Sociedad Andaluza para el Desarrollo de las Telecomunicaciones
CONSEJERÍA DE EMPLEO, EMPRESA Y COMERCIO

seguridad+ Y CONFIANZA DIGITAL | AndalucíaCERT CENTRO DE SEGURIDAD TIC

Informe de divulgación
Seguridad en Aplicaciones Web

Tipo de documento:	*Informe*
Autor del documento:	*AndalucíaCERT*
Código del Documento:	*CERT-IF-9831-160316*
Edición:	*0*
Categoría	*Uso Interno*
Fecha de elaboración:	*04/05/2016*
N° de Páginas	*1 de 21*

1 TABLA DE CONTENIDOS

2 OBJETIVO

El objeto de este documento es dar a conocer la situación actual del mercado de aplicaciones WEB desde el punto de vista de la seguridad de la información.

3 ALCANCE

Este documento va dirigido tanto al personal de la Junta de Andalucía, como al público en general. Pretende aportar las nociones básicas necesarias para entender y tener conocimiento sobre la situación actual en cuanto a las aplicaciones web se refiere: el estado del arte (estructura de una página web, principales plataformas e infraestructuras web, marco de desarrollo...), los riesgos y amenazas a las que están expuestas (sus vectores de ataque, las vulnerabilidades más comunes...), así como la forma de abordar estos problemas (el desarrollo seguro, las principales metodologías de desarrollo, las tecnologías usadas para combatir a los cibercriminales...).

4 APLICACIONES WEB. ESTADO DEL ARTE

En ingeniería del software una aplicación web es un programa que los usuarios pueden utilizar accediendo a un servidor remoto mediante un cliente ligero denominado navegador.

Las aplicaciones web siguen, por tanto, el modelo cliente-servidor. Su popularidad y ubicuidad residen en los siguientes principios:

- No es necesario distribuir e instalar el software de las aplicaciones web en los clientes. Bastará con que éstos tengan un navegador web actualizado para poder acceder y disfrutar de ellas.
- Son fáciles de mantener. Simplemente con modificar el código de la aplicación en el servidor, todos los usuarios accederán a la última versión.
- Independencia del sistema operativo. Las aplicaciones son multiplataforma, puesto que cualquier navegador puede ejecutarlas independientemente del sistema operativo usado por el cliente.
- Los clientes necesitarán poca capacidad de cómputo para ejecutarlas. El grueso de las operaciones computacionales recae sobre el servidor web, por lo que los clientes que hagan uso de estas aplicaciones únicamente necesitarán visualizarlas en su navegador.

Algunos casos de éxito relacionados con aplicaciones web que podemos mencionar son: webmails, foros y blogs de Internet, tiendas online...

4.1 Arquitecturas web

Las aplicaciones web comenzaron su andadura en Internet siendo páginas estáticas que se entrelazaban entre ellas (hipervínculos). Poco a poco se les fueron agregando funciones para generar contenido de forma dinámica, hasta llegar al momento actual (tecnologías como scripts ejecutados en el lado del cliente, Flash, Ajax, HTML5...).

En una primera aproximación, es posible definir la estructura de una página web como un modelo de tres capas: presentación, aplicación y almacenamiento.

- El navegador web sería el encargado de la primera capa, es decir, de la presentación de los datos al usuario final.
- La capa de aplicación corresponde a los servidores web, que mediante algún sistema de procesado que genere contenido dinámico, debe servir las páginas web solicitadas por los clientes.
- Por último, existe una capa de almacenamiento de información que debe encargarse de custodiar y tratar los datos en base a los que se generan las páginas web.

Imagen 1. Estructura de una página web según el modelo de tres capas. Fuente [20]

Obviamente, el modelo anterior supone una primera aproximación, puesto que en realidad la casuística puede ser mucho mayor: la capa de almacenamiento puede estar implementada por una simple base de datos, o bien enlazar con un ERP empresarial, o bien con toda una infraestructura cloud... De igual forma, la capa de aplicación puede estar gestionada por toda una granja de servidores, un contenedor docker alojado en un servidor compartido con otras aplicaciones... En muchos casos tiene más sentido hablar de un modelo de n-capas, según la complejidad del escenario.

Por otro lado, y como en cualquier aplicación que siga el modelo cliente-servidor, es posible hablar de una estructura separada en front-end y back-end, que se refieren a la separación de intereses entre una

capa de presentación y una capa de acceso a los datos, respectivamente. En las aplicaciones web esta estructura cobra aún más sentido, ya que generalmente la parte de front-end es ejecutada en los ordenadores de los propios usuarios, mientras que la parte de back-end es procesada por los servidores web.

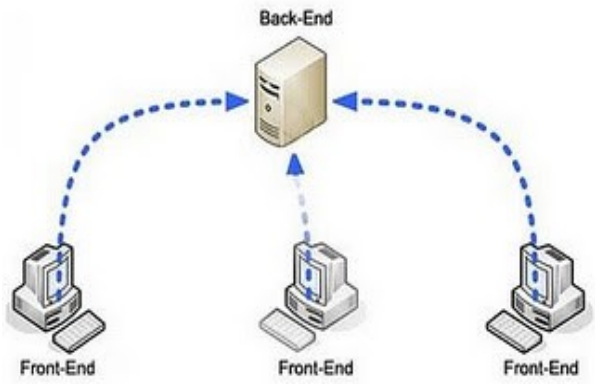

Imagen 2. Distinción entre front-end y back-end para una aplicación web, según el lugar en que se ejecute. Fuente [21]

Desde el punto de vista del desarrollo de aplicaciones web, la estructura front-end/back-end también tiene sus implicaciones, puesto que los desarrolladores de front-end se encargarán del diseño de la aplicación web (es decir, la vista que obtendrán los usuarios finales), mientras que los desarrolladores de back-end se encargarán de la parte lógica de la app.

