

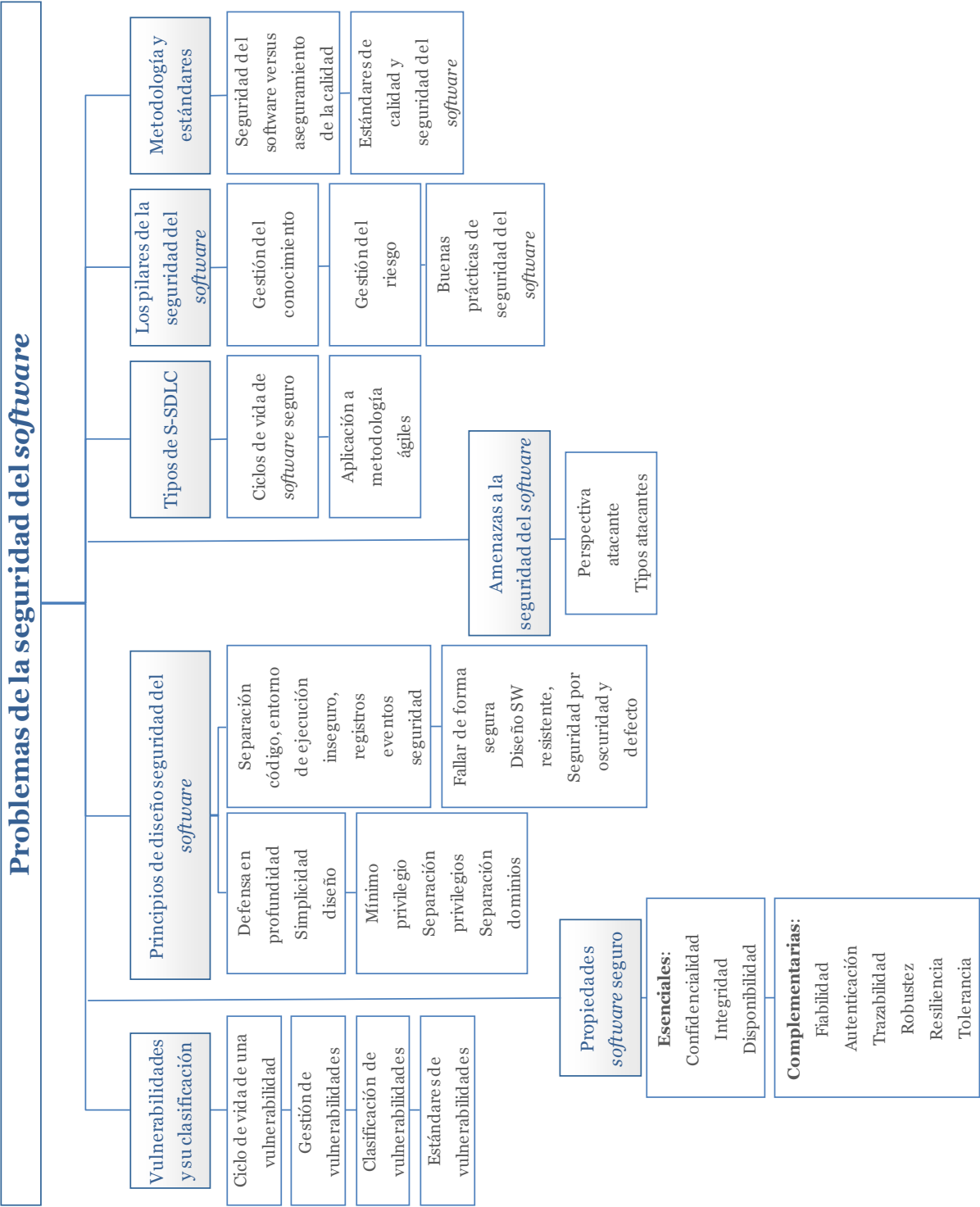
El problema de la seguridad en el *software*

- [1.1] ¿Cómo estudiar este tema?
- [1.2] Introducción al problema de la seguridad en el *software*
- [1.3] Vulnerabilidades y su clasificación
- [1.4] Propiedades *software* seguro
- [1.5] Principios de diseño seguridad del *software*
- [1.6] Amenazas a la seguridad del *software*
- [1.7] Tipos de S-SDLC
- [1.8] Los pilares de la seguridad del *software*
- [1.9] Metodologías y estándares

1

T E M A

Esquema



Ideas clave

1.1. ¿Cómo estudiar este tema?

Para estudiar este tema debes leer las ideas clave y de los apuntes elaborados por el profesor «Tema 1: El problema de la seguridad en el *software*».

El **objetivo** del presente tema es introducir al alumno en los principales conceptos que abarca la **seguridad del *software***, en cuanto a los **beneficios que produce**, su importancia en la seguridad global de un sistema, las **propiedades de un *software* seguro**, sus **principios de diseño**, los ataques a los que se tiene que enfrentar y los **estándares de seguridad aplicables a los procesos de desarrollo** seguro de *software*.

1.2. Introducción al problema de la seguridad en el *software*

La sociedad está cada vez más vinculada al ciberespacio, un elemento importante del mismo lo constituye el *software* o las aplicaciones que proporcionan los servicios, utilidades y funcionalidades. **Sin embargo estas aplicaciones presentan defectos, vulnerabilidades o configuraciones inseguras que pueden ser explotadas por atacantes de diversa índole desde aficionados hasta organizaciones de cibercrimen o incluso estados en acciones de ciberguerra**, utilizándolas como plataformas de ataque comprometiendo los sistemas y redes de la organización.

Las aplicaciones son amenazadas y atacadas, no solo en su fase de operación, sino que también en todas las fases de su ciclo de vida:

- » Desarrollo.
- » Distribución e instalación.
- » Operación.
- » Mantenimiento o sostenimiento.

Actualmente las **tecnologías de seguridad red pueden ayudar a aliviar los ciberataques, pero no resuelven el problema de seguridad real** ya que una vez que el atacante consigue vencer esas defensas, por ingeniería social por ejemplo, y comprometer una máquina del interior a través de la misma podrá atacar las demás empezando por las más vulnerables. Se **hace necesario por tanto el disponer de software seguro y confiable** que funcione en un entorno agresivo y malicioso.

