UNIVERSIDAD INTERNACIONAL DE LA RIOJA

Seguridad en la Web

Metodologías de Desarrollo Web Seguro

La Universidad en Internet

Objetivos

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- Conocer los motivos para Desarrollar Seguro
- Conocer las Metodologías más extendidas
- Conocer cómo aplicar una metodología
- Referencias externas

Introducción (I)
Motivación

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

> "Today over 70% of attacks against a company's Web site or Web application come at the 'Application Layer' not the Network or System layer"
>
> *(Gartner Group)*

Introducción (II)
Definiciones (i)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Metodología de Desarrollo**
  - Conjunto de filosofías, fases, procedimientos, reglas, técnicas, herramientas, documentación y aspectos de formación para los desarrolladores de SI [Maddison,1983]
  - Conjunto de procedimientos, técnicas, herramientas y soporte documental que ayuda a los desarrolladores a realizar nuevo software.
  - Desarrollar software de **manera sistemática**.
  - Objetivos:
    - Mejores aplicaciones.
    - Un **mejor Proceso de Desarrollo** que identifique salidas (o productos intermedios) de cada fase de forma que se pueda planificar y controlar un proyecto.
    - Un **Proceso Estándar** en la organización.
- **Seguridad**
  - No es un producto, es un **proceso** [Bruce Schneier]

Introducción (IV)
Metodologías más extendidas

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- Las metodologías de desarrollo de software seguro más extendidas hoy dia:
  - **Microsoft**
    - Iniciativa *Trustworthy Computing*
  - **OSSTMM** *(Open Source Security Testing Methodology Manual)*
  - **OWASP** (*Open Web Application Security Project*)
  - OASIS Web Application Security (**WAS**) project

Metodología Microsoft (I)
Conceptos Generales (i)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- La iniciativa ***Trustworthy Computing***.

- Abarca varios conceptos:
  - **Confianza**
  - **Estabilidad**
  - **Seguridad en la plataforma**, que se sustenta sobre 5 pilares básicos:
    - Aislamiento y flexibilidad
    - Calidad
    - Autenticación
    - Autorización y control de accesos
    - Orientación y formación.

Metodología Microsoft (II)
Conceptos Generales (ii)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- Aquellos fabricantes que se acogen a esta iniciativa se comprometieron junto con Microsoft a conseguir cuatro principios en el de desarrollo de aplicaciones.

- La aplicaciones deben ser:
  - **Seguras**
    - Ningún virus atentará contra nuestros sistemas o los volverá inutilizables.
  - **Privadas**
    - La información personal no será expuesta de ninguna forma, ni utilizada de forma que no sea la explícitamente indicada.

Metodología Microsoft (III)
Conceptos Generales (iii)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- La aplicaciones deben ser (continuación…):
  - **Fiables**
    - Cuando instalemos un programa, no provocará efectos colaterales sobre otro software instalado.
  - Deben garantizar la **Integridad de negocio**
    - Mi proveedor de servicios responderá rápida y efectivamente cuando informo de un problema.
- Fruto de ello, surge el **Modelo de Amenazas** y sus recomendaciones.

Metodología Microsoft (IV)
Conceptos Generales (iv)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- La Seguridad debe abarcar todas las fases. Por tanto, el Software debe ser:
  - **Seguro por Diseño**
    - Ninguna parte de la aplicación queda fuera del control de seguridad.
  - **Seguro por Defecto**
    - La aplicación recién instalada tiene un comportamiento suficientemente seguro.
    - *Un ejemplo, es el gestor de áreas de exposición de SQL Server.*
  - **Seguro en la Distribución**
    - Informar al usuario sobre la seguridad de la aplicación
    - Mecanismos de modificación de las características de seguridad
    - Crear parches de seguridad tan pronto como se detecte una nueva vulnerabilidad.
  - **Seguro en las Comunicaciones**

Metodologia Microsoft (V)
Threat Model (i)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Modelo de Amenazas *(Threat Model)***
  - La metodología a seguir puede establecerse en 5 fases:
    - Identificar activos de la aplicación
    - Crear información general sobre la arquitectura
    - Descomponer la aplicación
    - Identificar, documentar y clasificar las amenazas
    - Identificar las vulnerabilidades
  - Debe contemplarse las siguientes amenazas (**Modelo STRIDE**):
    - **S**poofing (Suplantación)
    - **T**ampering (Manipulación)
    - **R**epudiation (No Repudio)
    - **I**nformation Disclosure (Divulgación de Información)
    - **D**enial of Service (Denegación de Servicio)
    - **E**levation of privileges (Elevación de privilegios)