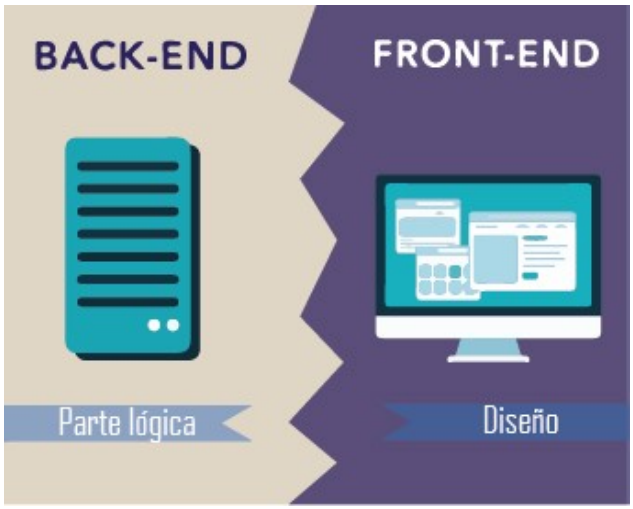

Imagen 3. Distinción entre front-end y back-end para una aplicación web y su implicación en el desarrollo. Fuente [19]

4.2 Tecnologías web

Llegados a este punto es necesario hablar de las tecnologías usadas en la programación de aplicaciones web. Debemos indicar que se trata de un ecosistema muy diverso y en continuo cambio. Para abordarlo se hará una distinción entre el código escrito para ser ejecutado en el lado del servidor y aquel que es escrito para ser ejecutado en el lado del cliente.

Programación en el lado del servidor

En la siguiente gráfica se muestran los lenguajes de programación más usados para la programación web en el lado del servidor. Dicha gráfica ha sido elaborada partiendo de un estudio realizado por la web W3Techs y sus datos están actualizados a junio de 2016:

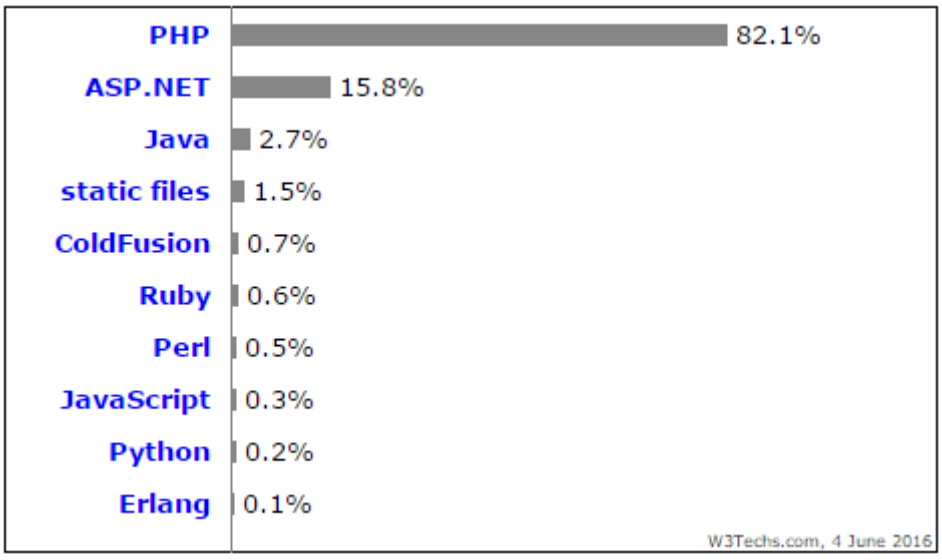

Imagen 4. Lenguajes de programación más usados en el lado del servidor (junio 2016). Fuente: W3Techs.

NOTA: Una web puede usar más de una tecnología en el lado del servidor.

Como se puede observar, la tecnología dominante es PHP. A mucha distancia se encuentran las soluciones ASP.NET de Microsoft y Java de Oracle. No obstante, hay que hacer referencia al creciente protagonismo que están adquiriendo tecnologías como Ruby y Python, además del intento por llevar el lenguaje JavaScript a la programación de aplicaciones web en el lado del servidor (Node.JS es un claro ejemplo de esto último).

No hay que olvidarse que en el lado del servidor es donde se debe gestionar el almacenamiento y tratamiento de los datos, por lo que al margen del lenguaje de programación usado, será necesario el uso

de otras tecnologías como bases de datos relacionales (MySQL, PostgreSQL, Oracle...), no-SQL (mongoDB, Cassandra...), sistemas de BigData, etc.

La complejidad del escenario no acaba ahí, puesto que existen una infinidad de gestores de contenidos (WordPress, Joomla!, Drupal, Magento, Typo3...), así como frameworks y librerías (Synfony, Laravel...) que ponen a disposición de los desarrolladores un amplio abanico de opciones donde poder elegir para implementar su proyecto.

Programación en el lado del cliente

Basándonos nuevamente en los datos proporcionados por la web W[3]Techs, si nos centramos en las tecnologías usadas en el lado del cliente hay un claro dominador: el lenguaje JavaScript. Combinado con los tradicionales HTML y CSS permite generar páginas web atractivas desde un punto de vista estético, que se adaptan a cualquier dispositivo en que sean visualizadas y que aporten cierto dinamismo.

Imagen 5. Lenguajes de programación más usados en el lado del cliente (junio de 2016). Fuente: W[3]Techs.

NOTA: Una web puede usar más de una tecnología en el lado del cliente.

Igual que sucedía con los lenguajes de programación usados en el lado del servidor, existen multitud de librerías y frameworks que complican más aún el escenario y amplían la diversidad de opciones donde elegir: JQuery, Bootstrap, MooTools, Angular.JS...

4.3 Infraestructuras web

Dada la diversidad y heterogeneidad presente en el mercado de las aplicaciones web, las distintas infraestructuras usadas para prestar estos servicios no podían ser menos y presentan una diversidad y complejidad crecientes.

Sin lugar a dudad, el rey en cuanto a los sistemas de infraestructura usados para las aplicaciones web es el sistema LAMP. Este acrónimo hace referencia al uso de las tecnologías Linux (como sistema ope-

rativo) + Apache (como servidor web) + MySQL (o MariaDB, como sistema relacional de bases de datos) + PHP (como lenguaje de scripting).

Pese a que existen variantes (WAMP para sistemas MS Windows, LEMP sustituyendo Apache por nginx...), la realidad es que el dominador absoluto continúa siendo la infraestructura LAMP.

Las aplicaciones web desarrolladas estarán ubicadas en la cima de toda esta pila de tecnologías.

Pese a lo indicado, no hay que olvidar algunas de las últimas tendencias, como la distribución y despliegue de aplicaciones web en contenedores o el uso de sistemas cloud: SaaS, PaaS... que en muchos casos presentan incluso la misma infraestructura subyacente.