En respuesta a lo expuesto anteriormente nace la **seguridad del software**, que se define como: «El conjunto de técnicas y buenas prácticas para detectar, prevenir y corregir los defectos de seguridad en el desarrollo y adquisición de aplicaciones *software* de confianza y robusto frente ataques maliciosos, libre de vulnerabilidades, ya sean intencionalmente diseñadas o accidentalmente insertadas durante su ciclo de vida, de forma que se asegure su integridad, disponibilidad y confidencialidad».

Un aspecto importante de la seguridad del *software* **es la confianza y garantía de funcionamiento conforme a su especificación y diseño y de que es lo suficientemente robusto para soportar las amenazas** que puedan comprometer su funcionamiento esperado en su entorno de operación.

Para conseguir lo anterior y **minimizar al máximo los ataques en la capa de aplicación y por tanto en número de vulnerabilidades explotables**, es necesario el **incluir la seguridad desde el principio en el ciclo de vida de desarrollo del software (SDLC)**, incluyendo requisitos, casos de abuso, análisis de riesgo, análisis de código, pruebas de penetración dinámicas, etc., en este sentido es importante el **aprovechamiento de las buenas prácticas de ingeniería de software ya existentes**.

Un beneficio importante que se obtendría de incluir un **proceso sistemático que aborde la seguridad desde las primeras etapas del SDLC, sería la reducción de los costes** de corrección de errores y vulnerabilidades.

1.3. Vulnerabilidades y su clasificación

Una **vulnerabilidad** es un fallo de programación, configuración o diseño que permite, de alguna manera, a los atacantes alterar el comportamiento normal de un programa y realizar algo malicioso como alterar información sensible, interrumpir o destruir una aplicación o tomar su control.

Se puede decir que son un subconjunto del fenómeno más grande que constituyen los *bugs* de *software*. Sus fuentes se deben:

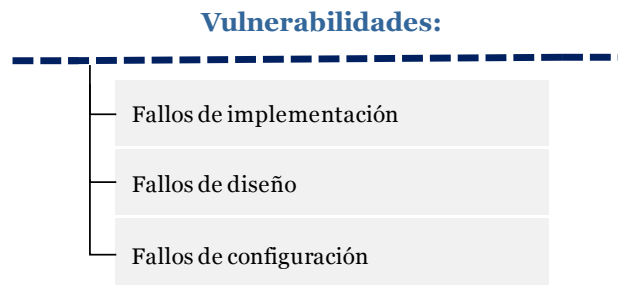


Figura 1. Tipos de vulnerabilidades del *software*

El **ciclo de vida de una vulnerabilidad** consta de las siguientes fases:

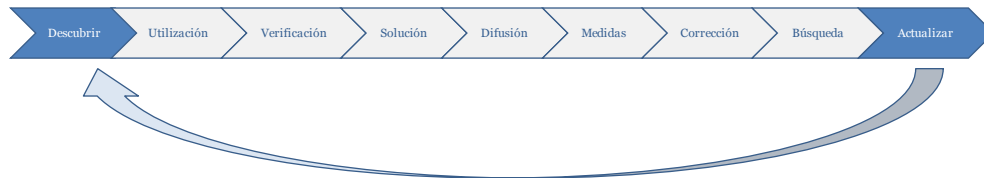


Figura 2. Ciclo de vida de una vulnerabilidad

Common Vulnerabilities and Exposures (CVE)

Es un diccionario o catálogo público de vulnerabilidades, administrado por MITRE, que normaliza su descripción y las organiza desde diferentes tipos de vista (vulnerabilidades web, de diseño, implementación, etc.).

Cada identificador CVE incluye:

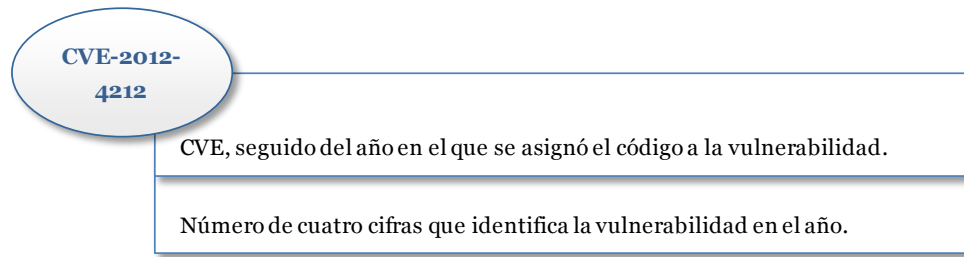


Figura 3. Identificador CVE

Common Weakness Enumeration (CWE)

Estándar internacional y de libre de uso que ofrece un conjunto unificado de debilidades o defectos de *software* medibles, que permite un análisis, descripción, selección y uso de herramientas de auditoría de seguridad de *software* y servicios que pueden encontrar debilidades en el código fuente y sistemas, así como una mejor comprensión y gestión de los puntos débiles de un *software* relacionados con la arquitectura y el diseño.