Metodologia Microsoft (VI)
Threat Model (ii)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Modelo de Amenazas *(Threat Model)* (continuación…)**
  - Directiva **DREAD**:
    - **D**año Potencial
    - Facilidad de **R**eproducción
    - Capacidad de **E**xplotación
    - Usuarios **A**fectados
    - Dificultad para su **D**escubrimiento
  - Cálculo del Riesgo DREAD
    - DREAD se usa para formar parte del razonamiento detrás de la clasificación de riesgos, y sirve directamente para ordenar riesgos.
    - Cada Riesgo se calcula como un promedio de los cinco parámetros anteriores, **valorados de 0 a 10**.
    - **A mayor número, mayo riesgo**.
    - Fórmula de Riesgo DREAD:

$$\text{RiesgoDREAD} = (D + R + E + A + D) / 5$$

Metodologia Microsoft (VII)
Threat Model (iii)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Modelo de Amenazas *(Threat Model)* (cont.)**
  - Cálculo del Riesgo DREAD (continuación…):
    - **D**año Potencial: *¿Cuánto daño causa?*
      - 0: Nada
      - 5: La información individual del usuario se ve comprometida.
      - 10: Destrucción completa del sistema.
    - Facilidad de **R**eproducción: *¿La amenaza se puede reproducir con facilidad?*
      - 0: Muy difícil o imposible, incluso para los Administradores del sistema.
      - 5: En uno o dos pasos. Quizá requiera de usuario autorizado.
      - 10: Basta con una barra de direcciones y sin estar registrado en el sistema.
    - Capacidad de **E**xplotación: *¿Qué se necesita para explotar la amenaza?*
      - 0: Habilidades avanzadas de programación y redes, herramientas de ataque avanzadas o personalizadas.
      - 5: Malware existente, o fácilmente realizado utilizando herramientas normales de ataque.
      - 10: Solamente un navegador.

Metodologia Microsoft (VIII)
Threat Model (iv)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Modelo de Amenazas *(Threat Model)* (continuación…)**
  - Cálculo del Riesgo DREAD (continuación…):
    - Usuarios **A**fectados: *¿Cuántos usuarios se verán afectados por esta amenaza?*
      - 0: Ninguno
      - 5: Algunos usuarios, pero no todos.
      - 10: Todos los usuarios.
    - Dificultad para su **D**escubrimiento: *¿Es fácil descubrir la amenaza?*
      - 0: De muy difícil a imposible. Requiere acceso al sistema o al código.
      - 5: Se podría, mediante pruebas u observando la huellas en la red.
      - 9: Los detalles de errores de este tipo son de dominio público y se pueden descubrir con facilidad con cualquier buscador.
      - 10: La información es visible en el la barra de direcciones del navegador Web o en un formulario.

Metodologia Microsoft (VII)
Threat Model (iii)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Modelo de Amenazas *(Threat Model)*** (cont.)
  - Recomendaciones para la defensa:
    - Adoptar el *principio mínimo privilegio*
    - Usar las defensas en profundidad
    - No confiar en los datos introducidos por el usuario
    - Utilizar opciones predeterminadas seguras
    - No depender de la seguridad por medio de ocultación
    - Validar todo acceso al sistema
    - Asumir que los sistemas externos no son seguros
    - Reducir el área de exposición
    - Cometer errores de forma segura
    - No olvidar que el alcance de la seguridad lo define su punto más débil
    - Si no se utiliza, hay deshabilitarlo

OSSTMM (I)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- ***Open Source Security Testing Methodology Manual*** (Manual de Metodología de Pruebas de Seguridad de Código Abierto)

- El Manual de la Metodología Abierta de Comprobación de la Seguridad es uno de los estándares de facto mas utilizado en Auditorías de Seguridad para revisar la Seguridad de los Sistemas desde Internet.

- Incluye un marco de trabajo que describe las fases que habría que realizar para la ejecución de la auditoría.

OSSTMM (II)

- Se encuentra en constante evolución y se compone de las siguientes fases:
  - Seguridad de la Información
  - Seguridad de los Procesos
  - Seguridad en las Tecnologías de Internet
  - Seguridad en las Comunicaciones
  - Seguridad Inalámbrica
  - Seguridad Física

OWASP (I)
Introducción (i)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- El proyecto **OWASP** por sus siglas en inglés ***Open Web Application Security Project.***

- Tiene como **objetivo** ofrecer una **metodología**:
  - De libre acceso y utilización.
  - Que pueda ser utilizada como material de referencia por parte de los arquitectos de software, desarrolladores, fabricantes y profesionales de la seguridad.
  - Todos ellos involucrados en el diseño, desarrollo, despliegue y verificación de la seguridad de las aplicaciones y servicios web.