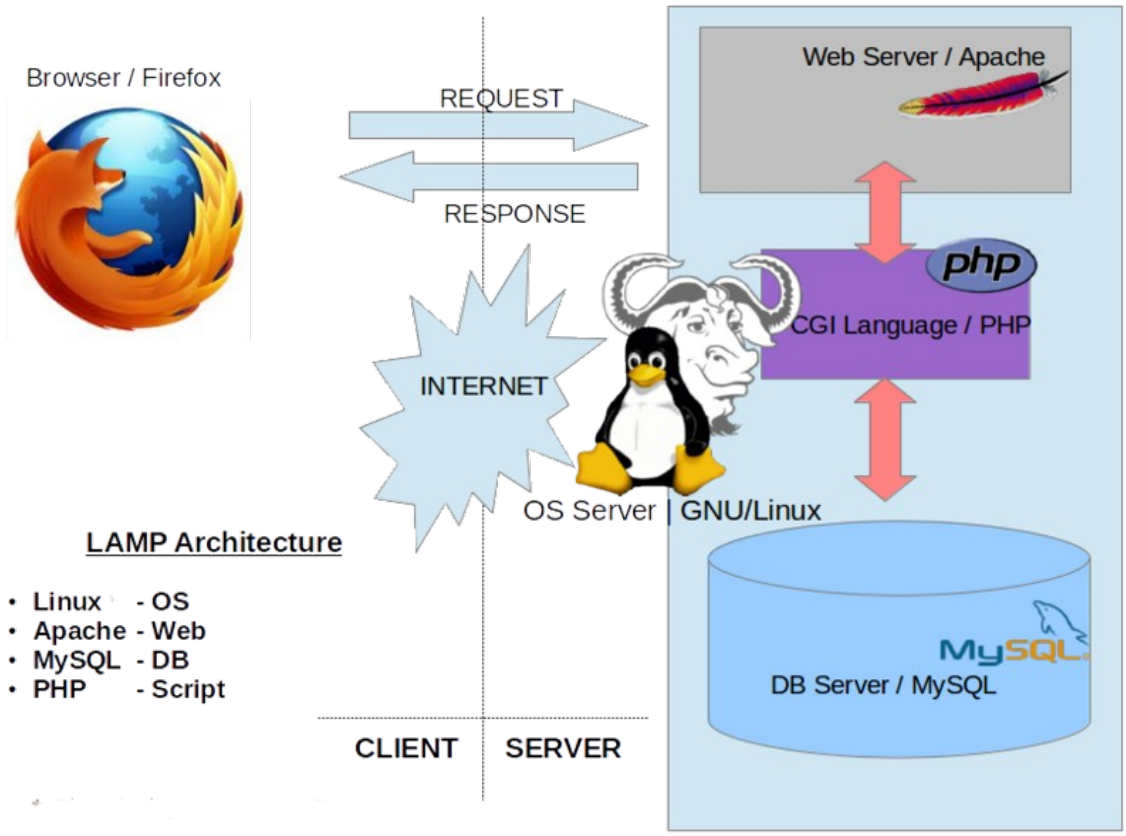

Imagen 6. Sistema de infraestructura LAMP.

Dentro de las infraestructuras web no hay que olvidarse de un componente fundamental que no está presente en el lado del servidor, sino precisamente en el lado del cliente: el navegador web. Todo

usuario que quiera acceder a servicios web debe tener instalado un browser en su equipo, lo cual supone un nuevo foco a tener en cuenta para la seguridad, puesto que no sólo las aplicaciones alojadas en servidores pueden ser vulnerables; los propios navegadores presentan vulnerabilidades y son susceptibles de sufrir ataques.

Desde principios de los años 90 del siglo pasado hasta la actualidad, los navegadores web han sufrido una evolución asombrosa y su complejidad ha ido creciendo de forma exponencial, por lo que actualmente suponen casi un sistema operativo en sí mismos.

En la siguiente imagen se muestra la evolución en el porcentaje de uso de los principales browsers. Como se puede observar el ecosistema móvil cada vez toma un mayor protagonismo:

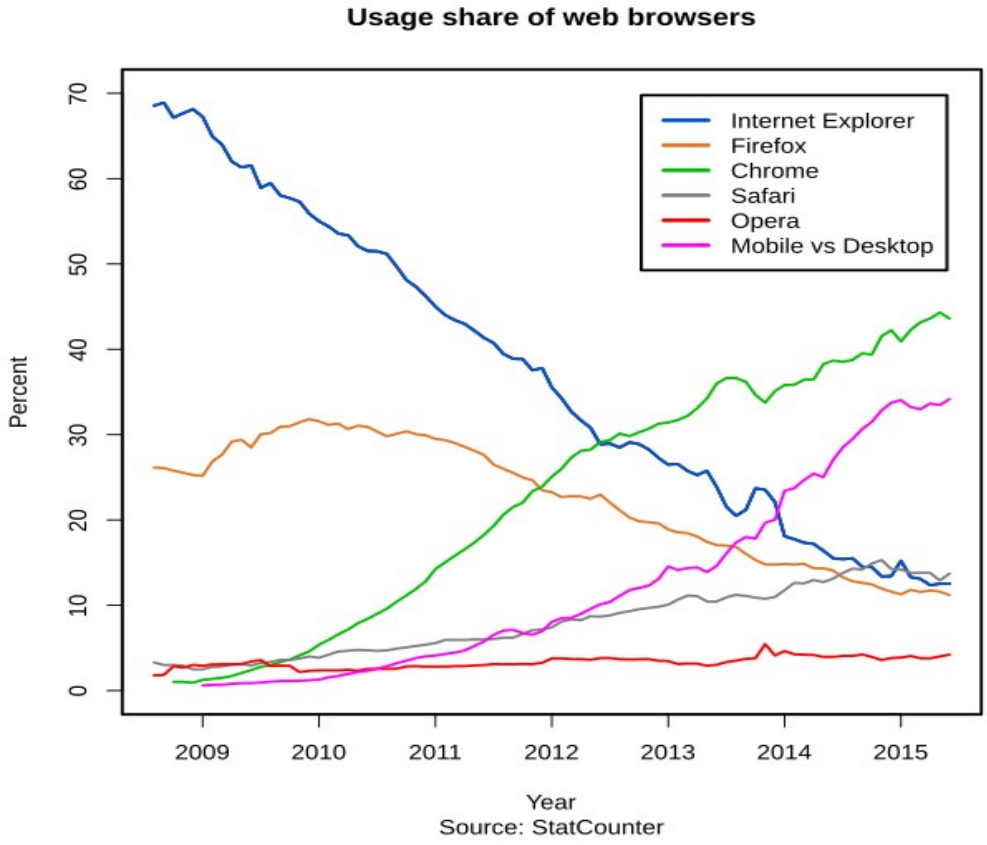

Imagen 7. Evolución del porcentaje de uso de los principales navegadores web en los últimos años.