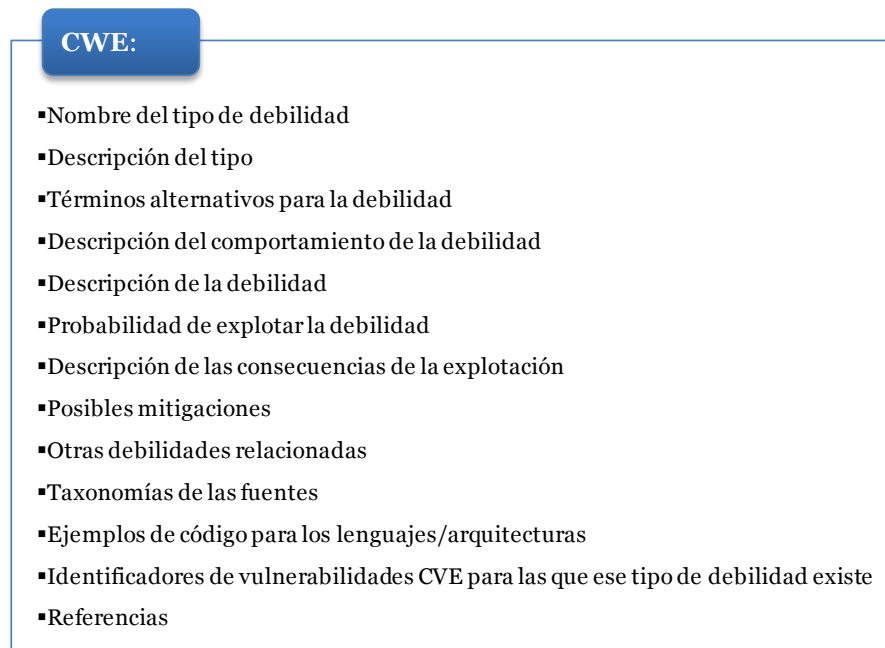


Figura 4. Campos de información de cada entrada CWE

1.4. Propiedades *software* seguro

Básicamente se tienen dos conjuntos de propiedades que definen a un *software* seguro, del que no lo es, las primeras son las **esenciales** (en amarillo), comunes a la seguridad de cualquier sistema, cuya ausencia afecta gravemente a la seguridad de una aplicación y un segundo conjunto, **complementarias** (en azul) a las anteriores que no influyen en su seguridad, pero que ayudan a mejorarla en un alto grado.



Figura 5. Propiedades seguridad del *software*

1.5. Principios de diseño seguridad del *software*

La adopción de estos principios de diseño constituye una **base fundamental de las técnicas de programación segura**, tanto para la protección de aplicaciones Web, normalmente más expuestas a los ciberataques al estar desplegadas masivamente en Internet, como otras más tradicionales del tipo cliente-servidor.

Las principales prácticas y principios de diseño tener en cuenta serían:

Objetivos Principios de seguridad	
Principio	Objetivo
Defensa en profundidad	Introducir múltiples capas de seguridad para reducir la probabilidad de compromiso del sistema.
Simplicidad del diseño	Reducir la complejidad del diseño para minimizar el número de vulnerabilidades explotables por el atacante y debilidades en el sistema.
Mínimo privilegio	Lo que no está expresamente permitido está prohibido.
Separación de privilegios	Asignación a las diferentes entidades de un rol que implique el acceso a un subconjunto de funciones o tareas y a los datos necesarios.
Separación de dominios	Minimizar la probabilidad de que actores maliciosos obtengan fácilmente acceso a las ubicaciones de memoria u objetos de datos en el sistema.
Separación código, ejecutables y datos configuración y programa	Reducir la probabilidad de que un ciberatacante que haya accedido a los datos del programa fácilmente pueda localizar y acceder a los archivos ejecutables y datos de configuración del programa.
Entorno de producción o ejecución inseguro	Evitar vulnerabilidades aplicando una serie de principios de validación de las entradas.
Registro de eventos de seguridad	Generar eventos (<i>logs</i>) de seguridad, para garantizar las acciones realizadas por un ciberatacante se observan y registran.
Fallar de forma segura	Reducir la probabilidad de que un fallo en el software, pueda saltarse los mecanismos de seguridad de la aplicación, dejándolo en un modo de fallo inseguro vulnerable a los ataques.
Diseño de software resistente	Reducir al mínimo la cantidad de tiempo que un componente de un software defectuoso o fallido sigue siendo incapaz de protegerse de los ataques.
La seguridad por oscuridad: error	Concienciarse que la seguridad por oscuridad es un mecanismo de defensa que puede proporcionar a un atacante información para comprometer la seguridad de una aplicación.
Seguridad por defecto	Reducir la superficie de ataque de una aplicación o sistema.

Tabla 1 Objetivos Principios de seguridad

1.6. Amenazas a la seguridad del *software*

El principal objetivo de la seguridad del *software* es el de mantener sus propiedades de seguridad frente a los ataques realizados por personal maliciosos sobre sus componentes y poseer el mínimo posible de vulnerabilidades explotables. Para conseguir que el desarrollo de una aplicación posea las propiedades y principios de diseño del *software* seguro presentados en los apartados anteriores, **se necesita que el personal de diseño y desarrollo desarrollen dos perspectivas.**

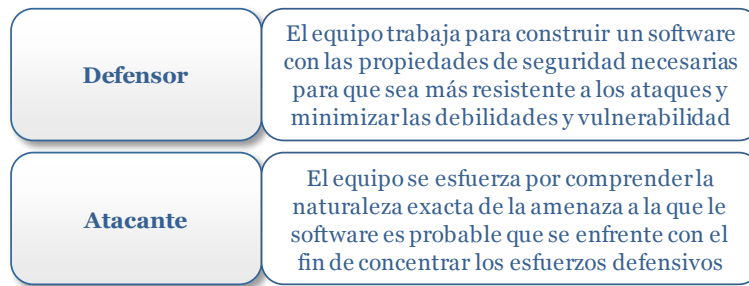


Figura 6. Perspectivas de modelado

En cuanto a los tipos de ataques o incidentes de seguridad, se presenta una taxonomía basada en **procesos y orientación al evento**, que el equipo de desarrollo y diseño podría usar para determinar qué categorías de ciberatacantes son más propensos a atacar la futura aplicación, al objeto de evaluar las futuras amenazas y riesgo a las que va estar expuesto y poder priorizar las defensas contra los métodos utilizados por estas categorías.