OWASP (II)
Introducción (ii)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- Organizado en **proyectos y capítulos locales** repartidos por todo el mundo, se desarrollan documentaciones, herramientas y estándares de fuentes abiertas (*GPL, GFDL, LGPL*). Cualquier persona y/o patrocinador puede participar.

- Pretende que se desarrolle **mejor Software** mediante:
  - El desarrollo de herramientas útiles para identificar los fallos y corregirlos.
  - La educación de los grupos involucrados, para evitar que se produzcan fallos.
  - El fomento de la discusión de problemas a través de una comunidad abierta.
  - La definición de estándares.

OWASP (III)
Introducción (iii)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- Patrocinadores principales:

- Proyectos más destacados:
  - OWASP **Top Ten Project**
    - Representa un consenso a nivel global sobre las 10 vulnerabilidades web más importantes.
  - **WebGoat**
    - Herramienta destinada a la educación y que permite practicar y explotar las vulnerabilidades mas frecuentes de un sitio Web, con el fin de poner en práctica una metodología de desarrollo seguro.
  - **WebScarab**
    - Es un framework para el análisis de aplicaciones que utilizan como base los protocolos HTTP y HTTPS. Estás escrito en Java y es multiplataforma.

OWASP (IV)
Introducción (iv)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- Guías más relevantes:
  - **Development Guide**
    - Guía para la construcción de aplicaciones Web seguras
    - Última versión completa 2.0.1 (Español e Inglés) (julio 2005)
    - Versión en desarrollo v3
  - **Code Review Guide**
    - Guía para la revisión de código para la garantía de software seguro
    - Última versión 2.0 (Enero 2011)
  - **Testing Guide**
    - Guía y herramientas para pruebas de intrusiones y garantía de software seguro.
    - Última versión 3.0 (diciembre 2008)
    - Versión en desarrollo 4.0

OWASP (V)
Principios básicos (i)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- Principios básicos de la seguridad de cualquier aplicación o servicio Web:
  - **Validación de la entrada y salida de información**
    - La entrada y salida de información es el principal mecanismo que dispone un atacante para enviar o recibir código malicioso contra el sistema.
    - Siempre debe verificarse que cualquier dato entrante o saliente es apropiado y en el formato que se espera.
    - Las características de estos datos deben estar predefinidas y debe verificarse en todas las ocasiones.
  - **Diseños simples**
    - Los mecanismos de seguridad deben diseñarse para que sean los más sencillos posibles, huyendo de sofisticaciones que compliquen excesivamente la vida a los usuarios.
    - Si los pasos necesarios para proteger de forma adecuada una función o modulo son muy complejos, la probabilidad de que estos pasos no se ejecuten de forma adecuada es muy elevada.

OWASP (VI)
Principios básicos (ii)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- Principios básicos de la seguridad (continuación….):
  - **Utilización y reutilización de componentes de confianza**
    - Debe evitarse "reinventar la rueda" constantemente.
    - Cuando exista un componente que resuelva un problema de forma correcta, lo más inteligente es utilizarlo.
  - **Defensa en profundidad**
    - Nunca confiar en que un componente realizará su función de forma permanente y ante cualquier situación.
    - Hemos de disponer de los mecanismos de seguridad suficientes para que cuando un componente del sistema fallen ante un determinado evento, otros sean capaces de detectarlo.

OWASP (VII)
Principios básicos (iii)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- Principios básicos de la seguridad (continuación….):
  - **Verificación de privilegios**
    - Los sistemas deben diseñarse para que funcionen con los menos privilegios posibles.
    - Igualmente, es importante que los procesos únicamente dispongan de los privilegios necesarios para desarrollar su función, de forma que queden compartimentados.
  - **Ofrecer la mínima información**
    - Ante una situación de error o una validación negativa, los mecanismos de seguridad deben diseñarse para que faciliten la mínima información posible.
    - De la misma forma, estos mecanismos deben estar diseñados para que una vez denegada una operación, cualquier operación posterior sea igualmente denegada.

OWASP (VIII)
Principios básicos (iv)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- Principios básicos de la seguridad (continuación….):
  - **Otros consideraciones**
    - De arquitectura.
    - Mecanismos de autenticación.
    - Gestión de sesiones de usuario.
    - Control de acceso.
    - Registro de actividad.
    - Consideraciones de privacidad y criptografía.

