Octubre de 2011.] http://www.ambysoft.com/unifiedprocess/agileUP.html.
29. **Stober, Thomas y Hansmann, Uwe.** *Agile Software Development: Best Practices for Large Software Development Projects.* Alemania : Springer, 2009. ISBN: 9783540708308.
30. **Núñez Mori, José Germán.** Tesis de maestria en Ingeniería de software, Usabilidad en metodologías ágiles. *Usabilidad en metodologías ágiles.* Madrid España : Universidad Politecnica De Madrid, 2010.
31. **Jacobson, Ivar, Spence, Ian y Bittner, Kurt.** *USE-CASE 2.0. The Guide to Succeeding with Use Cases.* s.l. : IvarJacobson International, 2011.
32. **Charles, Cobb G.** *Making Sense of Agile Project Management: Balancing Control and Agility.* Canadá : John Wiley & Sons, 2011. ISBN: 9780470943366.
33. **Software Guru.** Software Guru. [En línea] [Citado el: 30 de Enero de 2013.] http://sg.com.mx/content/view/1071.