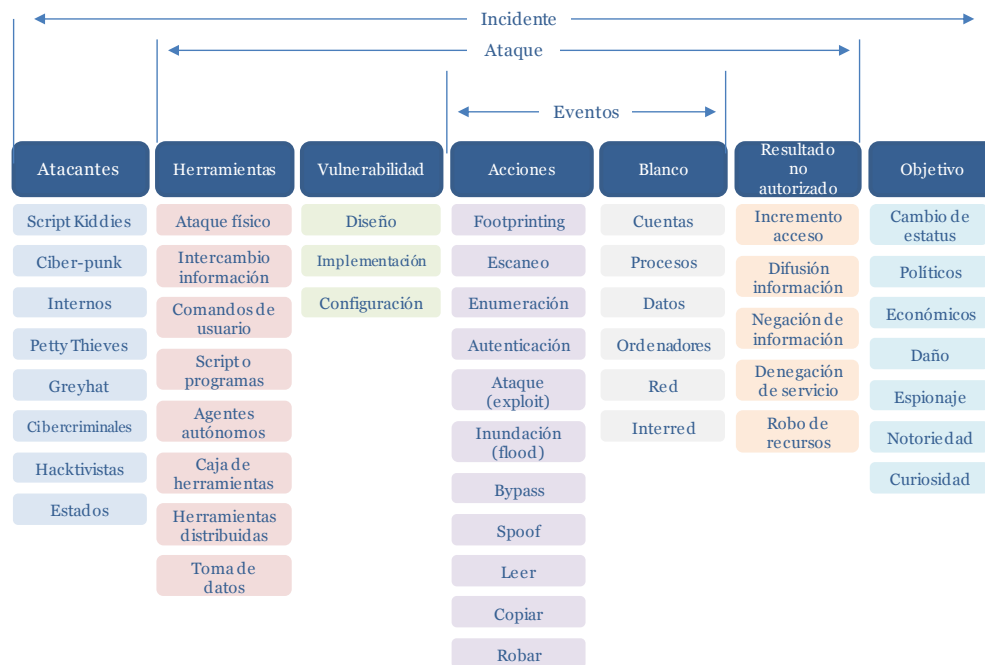


Figura 7. Taxonomía de incidentes

1.7. Tipos de S-SDLC

Las organizaciones deben **insertar buenas prácticas de seguridad en el proceso de desarrollo de *software* al objeto de obtener *software* más seguro o confiable**, bien adoptando una metodología segura de desarrollo que proporcione un marco integrado de mejora de la seguridad, como los presentados en este apartado, o a través de la evolución y mejora de sus actuales practicas de seguridad. Estas metodologías no modifican las actividades tradicionales de SDLC, insertan nuevas actividades con el objetivo de reducir el número de debilidades y vulnerabilidades en el *software*, **a este nuevo ciclo de vida con buenas prácticas de seguridad incluidas lo denominaremos S-SDLC**. Los elementos clave de un proceso de S-SDLC son:

- » Hitos de control en las fases del SDLC.
- » Principios y prácticas seguras de *software*.
- » Requisitos adecuados.
- » Arquitectura y diseño adecuados.
- » Codificación segura.
- » Integración de forma segura de los módulos y componentes del *software*.
- » Pruebas de seguridad.
- » Despliegue y distribución segura.
- » Sostenimiento seguro.
- » Herramientas de apoyo al desarrollo.
- » Gestión de configuración de sistemas y procesos.
- » Conocimientos de seguridad de los desarrolladores.
- » Gestión segura del proyecto y compromiso de la alta dirección.

Es el **modelo base** tomado para el desarrollo de este tema, McGraw propone un modelo de S-SDLC (cascada o iterativo) en el que define una serie de mejores prácticas de seguridad a aplicar a los artefactos de cada fase del desarrollo. En esencia, el proceso se centra en la incorporación de las siguientes prácticas ordenadas por orden de importancia:

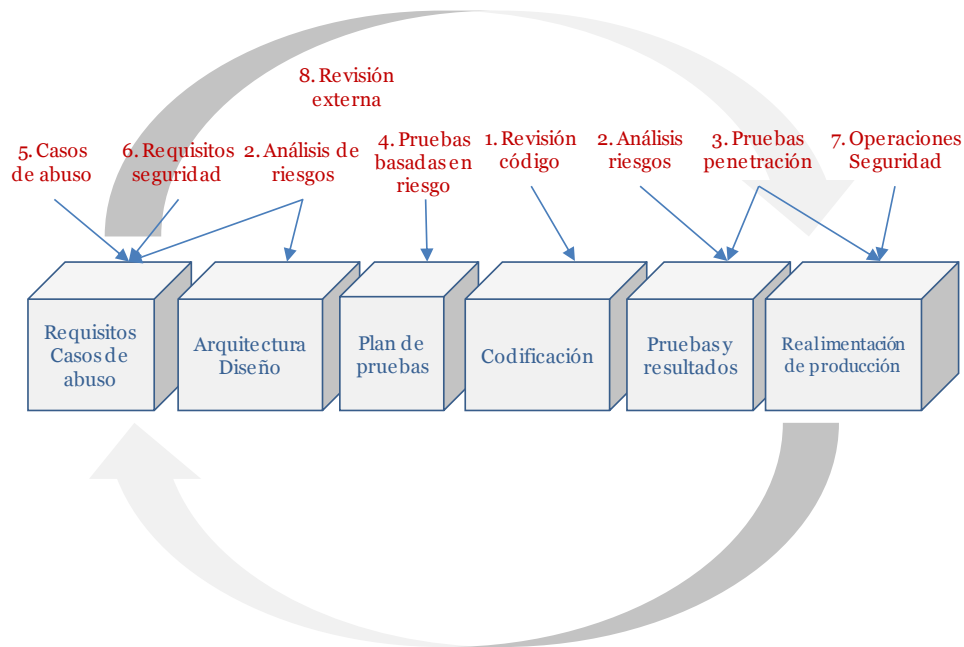


Figura 8. McGraw's Seven Touchpoints

Otras metodologías:

- » Microsoft Trustworthy Computing SDL.
- » CLASP: Comprehensive Lightweight Application Security Process.
- » Team Software Process (TSP).
- » Oracle Software Security Assurance.
- » Appropriate and Effective Guidance in Information Security (AEGIS).
- » Rational Unified Process-Secure (RUPSec).
- » Secure Software Development Model (SSDM).
- » Waterfall-Based Software Security Engineering Process Model.