OWASP (IX)
Guía de Desarrollo (i)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- Para garantizar el desarrollo seguro de software, la Guía de Desarrollo OWASP proporciona una pautas para salvaguardar de amenazas en las aplicaciones Web en los siguientes elementos:
  - Manejo de Pagos en el Comercio Electrónico
  - Phising
  - Servicios Web
  - Autenticación
  - Autorización
  - Manejo de Sesiones
  - Validación de Datos
  - Intérprete de Inyección
  - Canonicalización, Locales y Unicode

OWASP (X)
Guía de Desarrollo (ii)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- Elementos de la Guía de Desarrollo (continuación…):
  - Manejo de Errores, Auditoria y Generación de Logs
  - Sistemas de Ficheros
  - Desbordamientos de Memoria
  - Interfaces Administrativas
  - Cifrado
  - Configuración
  - Mantenimiento
  - Ataques de Denegación de Servicio
  - Licencia de Documentación Libre de GNU
  - Directivas sobre PHP

OWASP (XI)
Guía de Desarrollo (iii)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Manejo de Pagos en el Comercio Electrónico**
  - Objetivos
    - Manejar los pagos de una manera segura y equitativa de los usuarios de sistemas de comercio electrónico.
    - Minimizar el fraude de los usuarios de tarjetas en pagos no presenciales. (*CNP – Card Not Present*)
    - Maximizar la privacidad y confianza para los usuarios de sistemas de comercio electrónico.
    - Cumplir con todas las leyes locales y normas PCI (*Pay Card Industry*)

OWASP (XII)
Guía de Desarrollo (iv)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Manejo de Pagos en el Comercio Electrónico (continuación…)**
  - Buenas prácticas
    - Procese las transacciones online inmediatamente o pase el procesamiento a una tercera parte competente.
    - Nunca almacene ningún número de tarjeta de crédito (CC). Si deben almacenarse, debe seguir las directivas de PCI al pie de la letra. Se recomienda encarecidamente que no almacene datos de tarjetas de crédito.
    - Si se usa un servidor compartido para su sitio, no puede cumplir con las directivas PCI. Debe tener su propia infraestructura para cumplir con las directivas PCI.

OWASP (XIII)
Guía de Desarrollo (v)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Phising**
  - El *phishing* es una tergiversación donde el criminal utiliza ingeniería social para aparecer como una identidad legitima.
  - Los ataques de *phishing* son uno de los mayores problemas para los sitios bancarios y de comercio electrónico, con el potencial de destruir los medios de subsistencia y calificaciones crediticias de un cliente.
  - Hasta un 5% de los usuarios parecen ser atraídos en este tipo de ataques.

OWASP (XIV)
Guía de Desarrollo (vi)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Phising (continuación…)**
  - Pautas para evitar el problema del *Phising* en el desarrollo de aplicaciones Web:
    - Educación del usuario.
    - Haga fácil a sus usuarios informar de estafas.
    - Informe a los clientes a través de correo electrónico de lo siguiente:
      - Deben escribir la URL en sus navegadores para acceder su sitio.
      - Usted nunca proporciona enlaces para que ellos hagan clic.
      - Usted nunca preguntara por sus datos confidenciales.
      - Y en el caso que los usuarios reciban tales mensajes, deberán comunicarse inmediatamente con usted para informar a las autoridades competentes.
    - Nunca solicitar información secreta a sus clientes.
    - Solucionar todos los problemas de XSS.
    - No utilice ventanas emergentes.
    - No utilice frames (*frames* ni *iframes*).
    - Mueva su aplicación a un enlace de distancia de su página principal.

OWASP (XV)
Guía de Desarrollo (vii)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Phising (continuación…)**
  - Pautas para evitar el problema del Phising (continuación…):
    - Imponga el uso de referencias locales para imágenes y otros recursos.
    - Mantenga la barra de direcciones, utilice SSL, no utilice direcciones IP
    - No sea la fuente de robos de identidad.
    - Implemente protecciones dentro de su aplicación.
    - Monitorice actividad inusual en las cuentas.
    - Tome control de los nombres de dominio fraudulentos.
    - Trabaje con las autoridades competentes
    - Sea amable con su cliente cuando ocurre un ataque – él es la víctima.

OWASP (XVI)
Guía de Desarrollo (viii)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Autenticación**
  - Objetivo:
    - Proveer servicios de autenticación segura a las aplicaciones Web, mediante:
      - Vinculando una unidad del sistema a un usuario individual mediante el uso de una credencial
      - Proveyendo controles de autenticación razonables de acuerdo al riesgo de la aplicación.
      - Denegando el acceso a atacantes que usan varios métodos para atacar el sistema de autenticación.