5 APLICACIONES WEB. RIESGOS Y AMENAZAS

Una vez conocido el diverso ecosistema que suponen las tecnologías web actuales (aunque sea grosso modo), se hace necesario mencionar los riesgos y amenazas a las que se exponen.

Las grandes amenazas a la seguridad de las aplicaciones web están asociadas a las siguientes características intrínsecas a este tipo de entornos:

- Una superficie de ataque muy elevada: las aplicaciones web están públicamente disponibles, por lo que su ubicuidad y accesibilidad son totales. Hay una infinidad de recursos y servicios que están expuestos de cara a Internet en entornos de producción y que, por tanto, suponen un blanco potencial para los ciberdelincuentes y otro tipo de activistas.
- Los cortafuegos tradicionales son de poca utilidad en estos entornos: puesto que las aplicaciones web usan puertos estándar (TCP 80/443), la mayoría de los firewalls dejarán pasar el tráfico dirigido contra las webs, incluso si éste es malicioso.
- Suponen un punto de acceso al resto de la infraestructura de la organización: una vez comprometido un servidor web, se facilitaría el acceso a otros sistemas de la organización mediante técnicas de pivoting. Esto se une al hecho de que en ocasiones los servidores almacenen información sensible que puede llegar a ser robada.
- El anonimato por parte del atacante: dada la complejidad de Internet, así como la existencia de proxies y otros servicios como VPNs, servidores de tipo bulletproof, redes anónimas (Tor, I2P, FreeNET...), etc., se propicia el hecho de que en muchas ocasiones sea imposible determinar el origen real de un ataque, por lo que aumenta la sensación de impunidad por parte de los ciberdelincuentes.
- Soluciones out-of-the-box que no han sido correctamente bastionadas: en la mayoría de las ocasiones, una página web puede verse como una solución ou-of-the-box, es decir, lista para funcionar nada más ser entregada. Se trata de desarrollos realizados por un tercero que, en muchas ocasiones, no ha contemplado el problema de la seguridad de dicha aplicación. Hay cientos de webs vulnerables publicadas en Internet.

5.1 Proyecto OWASP – OWASP Top 10

Cuando se habla se seguridad en aplicaciones web hay un referente a nivel mundial que no puede ser obviado: el proyecto OWASP. El proyecto abierto de seguridad en aplicaciones web se dedica a determinar y combatir las causas que hacen que el software sea inseguro.

De entre todas las publicaciones de interés que libera el proyecto OWASP, tal vez la que más nos interese al hablar de los riesgos y amenazas a la seguridad de las aplicaciones web sea el “OWASP Top

10". Se trata de un documento publicado cada tres años, que recoge los diez riesgos de seguridad más importantes para las aplicaciones web.

La última publicación data del año 2013 y recoge las siguientes amenazas:

A1. Inyecciones: una inyección (ya sea de tipo SQL, LDAP, o algún otro), ocurre cuando un dato no confiable es enviado a un intérprete como parte de una consulta o comando, y dicho intérprete no es capaz de validarlo correctamente. Esto podría llegar a provocar la ejecución de comandos no deseados o el acceso a datos sin la correspondiente autorización.

A2. Pérdida de autenticación y gestión de sesiones: las funciones encargadas de los procesos de autenticación y gestión de sesiones están a menudo implementados de forma incorrecta, lo que puede permitir a un atacante comprometer contraseñas, tokens... o llegar a asumir la identidad de otros usuarios.

A3. Cross-Site Scripting (XSS): este tipo de problema de seguridad sucede cuando una aplicación web toma datos no confiables y los envía a un navegador web sin una correcta validación, pudiendo provocar la ejecución de código de tipo script en el browser del usuario.

A4. Referencia directa insegura a objetos: este tipo de problema de seguridad ocurre cuando un desarrollador expone una referencia a un objeto interno (un fichero, una base de datos...). Sin el correspondiente control de accesos, los atacantes podrían manipular dichas referencias.

A5. Configuración de seguridad incorrecta: las buenas prácticas de seguridad requieren la existencia de una serie de políticas de seguridad para la configuración y el despliegue de aplicaciones, frameworks, servers... Dichas políticas deben estar definidas, implementadas y mantenidas, así como ser diferentes de las configuraciones por defecto (ya que éstas suelen ser inseguras). Además, el software debe permanecer actualizado.

A6. Exposición de datos sensibles: Muchas aplicaciones web no protegen de forma adecuada los datos sensibles que puedan albergar (como tarjetas de crédito, información sobre impuestos, credenciales de autenticación...). Los atacantes podrían robar o modificar dicha información, por lo que se exige una capa extra de protección para estos datos.

A7. Ausencia de control de acceso a determinadas funciones: muchas aplicaciones web ocultan determinadas funcionalidades a los usuarios simplemente no mostrando dichas opciones en la interfaz web, pero no realizan una verificación a nivel de servidor sobre si un usuario tiene realmente privilegios para hacer uso de dicha función. Este hecho podría permitir a una atacante realizar peticiones maliciosas al servidor con el fin de usar ciertas funciones para las que realmente no tiene privilegios.

A8. Falsificación de peticiones de sitios cruzados (Cross-Site Request Forgery, CSRF): en un ataque de tipo CSRF se fuerza al navegador de un usuario ya autenticado en una aplicación web, a enviar solicitudes HTTP falsificadas (y que pueden contener cookies de sesión e información sobre la autenticación) a una aplicación web vulnerable. Con esto se consigue que el navegador de la víctima envíe peticiones que la aplicación web vulnerable cree que son legítimas.

A9. Uso de componentes con vulnerabilidades conocidas: en muchos casos, las librerías, los frameworks, ciertos módulos... se ejecutan en el sistema con todos los privilegios. Si un componente vulnerable es explotado, podría provocar el compromiso de todo el sistema, por lo que el uso de componentes con vulnerabilidades conocidas amplía la superficie de exposición y los posibles vectores de ataque.

