OWASP Top 10 – 2010

Los diez riesgos más importantes en aplicaciones web

Fabio Cerullo
Comité Global de Educación
OWASP
fcerullo@owasp.org

The OWASP Foundation
http://www.owasp.org/

¿Quién soy?

- Especialista en Seguridad de la Información en Allied Irish Banks.
- Ingeniero en Informática de la Universidad Católica Argentina.
- Certified Information Security Systems Professional (CISSP)
- Miembro del Comité Global de Educación OWASP.
- Presidente del Capitulo Irlandés de OWASP.

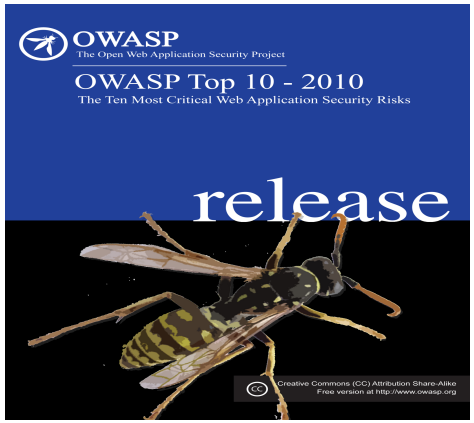
Misión del Comité Global de Educación OWASP: *Proporcionar sensibilización, capacitación y servicios educativos a las empresas, instituciones gubernamentales y educativas sobre seguridad en aplicaciones.*

Introducción al Top 10

- Es un documento EDUCATIVO.
- Es GRATUITO.
- DESCRIBE los riesgos más críticos en aplicaciones Web
- Versión 2010 próximamente en español

Para cada riesgo, aporta:

- Descripción del mismo
- Escenario de ejemplo de un ataque
- Pautas para verificar si nuestra aplicación es vulnerable
- Recomendaciones para prevenir dicho riesgo



¿Que ha cambiado en esta versión?

Se centra en los Riesgos, no solo en Vulnerabilidades

Su nuevo titulo es: "Los diez riesgos más importantes en aplicaciones web 2010"

Ranking de Riesgos en OWASP Top 10

Se basa en la metodologia de catalogacion de riesgos OWASP, utilizada para priorizar el Top 10