OWASP (XVII)
Guía de Desarrollo (ix)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Autenticación (continuación…)**
  - Buenas prácticas:
    - La autenticación es solo tan fuerte como los procesos de administración de usuarios a los que afecte dicha autenticación.
    - Use la forma más apropiada de autenticación para su clasificación de recursos.
    - Re-autenticar al usuario para transacciones de alto valor y acceso a áreas protegidas.
    - Autenticar la transacción, no el usuario.
    - Las contraseñas son un mecanismo débil por sí sólo y no son adecuadas para sistemas de alto valor.

OWASP (XVIII)
Guía de Desarrollo (xix)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Autorización**
  - Objetivos:
    - Asegurar que únicamente usuarios autorizados puedan realizar acciones permitidas con su correspondiente nivel de privilegio.
    - Controlar el acceso a recursos protegidos mediante decisiones basadas en el rol o el nivel de privilegio.
    - Prevenir ataques de escalada de privilegios, como por ejemplo utilizar funciones de administrativas siendo un usuario anónimo o incluso un usuario autenticado.
  - Prácticas para garantizar la autorización:
    - Principio de mínimo privilegio
    - Listas de Control de Acceso
    - Controles de autorización personalizados
    - Rutinas de autorización centralizadas
    - Matriz de autorización
    - Control y Protección de acceso a recursos

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Manejo de Sesiones**
  - Objetivos:
    - Asegurarse de que los usuarios autenticados tengan una asociación con sus sesiones robusta y criptográficamente segura.
    - Garantizar que se hagan cumplir los controles de autorización.
    - Se tienen que prevenir los típicos ataques web, tales como la reutilización, falsificación e interceptación de sesiones.

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Manejo de Sesiones (continuación…)**
  - Buenas prácticas:
    - Datos sobre autorización y roles deben ser guardados solamente del lado del servidor.
    - Datos sobre la navegación son ciertamente aceptables en la URL siempre y cuando los controles de validación y autorización sean efectivos.
    - Las preferencias del usuario (ej. temas y lenguaje del usuario) puede ser almacenado en cookies.
    - Datos de formularios no deberían contener campos ocultos – si se encuentran ocultos, probablemente necesiten estar protegidos y solo disponibles del lado del servidor. Sin embargo, los campos ocultos pueden (y deben) ser utilizados para la protección de secuencias y ataques de Pharming.

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Manejo de Sesiones (continuación…)**
  - Buenas prácticas (continuación…):
    - Los datos contenidos en formularios de varias páginas pueden ser enviados de vuelta al usuario en los siguientes dos casos:
      - Cuando existen controles de integridad para prevenir la manipulación.
      - Cuando los datos son validados después de cada envío del formulario, o al menos al final del proceso de envío.

OWASP (XXII)
Guía de Desarrollo (xxii)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Validación de Datos**
  - La debilidad de seguridad más común en aplicaciones web es la falta de validación apropiada de las entradas del cliente o del entorno.
    - Esta debilidad lleva a casi todas las principales vulnerabilidades en las aplicaciones, tales como intérprete de inyección, ataques Locale/Unicode, ataques al sistema de archivos y desbordamientos de memoria.
  - Nunca se debe confiar en los datos introducidos por el cliente, ya que tiene todas las posibilidades de manipular los datos.

  - Objetivo
    - Garantizar que la aplicación sea robusta contra todas las formas de ingreso de datos, ya sea obtenida del usuario, de la infraestructura, de entidades externas o de sistemas de base de datos.

OWASP (XXIII)
Guía de Desarrollo (xxiii)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Validación de Datos (continuación…)**
  - Contramedidas
    - **Revisiones de integridad**
      - Aseguran que los datos no han sido manipulados y que siguen siendo los mismos.
      - Las revisiones de integridad deben ser incluidas en cualquier lugar en que los datos pasen de una frontera confiable a una menos confiable.
        - Por ejemplo, en la aplicación al navegador del usuario en un campo oculto, o hacia un método de pago ofrecido por terceros, tal como un identificador utilizado internamente a su regreso.
      - El tipo de control de integridad (*checksum, HMAC, encriptación, firma digital*) se debe seleccionar en relación directa con el riesgo que representa la transición de los datos a través de una frontera confiable.

OWASP (XXIV)
Guía de Desarrollo (xxiv)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Validación de Datos (continuación…)**
  - Contramedidas (continuación…)
    - **Validación**
      - Asegura que los datos están sólidamente escritos, con la sintaxis correcta, dentro de los limites de longitud, que contenga solo caracteres permitidos, si es numérico que tenga el signo correcto y dentro de los limites del rango.
      - La validación debe ser llevada a cabo en cada capa de la aplicación. Sin embargo, la validación debería llevarse a cabo en función del servidor que esta ejecutando el código.
        - Por ejemplo, la capa de web/presentación debe validar problemas relacionados con la web, las capas de persistencia deberían validar problemas de persistencia tales como inyección de SQL/HQL;
        - las operaciones de búsqueda en directorio deberían revisar inyección de LDAP y así sucesivamente.