A10. Redirecciones y reenvíos no validados: es frecuente que las aplicaciones web redirijan a sus usuarios a otras webs, por lo que, sin una correcta validación, un atacante podría redirigir a una víctima hacia sitios web maliciosos.

Debe quedar claro que la verificación de los riegos recogidos en el OWASP Top 10 es un requisito necesario, pero no suficiente. Es decir, se debe considerar como el mínimo conjunto de riesgos y amenazas que deberían ser verificados en toda aplicación web, pero el trabajo debe ir más allá y considerar otros riesgos y aspectos de seguridad.

5.2 Otros riesgos y configuraciones seguras a tener en cuenta

En ocasiones, los riesgos y amenazas presentes en las aplicaciones web no vienen dados por la propia aplicación, sino por la infraestructura que se encarga de servirla. En estos casos, el problema no es de la aplicación en sí misma, sino del despliegue realizado para poner en producción la misma:

La propia OWASP ya recoge como riesgo la "configuración insegura", pero esto puede ir más allá. Algunos ejemplos son los siguientes:

- No facilitar el acceso mediante el protocolo seguro HTTPS a los formularios de autenticación: si únicamente está contemplado servir un formulario de login por el puerto 80 (HTTP), esto implica que las credenciales de los usuarios viajen en texto claro por la red y puedan ser fácilmente capturadas.
- Ausencia de ciertas cabeceras en las respuestas del servidor web: X-XSS-PROTECTION, X-FRAM-OPTIONS, el flag HTTPOnly en las cookies de sesión... todos estos mecanismos proporcionan una capa extra de seguridad que debe ser configurada en los servidores, pero que no es empleada de forma habitual.
- Contenedores no bastionados o desactualizados: cada vez es más común la distribución y despliegue de software por medio de contenedores (por ejemplo docker), pero esto supone un nuevo riesgo para la seguridad, ya que al igual que los servidores físicos, los contenedores deben ser bastionados y estar actualizados con los últimos parches de seguridad. En muchas ocasiones no queda claro quién es el responsable de dichas tareas o el proveedor encargado del empaquetado no lo ha tenido en cuenta, por lo que quedan en el limbo.

- Delegar el riesgo en un tercero: al igual que sucede con los contenedores usados para la distribución y el despliegue de software, cada vez es más común el uso de sistemas SaaS (Software as a Service) o PaaS (Platform as a Service). No dejan de ser tecnologías basadas en cloud en las que se delegan las funciones relacionadas con el bastionado y la aplicación de parches de seguridad en un tercero. Es necesario verificar cómo gestiona dichas tareas nuestro proveedor.

6 APLICACIONES WEB. ESTRATEGIA Y RECOMENDACIONES

A la hora de abordar la seguridad de una aplicación web será necesario considerar una serie de aspectos ya desde las fases de diseño y concepción de dicha web. La seguridad ha de verse como un elemento horizontal que debe estar presente en todo el ciclo de vida de la app. Esto es, la seguridad debe ser uno más de los requisitos funcionales.

En este sentido ayudará tener en cuenta las siguientes recomendaciones:

- Requisitos para una aplicación segura: Se debe definir qué significa que una aplicación sea segura y qué implicaciones pueda tener para el proyecto. Un ejemplo que puede servir como punto de partida para muchos casos sería el OWASP Secure Software Contact Annex.
- Arquitectura seguridad: La manera más económica de desarrollar software seguro es tener en cuenta la seguridad desde el comienzo. En este sentido, el uso de guías de desarrollo seguro, así como la definición de políticas de seguridad para el despliegue y mantenimiento de las infraestructuras que presten los servicios, suponen un gran punto de partida.
- Controles de seguridad estándar: Se deben definir e implementar controles de seguridad robustos que garanticen la seguridad de nuestra aplicación y de los datos que ésta almacena. No sólo eso, también se debe hacer un seguimiento de quién hace uso de sus privilegios para acceder a dichos datos. De esta forma lograremos, en cierta forma, una monitorización de nuestro sitio web.
- Ciclo de vida del desarrollo seguro: como todo desarrollo software, la creación de una aplicación web implica la existencia de un ciclo de vida en el que la seguridad debe estar presente en todos sus estadios. Existen diversas metodologías que se pueden usar para este epígrafe: S-SDLC de Microsoft, CLASP de OWASP…
- Educación/Formación en la seguridad de las aplicaciones web: las apps son desarrolladas por personas, por lo qué éstas deberán tener una formación acorde que les permita conocer las principales amenazas que pueden dar al traste con su trabajo. De igual forma, una vez creada la aplicación, son personas quienes la usarán, por lo que hay que realizar un proceso de concienciación sobre éstas para que comprendan los peligros a los que se enfrentan.

6.1 Desarrollo seguro. Metodologías y ciclo de vida de una aplicación web segura.

Como ya se ha indicado, el uso de guías de desarrollo seguro, así como la aplicación de ciertas metodologías de desarrollo centradas en la seguridad web permitirá crear un código de calidad y libre de vulnerabilidades.

En cualquier desarrollo software debe estar presente el concepto SDLC (ciclo de vida del desarrollo de software, por sus siglas en inglés). Se trata de un proceso lógico de creación o modificación de los sistemas, modelos y metodologías que la gente usa para desarrollar software. Consta de las siguientes fases:

- Análisis y definición de requisitos.
- Diseño.
- Implementación/Desarrollo.
- Pruebas.
- Evaluación.