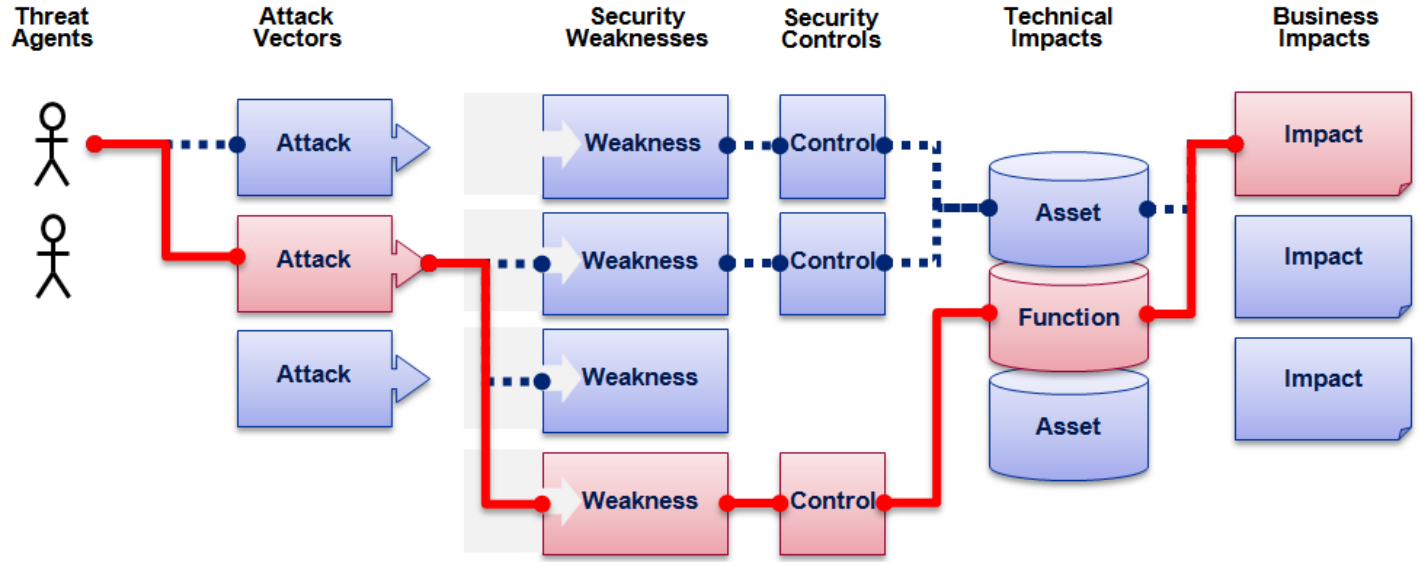
Riesgos agregados y eliminados

- Agregado: A6 – Configuración Defectuosa de Seguridad
  - Se encontraba en el Top 10 2004: Gestión Insegura de la Configuración
- Agregado: A10 – Redirecciones y Destinos no Validados
  - Falla relativamente común y MUY peligrosa que no es bien conocida
- Eliminado: A3 – Ejecución de Ficheros Malintencionados
  - Principalmente una falla en PHP que esta perdiendo relevancia
- Eliminado: A6 – Revelación de Información y Gestión Incorrecta de Errores
  - Falla muy prevalente, que no introduce demasiado riesgo (normalmente)

Comparación del Top 10 2010 con 2007

OWASP Top 10 – 2007 (Anterior)		OWASP Top 10 – 2010 (Nuevo)
A2 – Fallas de Inyección	↑	A1 – Inyección
A1 – Secuencia de Comandos en Sitios Cruzados (XSS)	↓	A2 – Secuencia de Comandos en Sitios Cruzados (XSS)
A7 – Perdida de Autenticación y Gestión de Sesiones	↑	A3 – Perdida de Autenticación y Gestión de Sesiones
A4 – Referencia Directa Insegura a Objetos	=	A4 – Referencia Directa Insegura a Objetos
A5 – Falsificación de Petición en Sitios Cruzados (CSRF)	=	A5 – Falsificación de Petición en Sitios Cruzados (CSRF)
<era T10 2004 A10 – Gestión Insegura de la Configuración>	+	A6 – Configuración Defectuosa de Seguridad (NUEVO)
A8 – Almacenamiento Criptográfico Inseguro	↑	A7 – Almacenamiento Criptográfico Inseguro
A10 – Falla de restricción de acceso a URL	↑	A8 – Falla de restricción de acceso a URL
A9 – Comunicaciones Inseguras	=	A9 – Protección Insuficiente en la Capa de Transporte
<no disponible en T10 2007>	+	A10 – Redirecciones y Destinos No Validados (NUEVO)
A3 – Ejecución de Ficheros Malintencionados	-	<eliminado del T10 2010>
A6 – Revelación de Información y Gestión Incorrecta de Errores	-	<eliminado del T10 2010>

Metodología de Catalogación de Riesgo OWASP



Agente de Amenaza	Vector de Ataque	Prevalencia de Debilidad	Detectabilidad de Debilidad	Impacto Tecnico	Impacto al Negocio
	1 Facil	Amplia	Facil	Severo	
?	2 Mediano	Comun	Mediano	Moderado	?
	3 Dificil	Poco Comun	Dificil	Menor	
	1	2	2	1	

Ejemplo de Inyección

1.66 * 1

1.66 calificación de riesgo ponderado

OWASP Top Ten (Edición 2010)

OWASP
The Open Web Application Security Project
http://www.owasp.org

http://www.owasp.org/index.php/Top_10

A1 – Inyección

Inyección significa…

- Incluir comandos mal intencionados en los datos de una aplicación los cuales son enviados a un interprete

Los interpretes…

- Toman estos datos y los interpretan como comandos validos
- SQL, OS Shell, LDAP, XPath, Hibernate, etc…