OWASP (XXV)
Guía de Desarrollo (xxv)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Validación de Datos (continuación…)**
  - Contramedidas (continuación…)
    - **Validación de reglas de negocio**
      - Garantizan que los datos no solamente sean validos, sino que cumplas con las reglas denegocio.
        - Por ejemplo, las tasas de interés entran dentro de los límites permitidos..
      - Las reglas de negocio se conocen durante el diseño e influyen durante la implementación. Sin embargo, hay enfoques malos, buenos y “mejores”.
      - Frecuentemente el mejor enfoque es el más simple en términos de código.

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Validación de Datos (continuación…)**
  - Contramedidas (continuación…)
    - **Ejemplo – Escenario (Validación de reglas de negocio)**
      - Usted va a llenar una lista con cuentas proporcionada por el *backend* del sistema:
      - El usuario seleccionara una cuenta, selecciona un vendedor y presiona siguiente.
      - Solución incorrecta: La opción seleccionar cuenta es leída directamente y proporcionada en un mensaje de regreso al sistema *backend* sin validar si el numero de cuenta es una de las cuentas proporcionadas por sistema de *backend*.
      - Un atacante puede cambiar el código HTML de cualquier manera que elija:
        - La carencia de validación requiere un viaje de vuelta al *backend* para proveer un mensaje de error que el código de la interfaz de usuario podría fácilmente haber eliminado.
        - El *backend* puede ser incapaz de enfrentarse a la carga útil de datos que la interfaz del usuario fácilmente podría haber eliminado. Por ejemplo desbordamientos de memoria, inyección de XML o similares.

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Validación de Datos (continuación…)**
  - Contramedidas (continuación…)
    - **Ejemplo – Escenario (Validación de reglas de negocio)**
      - Solución aceptable: La opción de seleccionar el parámetro cuenta se lee por el código, y comparado con la lista previamente desplegada.

```
if ( account.inList(session.getParameter('payeelstid') ) {
            backend.performTransfer(session.getParameter('payeelstid'));
}
```

        - Esto evita la manipulación de parámetros, pero todavía hace que el navegador haga mucho trabajo.

      - La mejor solución: El código original mostraba índices `<option value="1" … >` en vez de nombres de cuenta.

```
int payeeLstId = session.getParameter('payeelstid');
accountFrom = account.getAcctNumberByIndex(payeeLstId);
```

        - Esto no solo es más fácil que desplegar HTML, sino que hace trivial la validación y las reglas de negocio. El campo no puede ser manipulado.

OWASP (XXVIII)
Guía de Pruebas (i)

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- Proporciona pautas sobre qué pruebas conviene hacer para garantizar la seguridad del software y cómo realizarlas de forma sistemática a lo largo de vida del desarrollo del software.

- La guía está **orientada principalmente a Aplicaciones Web**.

- Proporciona dividida conceptualmente en dos grandes partes:
  - **Marco de trabajo** de pruebas en el Ciclo de Vida del Desarrollo del Software (*SDLC Testing Framework*)
  - **Vulnerabilidades** más comunes en Aplicaciones Web

- Es una guía y no documento formal que seguir al "pie de la letra".

OWASP (XXIX)

Guía de Pruebas (ii) – Framework de Pruebas

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- Describe un marco de pruebas típico que puede ser desarrollado en una organización.
- Hay que verla como un marco de referencia que comprende tanto técnicas como tareas que es apropiado realizar en varias fases del ciclo de vida de desarrollo del software (SDLC)
- El **Framework de Pruebas de OWASP** está estructurado de la siguiente forma:
  - Fase 1: Antes de Empezar el Desarrollo
    - 1.a: Revisión de Estándares y Políticas
    - 1.b: Desarrollo de Métricas y Criterios de Medición (Asegurar la Trazabilidad)
  - Fase 2: Durante el Diseño y Definición
    - 2.a: Revisión de los Requisitos de Seguridad
    - 2.b: Diseño de una Arquitectura de Revisión
    - 2.c: Creación y Revisión de Modelos UML
    - 2.d: Creación y Revisión de Modelos de Amenaza

OWASP (XXX)