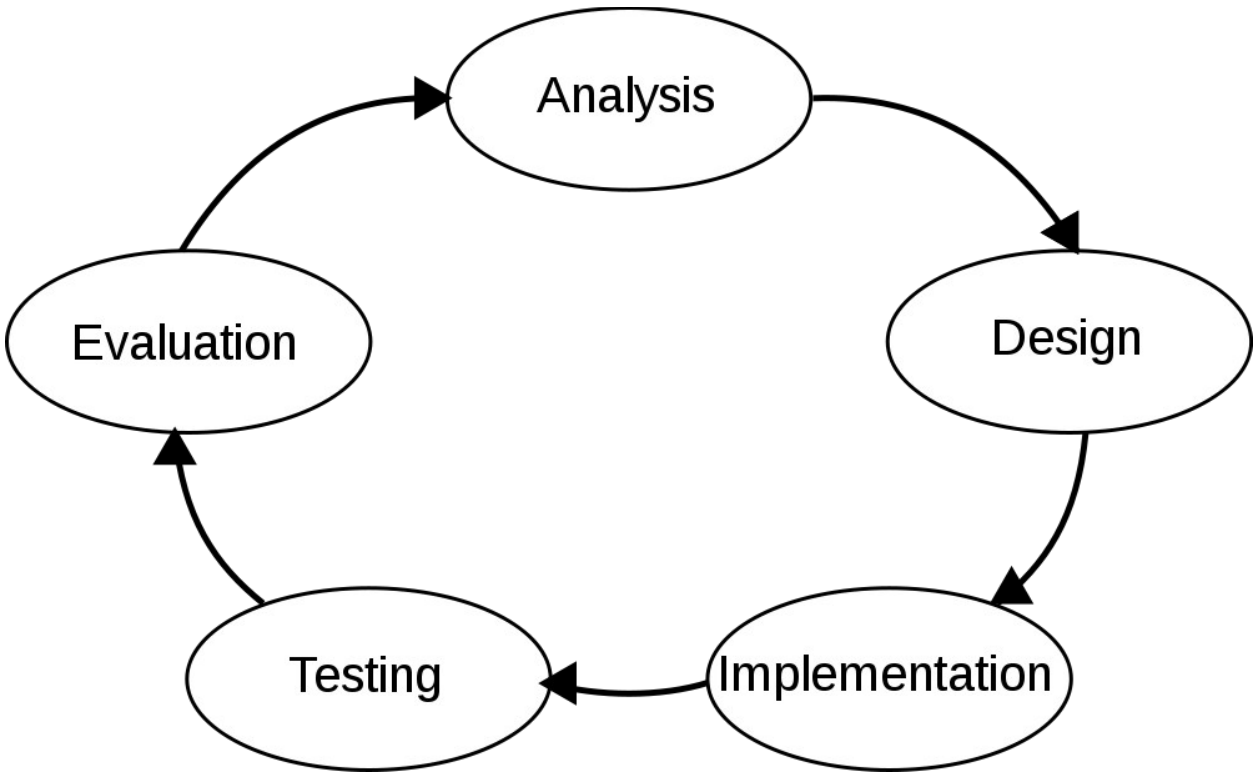

Imagen 8. Burbuja de mantenimiento en el Ciclo de Vida del Desarrollo Software.

Hay que entender que estas fases son cíclicas, puesto que siguen el principio de mejora continua (o ciclo de Deming).

¿Y cómo encaja la seguridad en el modelo SDLC? Como se puede observar en la siguiente imagen:

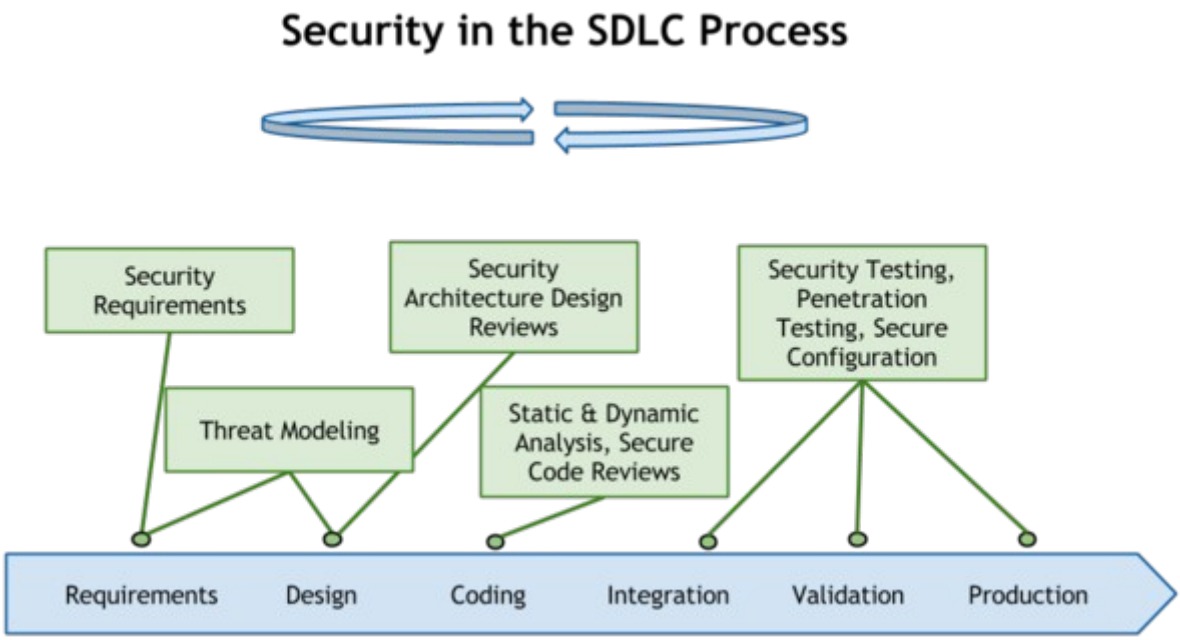

Imagen 9. La seguridad en el proceso SDLC. Fuente [22]

Tal y como se ha indicado con anterioridad, la seguridad debe estar presente ya en el momento de definir los requisitos funcionales de una aplicación. Además, ya en esa fase, así como en la fase de diseño, se deben tener presentes las distintas amenazas a la seguridad con las que nos podemos encontrar y las arquitecturas que usaremos (así como la configuración de las mismas).

Durante la fase de implementación se deberán realizar análisis de código (tanto estáticos como dinámicos) y practicar una auditoría a dicho código para garantizar su seguridad.

Ya en las fases de pruebas y despliegue de la aplicación, será necesario realizar un test de penetración para dicha aplicación web, lo cual servirá para conocer de primera mano, no sólo la seguridad de la propia app, sino también de la infraestructura que se encarga de servirla.

Por último, y tal y como se ha indicado, el proceso será cíclico, por lo que una vez desplegada la aplicación será necesario una fase mantenimiento y búsqueda de errores (o de posibles mejoras). Esto forzará a los desarrolladores/mantenedores a entrar nuevamente en el ciclo.