1.8. Los pilares de la seguridad del *software*

Gary McGraw en su libro *Software Security: Building Security In* propone un método o proceso para ayudar a solucionar el problema de la seguridad del *software* en las **organizaciones que realizan desarrollos**, que requiere un cambio cultural de sus desarrolladores en lo que respecta sobre todo en lo relativo a **NO anteponer la seguridad respecto de la funcionalidad**, independiente del modelo o tipo del ciclo de vida del *software* que se esté utilizando y basado en tres pilares fundamentales:

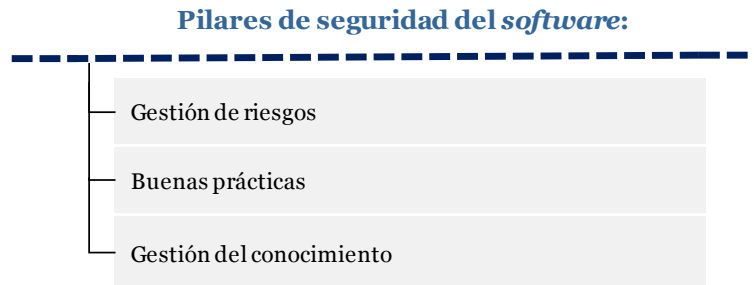


Figura 9. Pilares de seguridad de *software*

Se necesita un **proceso continuo de gestión de riesgos que abarque todo el ciclo de vida del *software***, que permite a una organización el identificar, clasificar, comprender y realizar un seguimiento de los riesgos en el tiempo en el que un proyecto *software* se desarrolla.

En la figura siguiente se puede ver un diagrama de dicho marco.

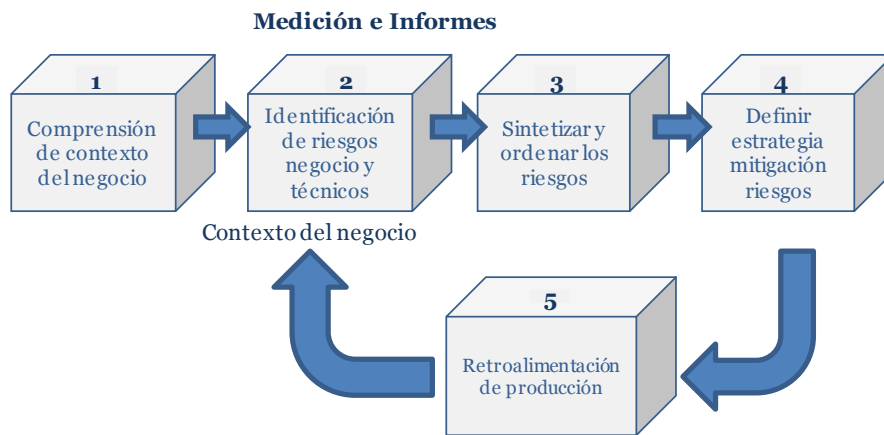


Figura 10. Marco gestión de riesgos

Un aspecto importante de la seguridad del *software*, es la adopción por los gerentes de un enfoque sistémico para **incorporar principios de buenas prácticas de seguridad del *software* «touchpoint» en su ciclo de vida de desarrollo de *software***, para lograr el doble objetivo de producir sistemas con *software* más seguro y poder verificar su seguridad, a este nuevo ciclo de vida le denominaremos ciclo de vida del *software* seguro S-SDLC.

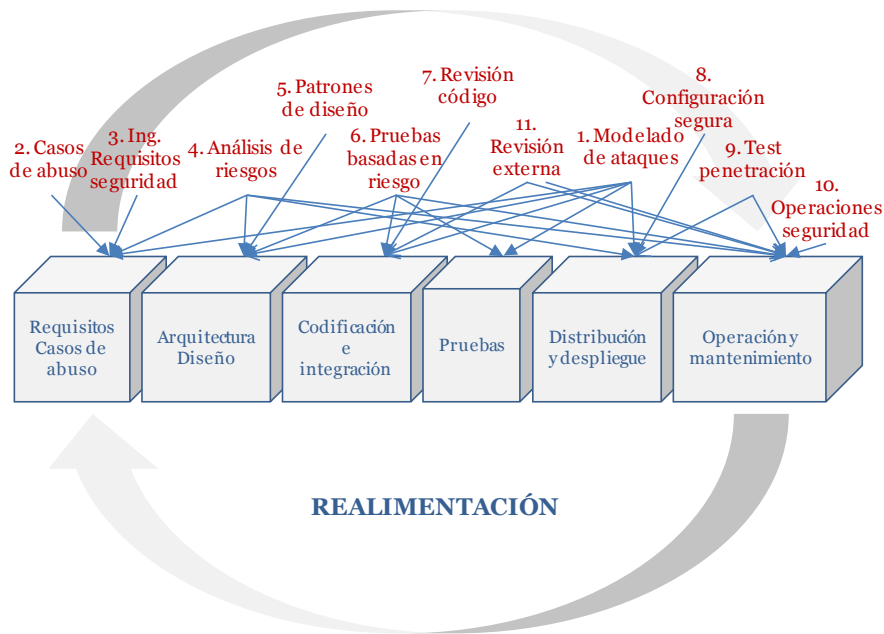


Figura 11. Mejores prácticas de seguridad del *software* en el SDLC

En lo relativo a la gestión del conocimiento es cíclica y su principal papel en condensar y difundir las buenas prácticas de seguridad del *software*, los principios de seguridad del diseño y disciplinas de codificación, para conseguir ***software* más seguro y confiable**.

El conocimiento aplicado a la seguridad del *software* abarca tres áreas:

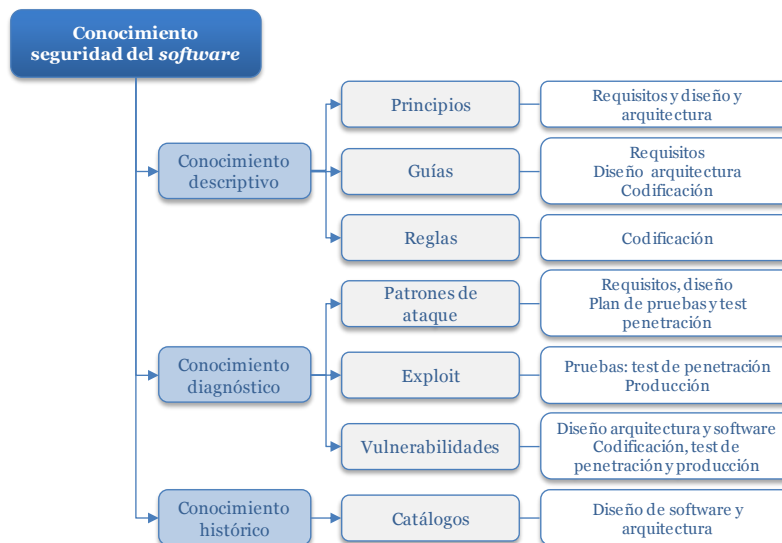


Figura 12. Catálogo del conocimiento de la seguridad del *software*

1.9. Metodologías y estándares

Las metodologías y estándares, consisten básicamente en **guías de comportamiento** específicas destinadas a aplicar una política o políticas. Su aplicación a la seguridad del *software* consiste en el desarrollo de procesos que implementen técnicas de seguridad en todas las actividades que intervienen en el ciclo de vida de desarrollo de *software*. Para conseguir lo anterior, las organizaciones deben implantar un **estándar de aseguramiento de la calidad de *software* y un estándar de seguridad**.