Guía de Pruebas (iii) – Framework de Pruebas

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- El **Framework de Pruebas de OWASP**: (continuación…)
  - Fase 3: Durante el Desarrollo
    - 3.a: Inspección de Código por Fases
    - 3.b: Revisiones de Código
  - Fase 4: Durante la Implementación
    - 4.a: *Testing* de Penetración de Aplicaciones
    - 4.b: *Testing* de Gestión de Configuraciones
  - Fase 5: Mantenimiento y Operación
    - 5.a: Ejecución de Revisiones de la Gestión Operativa
    - 5.b: Ejecución de Comprobaciones Periódicas de Mantenimiento
    - 5.c: Asegurar la Verificación de Cambios
  - Flujo de Pruebas típico en un SDLC

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Fase 1: Antes de Empezar el Desarrollo**
  - Comprobación para asegurar que existe un SDLC adecuado, en el cual la seguridad sea inherente.
  - Comprobación para asegurar que están implementados la política y estándares de seguridad adecuados para el equipo de desarrollo.
  - Desarrollar las métricas y criterios de medición.

  - **1.a: Revisión de Estándares y Políticas**
    - Asegurar que las políticas, documentación y estándares adecuados están implementados.
    - *Las personas pueden hacer las cosas correctamente, solo si saben que es lo correcto.*
  - **1.b: Desarrollo de Métricas y Criterios de Medición (Asegurar la Trazabilidad)**
    - Antes de empezar el desarrollo, planifica el programa de medición.
    - Definir los criterios que deben ser medidos proporciona visibilidad de los defectos tanto en el proceso como en el producto

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Fase 2: Durante el Diseño y Definición**
  - **2.a: Revisión de los Requisitos de Seguridad**
    - Los requisitos de seguridad definen cómo funciona una aplicación desde una perspectiva de la seguridad.
    - Es indispensable que los requisitos de seguridad sean probados.
    - A la hora de buscar inconsistencias en los requisitos, ten en cuenta mecanismos de seguridad como:
      - *Gestión de Usuarios, Autenticación , Autorización, Confidencialidad de los Datos, Integridad, Contabilidad, Gestión de Sesiones, Seguridad de Transporte, Segregación de Sistemas en Niveles, Privacidad .*
  - **2.b: Diseño de una Arquitectura de Revisión**
    - Las aplicaciones deberían tener una arquitectura y diseño documentados.
    - Identificar fallos de seguridad en la fase de diseño no es solo una de las fases más efectiva por costes a la hora de identificar errores, sino que también puede ser la fase más efectiva para realizar cambios.

OWASP (XXXIII)

Guía de Pruebas (vi) – Framework de Pruebas

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Fase 2: Durante el Diseño y Definición (cont.)**
  - **2.c: Creación y Revisión de Modelos UML**
    - Una vez completados el diseño y arquitectura, construye modelos UML que describan cómo funciona la aplicación.
    - Emplea estos modelos para confirmar junto a los diseñadores de sistemas una comprensión exacta de como funciona la aplicación.
    - Si se descubre alguna vulnerabilidad, debería serle transmitida al arquitecto del sistema para buscar aproximaciones alternativas.
  - **2.d: Creación y Revisión de Modelos de Amenaza**
    - Desarrolla escenarios de amenaza realistas.
    - Analiza el diseño y la arquitectura para asegurarte de que esas amenazas son mitigadas, aceptadas por negocio, o asignadas a terceros (*como puede ser una aseguradora)*.
    - Cuando las amenazas identificadas no tienen estrategias de mitigación, revisa el diseño y la arquitectura con los arquitectos de los sistemas para modificar el diseño.

OWASP (XXXIV)

Guía de Pruebas (vii) – Framework de Pruebas

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Fase 3: Durante el Desarrollo**
  - **3.a: Inspección de Código por Fases**
    - El equipo de seguridad debería realizar una inspección del código por fases con los desarrolladores y, en algunos casos, con los arquitectos del sistema.
    - El propósito de una inspección es entender el flujo de programación a alto nivel, su esquema y la estructura del código que conforma la aplicación.
  - **3.b: […]**

OWASP (XXXV)

Guía de Pruebas (viii) – Framework de Pruebas

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Fase 3: Durante el Desarrollo**
  - **3.a: […]**
  - **3.b: Revisiones de Código**
    - El probador puede examinar ahora el código real en busca de defectos de seguridad.
    - Las revisiones de código estático validan el código contra una serie de listas de comprobación, que incluyen:
      - Requisitos de negocio de disponibilidad, confidencialidad e integridad.
      - Guía del OWASP o Lista de Comprobación de los Top 10 de exposición técnica.
      - Incidencias específicas relativas al lenguaje o marco de trabajo en uso, como el *Scarlet paper* para PHP o las *Microsoft Secure Coding checklists* para ASP.NET.
      - Cualquier requisito específico de la industria, como *Sarbanes-Oxley 404, COPPA, ISO 17799, APRA, HIPAA, Visa Merchant guidelines* o cualquier otro.