Ya hemos indicado que existen diversas metodologías de diseño seguro que se pueden aplicar al ciclo de vida de una aplicación web. Algunos de ellos son: Microsoft Trustworthy Computing SDL, OWASP Clasp... Por ejemplo, McGraw propone los siguientes epígrafes para el ciclo de vida del desarrollo seguro:

- Revisión de código (code review). Tarea de análisis de código estático, el cual debe ser escrito teniendo conocimientos de seguridad y buenas prácticas de programación. Fase de implementación.
- Análisis de riesgo. Esta tarea es ejecutada en tres fases y es de vital importancia en la toma de decisiones del proceso. Fase de requisitos, análisis, diseño y testing.
- Test de intrusión (Pentesting). Tanto en la fase de testing como la liberación de la herramienta, este tipo de tareas pueden descubrir comportamientos anómalos en la herramienta. Fase de testing.
- Test de caja negra basados en riesgos. Fase de testing.
- Casos de abuso o fuzzing a los inputs de la herramienta para comprobar su comportamiento. Fase de testing.
- Requisitos de seguridad por parte de los desarrolladores. Fase de requisitos y análisis.
- Operaciones de seguridad.
- Análisis externo, no obligatorio. Durante todas las fases.

Puede encontrar más información en nuestro Informe Divulgativo "Seguridad en el ciclo de vida de los sistemas TIC" de diciembre de 2015.

6.2 Otros aspectos y estrategias a tener en cuenta en el desarrollo/despliegue de una app.

Al margen de los aspectos de seguridad ya abordados, desde AndalucíaCERT nos gustaría hacer las siguientes recomendaciones:

- Pentesting persistente: Tal y como se ha indicado al hablar del ciclo de vida del desarrollo seguro, éste es un proceso cíclico. El empleo de sistemas de pentesting persistente o test de intrusiones continuos, facilitará la búsqueda de errores en nuestros sistema y reducirá los tiempos de corrección de las vulnerabilidades detectadas.

- Uso de un sistema WAF: Como se indicó al comienzo de este informe, el tráfico web se suele cursar por puertos estándar (TCP 80/443), por lo que los cortafuegos lo dejan pasar incluso si dicho tráfico es malicioso. Es por ello que se hace necesario emplear otro tipo de firewalls, en este caso de nivel de aplicación, que protejan nuestra aplicación web.
- Uso de sistemas SIEM: Cualquier aplicación web, por el simple hecho de estar accesible en Internet, se convertirá en un posible blanco de ataques. Es por ello que se hace necesario monitorizar los sistemas y estar pendientes ante posibles incidentes.
- Contar con personal de respuesta ante incidentes (ERI): Entroncando con el anterior punto, no sólo será necesario estar monitorizando nuestros sistemas en busca de posibles amenazas e incidentes, sino que una vez que éstos se hayan producido, será necesario contar con personal experto que gestione el caso y aporte soluciones a la incidencia.
- Uso de sistemas con factores de autenticación múltiples: Soluciones como Google Authenticator o Latch permiten añadir una nueva capa de seguridad que ayudará a las aplicaciones a fortificar las cuentas de los usuarios del sistema y a mejorar los procesos de control de acceso.

6.3 El eslabón mas débil. El usuario final

Desde AndalucíaCERT queremos volver a insistir en que la seguridad en los entornos web no sólo depende de los desarrolladores/administradores de sistemas. Las aplicaciones están pensadas para que sean usadas por personas y éstas son, en no pocas ocasiones, el blanco de los ataques realizados.

El primer punto a tener en cuenta desde el punto de vista de un usuario final es el sistema que usa para navegar por la web. El browser se ha convertido en un software complejo, con cientos de funcionalidades y plugins que lo convierten en un mini sistema operativo. Tenerlo actualizado y con todos los parches de seguridad aplicados es vital para evitar que se produzcan incidentes.

Por otro lado, desde AndalucíaCERT hace ya tiempo que hemos recomendado que se deshabiliten ciertas extensiones como Adobe Flash y Oracle Java. Únicamente se deberían habilitar en momentos concretos y bajo situaciones controladas.

Al margen del software usado para navegar por la web, la labor de educación/concienciación para con los usuarios finales debe ser constante. Enseñar a un usuario a determinar si una web es legítima, a no seguir ciertos enlaces que considere sospechosos, a dudar cuando se le solicita información confidencial en algún formulario web.... puede ayudar a mitigar o erradicar incidentes de phishing, infecciones por malware, etc.

7 CONCLUSIONES

Tras la lectura del presente informe debe quedar claro que el ecosistema web es complejo y muy heterogéneo. Además, y dadas sus características intrínsecas, dichos entornos están expuestos a multitud de riesgos y amenazas que fuerzan a los desarrolladores/administradores de sistemas a emplear diversos métodos para garantizar su seguridad.

Desde el punto de vista del desarrollo web, debe quedar claro que la seguridad es un requisito funcional más. Debe tenerse en cuenta desde las primeras fases del diseño de una aplicación. Desde el punto de vista de los administradores, la correcta configuración y monitorización de una app, facilitarán la detección de problemas y acortarán los tiempo de resolución de incidentes.

No hay que olvidar al que, en no pocas ocasiones, es el eslabón más débil en la seguridad: el usuario final. El mantenimiento de un navegador web actualizado y con todos los parches de seguridad aplicados, así como deshabilitar ciertas extensiones y, por supuesto, la formación y concienciación, podrán evitar ciertos incidentes y mejorar la experiencia de uso de nuestros usuarios.

8 GLOSARIO

Back-end: también conocido como motor o dorsal. Hace referencia a la capa de acceso a los datos.

Base de datos: Banco de información que contiene datos relativos a diversas temáticas y categorizados de distinta manera, pero comparten entre sí algún tipo de vínculo o relación, para poder ordenarlos y clasificarlos. Las bases de datos pueden ser relacionales y no relaciones (o no-SQL).

BigData: es un concepto que hace referencia al almacenamiento de grandes cantidades de datos y a los procedimientos usados para procesarlos.

Browser o navegador web: Programa informático usado por un cliente para acceder a contenidos de la web.

Bulletproof server: o servidor a prueba de balas. Se trata de un tipo particular de servidores para los cuales nuestro proveedor nos garantiza que permitirá que lo usemos para cualquier fin que deseemos (incluso si éste es malicioso) y que no atenderá ningún tipo de reclamación o rogatoria que se realice sobre el mismo.

Cloud: también conocido como computación en la nube, o nube. Es un paradigma que permite ofrecer servicios de computación a través de una red.

Coockie: Pequeña cantidad de información enviada por un sitio web y almacenada en el navegador del usuario, de tal manera que el sitio web puede consultar la actividad previa de dicho usuario.