System Security Engineering Capability Maturity Model (SSE-CMM norma ISO/IEC 21827)

El Modelo de Capacidad y Madurez en la Ingeniería de Seguridad de Sistemas es un modelo derivado del CMM y que describe **áreas de ingeniería de seguridad** y las características esenciales de los procesos que deben existir en una organización para la mejora y evaluación de la madurez de la ingeniería de los procesos de seguridad utilizados para producir productos de **sistemas confiables y seguros**. El alcance de los procesos incluidos en la norma abarca todas las actividades del ciclo de vida del sistema e ingeniería de seguridad, incluyendo la definición conceptual, análisis de requisitos, diseño, desarrollo, integración, instalación, operación, mantenimiento y baja.

AREA DE PROCESOS	OBJETIVOS
1. Administrar los controles de seguridad	Asegurar que los controles de seguridad estén correctamente configurados y operados.
2. Evaluar el impacto	Alcanzar un entendimiento de los riesgos de seguridad asociados con la operación del sistema dentro de un ambiente operación o producción definido.
3. Evaluar riesgos de seguridad	Identificar las vulnerabilidades del sistema y determinar su potencial de explotación.
4. Valoración de la amenaza	Alcanzar un entendimiento de las amenazas a la seguridad del sistema.
5. Valoración vulnerabilidades	Alcanzar un entendimiento de las vulnerabilidades de seguridad del sistema.
6. Construir argumentos seguridad	Asegúrese de que los artefactos y procesos de trabajo claramente proporcionan la evidencia que las necesidades del cliente relativas a la seguridad se han cumplido.
7. Coordinar la seguridad	Asegúrese de que todos los miembros del equipo del proyecto son conscientes y participan en las actividades de ingeniería de seguridad en la medida que son necesarios para desempeñar sus funciones, coordinar y comunicar todas las decisiones y recomendaciones relacionadas con la seguridad.
8. Monitorización seguridad	Detección y seguimiento de los incidentes internos y externos relacionados con la seguridad, responder a los incidentes de acuerdo con la política, identificar y controlar los cambios de la política de seguridad en conformidad con los objetivos de seguridad.
9. Seguridad en la entradas	Revisión de seguridad de todas entradas del sistema con consecuencias para la seguridad, resolver los problemas de acuerdo a los objetivos de seguridad y garantizar que todos los miembros del equipo del proyecto entienden la seguridad para que puedan desempeñar sus funciones. Garantizar que la solución refleja las aportaciones de seguridad proporcionada.
10. Especificar las necesidades de seguridad	Aplicables a todas las partes, incluido el cliente, llegar a una común comprensión de las necesidades de seguridad
11. Verificar y validar la seguridad	Asegurar que las soluciones satisfacen todas sus necesidades de seguridad y operativas de los clientes de seguridad.

Tabla 2 SSE-CMM Áreas de Procesos de ingeniería seguridad sus metas y objetivos.

Material complementario

Lecciones magistrales

Tipos de ciclos de desarrollo de software seguro S-SDLC

En la presente clase, se presentan varios modelos de ciclo de vida S-SDLC con prácticas de seguridad incluidas, que se pueden aplicar independientemente del tipo de modelo de ciclo de vida, en espiral, *extreme programming*, etc. Además, algunos métodos estándar de desarrollo se ha demostrado que aumentan la probabilidad de que el *software* producido por ellos será seguro y confiable. Así mismo, debido a la popularidad de los métodos ágiles, es importante el estudiar los problemas de seguridad que surgen cuando se utilizan este tipo de ciclos de vida.



El vídeo está disponible en el aula virtual.

No dejes de leer...

CWE/SANS Top 25 Most Dangerous Software Errors

Martin, B.; Brown, M.; Paller, A. y Kirby, D. (2011). *CWE/SANS Top 25 Most Dangerous Software Errors version: 1.0.3*. MITRE Corporation.

El documento presenta una lista de los 25 errores en el *software* más dañinos, extendidos y críticos que representan vulnerabilidades fácilmente detectables y explotables que permiten al atacante tomar control de la máquina.

Accede al artículo desde el aula virtual o a través de la siguiente dirección web:

<http://cwe.mitre.org/top25/>

Building Secure Software

McGraw, G. (2003). *Building Secure Software: A Difficult But Critical Step in Protecting Your Business*. Cigital, Inc.

El documento describe una visión unificada de la gestión de riesgos de seguridad enfocada a la obtención de *software* maduro.

Accede al artículo desde el aula virtual o a través de la siguiente dirección web:

http://www.cigital.com/whitepapers/dl/Building_Secure_Software.pdf

No dejes de ver...

The Building Security In Maturity Model (BSIMM)

En este vídeo se da una visión general de una importante iniciativa de Seguridad del *Software* llamada BSIMM y su forma de uso como una herramienta de medida para su organización, proveedores y su combinación con otros métodos de medición de seguridad.



Accede al vídeo desde el aula virtual o a través de la siguiente dirección web:

<https://vimeo.com/99350519>

Security Design Reviews

La seguridad no es una tarea añadida solo al final de la fase de implantación, debe abarcar todas las fases del ciclo de vida del desarrollo de una aplicación. En este vídeo se presenta más que suficientes razones de la importancia de las evaluaciones y revisiones de seguridad del diseño y las diferentes practicas de seguridad de cada una de las fases de SDLC.