OWASP (XXXVI)

Guía de Pruebas (ix) – Framework de Pruebas

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Fase 4: Durante la Implementación**
  - **4.a: *Testing* de Penetración de Aplicaciones**
    - Tras haber comprobado los requisitos, analizado el diseño y realizado la revisión de código, cabría esperar que se hayan identificado todas las incidencias. Pero no hay que darlo por hecho.
    - El *testing* de penetración de la aplicación después de que haya sido implementada nos proporciona una última comprobación.
  - **4.b: *Testing* de Gestión de Configuraciones**
    - El test de penetración de la aplicación debería incluir la comprobación de como se implementó y securizó su infraestructura.
      - Aunque la aplicación puede ser segura, un pequeño detalle de la configuración podría estar en una etapa de instalación por defecto, y ser vulnerable a explotación.

OWASP (XXXVII)

Guía de Pruebas (x) – Framework de Pruebas

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Fase 5: Mantenimiento y Operación**
  - **5.a: *Ejecución de Revisiones de la Gestión Operativa***
    - Debe existir un proceso que detalle como es gestionada la sección operativa de la aplicación y su infraestructura
  - **5.b: Ejecución de Comprobaciones Periódicas de Mantenimiento**
    - Deberían realizarse comprobaciones de mantenimiento mensuales o trimestrales, sobre la aplicación e infraestructura, para asegurar que no se han introducido nuevos riesgos de seguridad y que el nivel de seguridad sigue intacto.
  - **5.c: Asegurar la Verificación de Cambios**
    - Es vital que como parte del proceso de gestión de cambios, el cambio sea comprobado para asegurar que el nivel de seguridad no haya sido afectado por dicho cambio.

OWASP (XXXVIII)

Guía de Pruebas (xi) – Vulnerabilidades

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- La Guía de Pruebas de OWASP contempla y analiza en detalle las pruebas que hay que realizar para los siguientes **grupos de vulnerabilidades**:
  - Recolección de Información
    - Spiders, Robots y Crawlers, Descubrimiento con motor de búsquedas, Puntos de entrada de aplicación, Fingerprint de aplicaciones Web, Descubrimiento de aplicación, Análisis de Códigos de Error.
  - Pruebas de Gestión de Configuración
    - SSL/TLS, Listeners de Bases de Datos, Gestión de Configuración de Infraestructura, Gestión de Configuración de Aplicación, Manejo de extensiones de archivo, ficheros viejos, de backupo o sin referenciar, interfaces de administración, XST y métodos HTTP, etc.
  - Pruebas de Autenticación
    - Transporte de credenciales mediante un canal cifrado, enumeración de usuarios, cuentas de usuario predeterminadas, fuerza bruta, esquema de puenteo de autenticación, vulnerabilidades de recordatorio de passwords, etc.

OWASP (XXXIX)

Guía de Pruebas (xii) – Vulnerabilidades

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Grupos de vulnerabilidades (continuación…)**:
  - Pruebas de Gestión de Sesiones
    - Esquema de gestión de sesiones, cookies, fijación de sesión, variables de sesión expuestas, etc.
  - Pruebas de Autorización
    - Puenteo del esquema de autorización, elevación de privilegios, lógica de negocio, etc.
  - Pruebas de Validación de Datos
    - Cross Site Scriptting (en sus diferentes modalidades), Cross Site Flashing, SQL Injection, pruebas de bases de datos (MySQL, SQL Server, Oracle, Access, PostgreSQL), Injection (LDAP, ORM, XML, SSI, Xpath, IMAP/SMTP, Code), Comandos de Sistema Operativo, etc.

OWASP (XL)

Guía de Pruebas (xiii) – Vulnerabilidades

UNIVERSIDAD INTERNACIONAL DE LA RIOJA

- **Grupos de vulnerabilidades (continuación…)**:
  - Pruebas de Denegación de Servicio
    - SQL Wildcards, Bloqueo de cuentas de usuario, Desbordamientos de buffers, fallo de liberación de recursos, etc.
  - Pruebas de Servicios Web
    - WSDL, XML, parámetros GET, pruebas de respuesta, SOAP, etc.
  - Pruebas de AJAX

Referencias

- **Microsoft Trustworthy Computing,** http://www.microsoft.com/mscorp/twc/default.mspx
- **Modelo de Amenazas de Microsoft (Microsoft Threat Modeling (STRIDE and DREAD),** http://msdn.microsoft.com/en-us/library/aa302419.aspx
- **Open Web Application Security Project OWASP**, http://www.owasp.org
- **Open Source Security Testing Methodology Manual (OSSTMM)**, http://www.osstmm.org