CSS: hojas de estilo en cascada. Es un lenguaje usado para definir y crear la presentación de un documento estructurado escrito en HTML.

ERP: Enterprise resource planning. Son sistemas de información gerenciales que integran y manejan muchos de los aspectos relacionados con el negocio, las operaciones de producción y las operaciones de distribución de una compañía.

Firewall o cortafuegos: Parte de un sistema o red que está diseñado para bloquear el acceso no autorizado, permitiendo al mismo tiempo las comunicaciones que sí lo están.

Front-end: también conocido como interfaz o frontal. Hace referencia a la capa de presentación de datos.

Framework: Conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia para enfrentar y resolver problemas de índole similar.

Hipervínculo: también llamado enlace, vínculo o hiperenlace. Se trata de un elemento de un documento electrónico que hace referencia a otro recurso (que puede estar presente en el mismo o en otro documento).

HTML: HypèrText Markup Language. Lenguaje de marcado usado para la elaboración de páginas web.

HTTP: HyperText Transfer Protocol. Es el protocolo de comunicaciones que permite las transferencias de información en la web.

LDAP: Lightweight Directory Access Protocol. Se trata de un protocolo de nivel de aplicación que permite el acceso a un servicio de directorio ordenado y distribuido para buscar información diversa en un entorno de red.

Proxy: Servidor que hace de intermediario en las peticiones de recursos que realiza un cliente a otro servidor.

Script: También conocido como archivo de órdenes o archivo de procesamiento por lotes. Se trata de un programa generalmente simple, que normalmente se almacena en un fichero de texto plano y es interpretado por otro sistema para irlo ejecutando paso por paso.

Servidor: Aplicación software capaz de atender las peticiones de un cliente.

SIEM: Security Information and event management. Sistema que proporciona un sistema de alertas en tiempo real sobre posibles violaciones de políticas en las redes, aplicaciones y sistemas que monitoriza.

SQL: Structured Query Language. Lenguaje declarativo que permite el acceso y la consulta a una base de datos relacional.

VPN: o Red Privada Virtual, es una tecnología que permite la extensión segura de una red de área local sobre una red pública o no controlada como Internet.

WAF: Web application firewall. Cortafuegos de nivel de aplicación.

Web : Relativo a la World Wide Web, una red informática mundial usada para la distribución de documentos de hipertexto interconectados y accesibles desde Internet.

9 DOCUMENTACION DE REFERENCIA

[1] Aplicación Web. Wikipedia en español. https://es.wikipedia.org/wiki/Aplicación_web (Fecha de consulta: 26/05/2016).

[2] Web application. Wikipedia en inglés. https://en.wikipedia.org/wiki/Web_application (Fecha de consulta: 26/05/2016).

[3] Front-end y back-end. Wikipedia en español. https://es.wikipedia.org/wiki/Front-end_y_back-end (Fecha de consulta: 31/05/2016).

[4] Usage of server-side programming languages for web-sites. W3Techs. https://w3techs.com/technologies/overview/programming_language/all (Fecha de consulta: 04/06/2016).

[5] Usage of clien-side programming languages for websites. W3Techs. https://w3techs.com/technologies/overview/client_side_language/all (Fecha de consulta: 04/06/2016).

[6] LAMP (software bundle). Wikipedia en inglés. https://en.wikipedia.org/wiki/LAMP_(software_bundle) (Fecha de consulta: 02/062016).

[7] Web browser. Wikipedia en inglés. https://en.wikipedia.org/wiki/Web_browser (Fecha de consulta: 06/06/2016).

[8] Open Web Application Security Project. Wikipedia en español. https://es.wikipedia.org/wiki/Open_Web_Application_Security_Project (Fecha de consulta: 03/06/2016).

[9] Sobre OWASP. OWASP Wiki. https://www.owasp.org/index.php/Sobre_OWASP (Fecha de consulta: 03/06/2016).

[10] OWASP Top 10. Wikipedia en español. https://es.wikipedia.org/wiki/OWASP_Top_10 (Fecha de consulta: 02/06/2016).

[11] OWASP Top 10 2013. Proyecto OWASP. http://owasptop10.googlecode.com/files/OWASP%20Top%2010%20-%202013.pdf (Fecha de consulta: 02/06/2016).

[12] OWASP Secure Software Contract Annex. Proyecto OWASP. https://www.owasp.org/index.php/OWASP_Secure_Software_Contract_Annex (Fecha de consulta: 06/06/2016).

[13] Systems Develpment Life Cycle. Wikipedia en español. https://es.wikipedia.org/wiki/Systems_Development_Life_Cycle (Fecha de consulta: 06/06/2016).

[14] Ciclos de Vida del Software Seguros (S-SDLC) (Parte I). Flu-Project. http://www.flu-project.com/2014/05/ciclos-de-vida-del-software-seguros-s.html (Fecha de consulta: 06/06/2016).

[15] Ciclos de Vida del Software Seguros (S-SDLC) (Parte II). Flu-Project. http://www.flu-project.com/2014/05/ciclos-de-vida-del-software-seguros-s_19.html (Fecha de consulta: 06/06/2016).

[16] Ciclos de Vida del Software Seguros (S-SDLC) (Parte III). Flu-Project. http://www.flu-project.com/2014/05/ciclos-de-vida-del-software-seguros-s_27.html (Fecha de consulta: 06/06/2016).

[17] Ciclos de Vida del Software Seguros (S-SDLC) (Parte IV). Flu-Project. http://www.flu-project.com/2014/05/ciclos-de-vida-del-software-seguros-s_29.html (Fecha de consulta: 06/06/2016).

[18] 5 Tips To Reduce Risks From Modern Web Threats. Sophos. https://www.sophos.com/es-es/security-news-trends/whitepapers/five-tips-to-reduce-risk-from-modern-web-threats.aspx (Fecha de consulta: 05/06/2016).

[19] Imagen con derecho de autor: https://www.facilcloud.com/noticias/es_ES/desarrollo-front-end-vs-desarrollo-back-end/

[20] Imagen con derecho de autor: http://web.opalsoft.net/landing/webapps/index.html

[21] Imagen con derecho de autor: http://elmundomagicodeltaribo.blogspot.com.es/2009/04/sobre-front-end-back-end-y-mde.html

[22] Imagen con derecho de autor: http://www.flu-project.com/2014/05/ciclos-de-vida-del-software-seguros-s.html