Accede al vídeo desde el aula virtual o a través de la siguiente dirección web:

<http://channel9.msdn.com/Blogs/Jossie/Security-Design-Reviews>

A fondo

Cyber Risk Report 2015

Informe del año 2015, realizado por la compañía HP Security Research, que proporciona una visión amplia del panorama de las vulnerabilidades del *software*, así como una profunda investigación y análisis de los ataques de y tendencias. Para acceder al artículo hay que registrarse.

Accede al documento desde el aula virtual o a través de la siguiente dirección web:

<http://www8.hp.com/us/en/software-solutions/cyber-risk-report-security-vulnerability/>

2011 top cyber security risks report

Informe del año 2011 que proporciona una visión amplia del panorama de las vulnerabilidades del *software*, así como una profunda investigación y análisis de los ataques y tendencias.

Accede al documento desde el aula virtual o a través de la siguiente dirección web:
<http://www.hpenterprisesecurity.com/collateral/report/2011FullYearCyberSecurityRisksReport.pdf>

Secure Coding: Building Security into the Software Development Life Cycle

Artículo de Russell L. Jones y Abhinav Rastogi que trata la inclusión de buenas prácticas de seguridad en el ciclo de vida de desarrollo del *software*.

Accede al artículo desde el aula virtual o a través de la siguiente dirección web:
<http://www.infosectoday.com/deloitte/jones.pdf>

Improving Software Security During Development

En este artículo de Robert W. Usher se exploran las bases para la creación de *software* y sistemas seguros durante su etapa de desarrollo. La seguridad del *software* se relaciona directamente con los procesos de calidad.

Accede al artículo desde el aula virtual o a través de la siguiente dirección web:
http://www.sans.org/reading_room/whitepapers/securecode/improving-software-security-development_384

Cuaderno de notas del Observatorio ¿Qué son las vulnerabilidades del *software*?

Artículo que analiza los aspectos básicos de las vulnerabilidades: por qué ocurren y cómo gestionarlas.

Accede al artículo desde el aula virtual o a través de la siguiente dirección web:

http://www.egov.ufsc.br/portal/sites/default/files/vulnerabilidades_notasobs.pdf

The Standard for Information Security Vulnerability Names

Artículo introductorio al formato CVE de gestión de vulnerabilidades de *software*.

Accede al artículo desde el aula virtual o a través de la siguiente dirección web:

<http://makingsecuritymeasurable.mitre.org/docs/cve-intro-handout.pdf>

A Community-Developed Dictionary of Software Weakness Types

Artículo introductorio al formato CWE de gestión de debilidades y vulnerabilidades del *software*.

Accede al artículo desde el aula virtual o a través de la siguiente dirección web:

<http://makingsecuritymeasurable.mitre.org/docs/cwe-intro-handout.pdf>

Practical Measurement Framework for Software Assurance and Information Security

Documento de medidas de seguridad del *software* e información que proporciona un método para medir la eficacia de las medidas de seguridad a nivel organizacional, programa o proyecto.

Accede al artículo desde el aula virtual o a través de la siguiente dirección web:

<http://www.psmc.com/Downloads/TechnologyPapers/SwA%20Measurement%2010-o8-o8.pdf>

Webgrafía

Build Security In Home

El U.S. Department of Homeland Security, DHS, desarrolla un portal de seguridad de *software*, junto con el Instituto de Ingeniería de Software de Carnegie Mellon y Cigital. Este portal ayuda a proporcionar un conjunto común, accesible, bien organizado de información de prácticas de seguridad de *software*.

El esfuerzo del portal expresamente apunta al problema, de ampliar y extender el conocimiento de seguridad de *software*.



Accede a la página web a través del aula virtual o desde la siguiente dirección:

<https://buildsecurityin.us-cert.gov/bsi/home.html>

Open Web Application Security Project (OWASP)

El Open Application Web Security Project (OWASP) es una organización a nivel mundial sin ánimo de lucro, enfocada en mejorar la seguridad del *software*. Dispone de gran cantidad de recursos como guías de diseño y pruebas, herramientas de test, *blog*, etc.



Accede a la página web a través del aula virtual o desde la siguiente dirección:

<http://www.owasp.org>

Microsoft Security Developer Center

Sitio web que describe las buenas prácticas de desarrollo seguro del modelo de S-SDLC de Microsoft y sus herramientas asociadas.



Accede a la página web a través del aula virtual o desde la siguiente dirección:

<http://msdn2.microsoft.com/en-us/security/default.aspx>

Secure Software Inc. Resources

Sitio web de la empresa HP con recursos y artículos sobre seguridad del *software*.



Accede a la página web a través del aula virtual o desde la siguiente dirección:

<http://www8.hp.com/us/en/software-solutions/software.html?compURI=1214365>

Cigital Inc. Resources

Sitio web de la empresa CIGITAL Inc. con recursos, libros, vídeos y artículos sobre seguridad del *software*.



Accede a la página web a través del aula virtual o desde la siguiente dirección:

<http://www.cigital.com/resources>

SysAdmin, Audit, Networking, and Security (SANS) Reading Room

Sitio web del Instituto SANS con gran cantidad de artículos, ver las secciones: «Application/Database Sec», «Best Practices», «Auditing & Assessment», «Malicious Code», «Scripting Tips», «Securing Code» and «Threats/Vulnerabilities».



Accede a la página web a través del aula virtual o desde la siguiente dirección:

http://www.sans.org/reading_room/

Bibliografía

Allen, J. H.; Barnum, S.; Ellison, R. J.; McGraw, G.; Mead, N. R. (2008). *Software Security Engineering: A Guide for Project Managers*. Addison Wesley Professional.

Graff, M. G. y R. van Wyk, K. (2003). *Secure Coding: Principles & Practices*. O'Reilly.

Howard, M.; LeBlanc, D. (2003). *Writing Secure Code*. Microsoft Press.

Howard, M.; Lipner, S. (2006). *The Security Development Lifecycle: SDL: A Process for Developing Demonstrably Secure Software*. Microsoft Press.

Krassowski, A. and Meunier, P. (2008). *Secure Software Engineering: Designing, Writing, and Maintaining More Secure Code*. Addison-Wesley.

McGraw, G. (2005). *Software Security: Building Security In*. Addison Wesley Professional.

Redwine, S. T. Jr. (Editor). (2006). *Software Assurance: A Guide to the Common Body of Knowledge to Produce, Acquire, and Sustain Secure Software Version 1.1*. US Department of Homeland Security.

Actividades

Trabajo: Comparación de ciclos de vida de desarrollo de *software* seguro (S-SDLC)

Como actividad puntuable para el tema 1, te propongo seguir profundizando en los modelos S-SDLC, realizando un trabajo que contenga al menos el siguiente contenido:

- » Introducción a los S-SDLC.
- » Descripción resumida de los diferentes tipos de S-SDLC.
- » Comparación de los diferentes S-SDLC, cubriendo al menos las siguientes características:
 - Actividades de Ingeniería de Requisitos.
 - Actividades de diseño.
 - Actividades de implementación.
 - Actividades de pruebas de verificación y validación.
 - Recursos.
 - Uso por la empresa.
 - Etc.
- » Propuesta de un nuevo S-SDLC.
- » Conclusiones.

Extensión máxima de la actividad: 8 a 12 páginas, fuente Georgia 11 e interlineado 1,5.

Test

1. Indica cuál de las siguientes respuestas no es una causa de aparición de vulnerabilidades en el *software*:
 - A. No seguimiento, por los desarrolladores, de guías de normalizadas de estilo en la codificación.
 - B. Desarrollo de las aplicaciones por una asistencia técnica o entidades subcontratadas.
 - C. No realización de pruebas seguridad basadas en riesgo.
 - D. No control de la cadena de suministro del *software*, lo puede dar lugar a la introducción de código malicioso en origen.

2. Indica las respuestas NO correctas respecto al ataque a las aplicaciones durante las diferentes fases de su ciclo de vida.
 - A. Desarrollo. Un desarrollador puede alterar de forma intencionada o no el *software* bajo desarrollo.
 - B. Distribución e instalación. Ocurre cuando el instalador del *software* bastiona la plataforma en la que lo instala.
 - C. Operación. Cualquier *software* que se ejecuta en una plataforma conectada a la red tiene sus vulnerabilidades expuestas durante su funcionamiento, excepto si está protegido por dispositivos de protección de la infraestructura de red.
 - D. Mantenimiento o sostenimiento. No publicación de parches de las vulnerabilidades detectadas en el momento oportuno o incluso introducción de código malicioso por el personal de mantenimiento en las versiones actualizadas del código.

3. Las fuentes de las vulnerabilidades se deben a (indicar la incorrecta):
 - A. Fallos provenientes de la codificación de los diseños del *software* realizados.
 - B. Fallos provenientes de la cadena de distribución del *software*.
 - C. Los sistemas *hardware* o *software* contienen frecuentemente fallos de diseño que pueden ser utilizados para realizar un ataque.
 - D. La instalación de *software* por defecto implica por lo general la instalación de servicios que no se usan, pero que pueden presentar debilidades que comprometan la máquina.

4. Señalar la incorrecta. Entre las técnicas y mecanismos que se tienen para salvaguardar la integridad, tenemos por ejemplo:
- A. Identificación del modo de transmisión y procesado de los datos por la aplicación.
 - B. Uso de arquitecturas de alta disponibilidad, con diferentes tipos de redundancias.
 - C. Uso de firma digital.
 - D. Estricta gestión de sesiones
5. Señalar las respuestas correctas. Algunas de las opciones específicas de diseño del *software* que lo simplifican son:
- A. Limitar el número de estados posibles en el *software*.
 - B. Aplicación de derechos de administrador.
 - C. Diseñar los componentes del software con el conjunto mínimo de características y capacidades que se requieran para realizar sus tareas en el sistema.
 - D. Acceso a los datos necesarios que debe gestionar para llevar a cabo su función en base a una serie de roles definidos.
6. Señalar las respuestas correctas. Para conseguir que el desarrollo de una aplicación posea las propiedades y principios de diseño del *software* seguro presentados en los apartados anteriores, se necesita que el personal de diseño y desarrollo desarrollen dos perspectivas:
- A. Perspectiva administrador.
 - B. Perspectiva atacante.
 - C. Perspectiva usuario.
 - D. Perspectiva defensor.

7. Señala las respuestas correctas. Para cada principio básico en el Manifiesto Ágil su contribución, neutral o obstrucción al desarrollo de un *software* seguro.

- A. La forma más eficiente y eficaz de transmitir información a los componentes de un equipo de desarrollo es a través de la comunicación cara a cara. Neutral.
- B. Sencillez, que se define como el arte de maximizar la cantidad de trabajo no realizado, es esencial para el éxito de proyectos de *software*. Contributiva.
- C. Producción de frecuentes entregas de *software*. Idealmente, una nueva cada varias semanas. Se da preferencia a la reducción de los tiempos de entrega. Neutral.
- D. Prioridad principal es satisfacer al cliente. Esto se debe lograr de forma temprana y continua a través de entrega de *software* utilizable. Obstructiva.

8. Señalar las respuestas correctas. La información utilizada durante un proceso de evaluación de riesgos de seguridad incluye:

- A. Patrones de ataque.
- B. El tipo de vulnerabilidades que se tienen.
- C. Resultados de pruebas.
- D. La naturaleza de las amenazas al *software*.

9. Señala la práctica de seguridad a la que corresponde la siguiente afirmación. «Son soluciones generales repetibles a un problema de ingeniería de *software* recurrente, destinadas a obtener un *software* menos vulnerable y un diseño más resistente y tolerante a los ataques, normalmente se limitan a funciones y controles de seguridad a nivel del sistema, comunicaciones e información».

- A. Test de Penetración.
- B. Revisión de código.
- C. Casos de abuso.
- D. Principios de diseño.

10. Señala las respuestas correctas. ¿En qué fases modelo de ciclo de vida seguro McGraw son aplicables el catálogo conocimiento de seguridad «exploit»?

- A. Codificación.
- B. Pruebas y resultados.
- C. Realimentación de producción.
- D. Plan de pruebas.