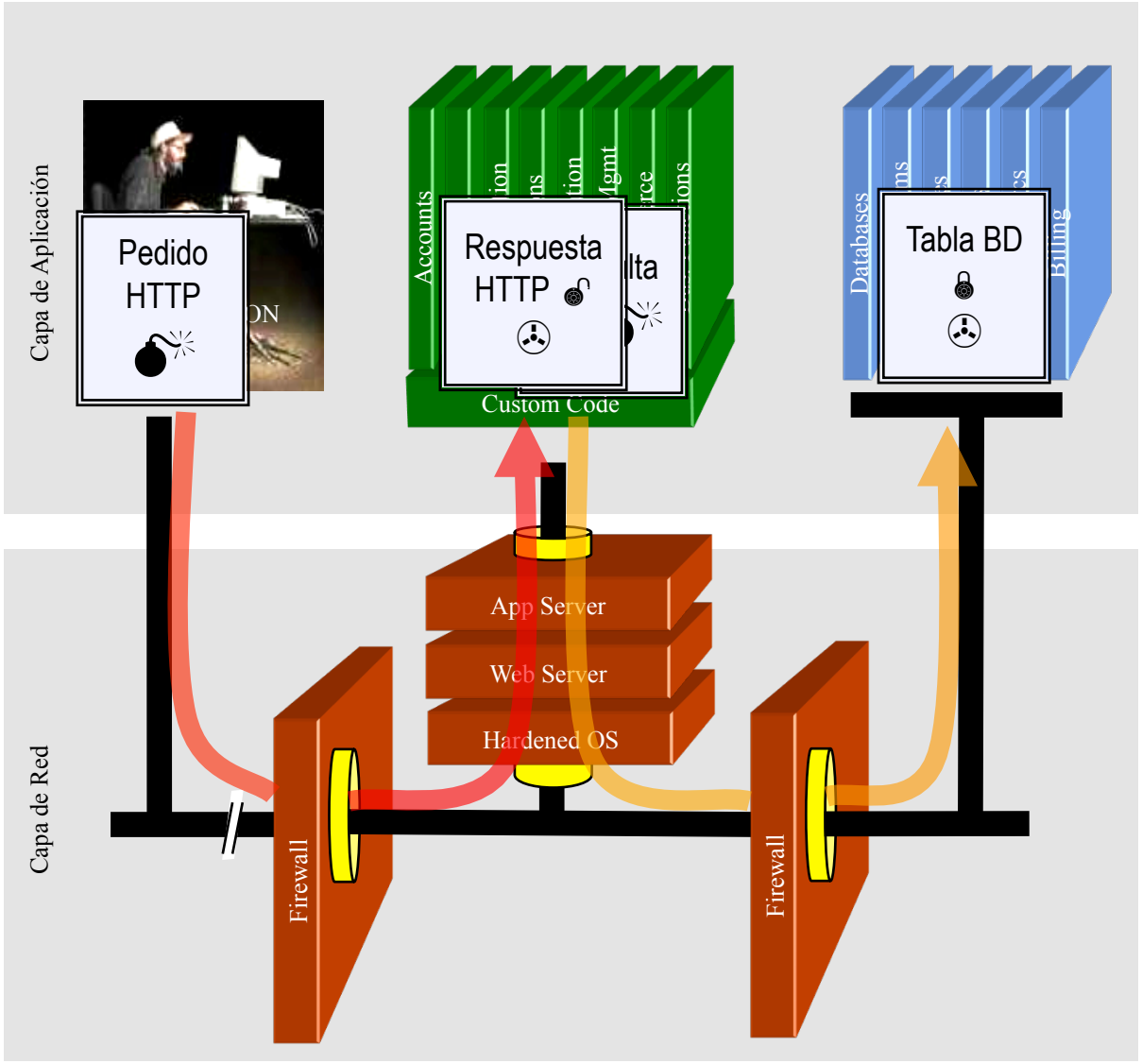
La Inyección SQL es aun bastante frecuente

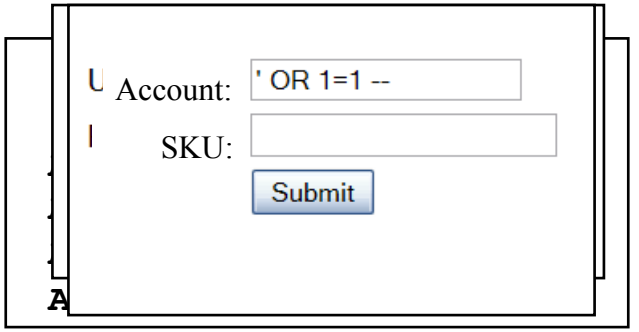
- Muchas aplicaciones todavía son susceptibles (no tendría que ser así)
- Por lo general es muy fácil de evitar

Impacto Típico

- Por lo general severo. Todos los contenidos de una base de datos pueden potencialmente ser leídos o modificados.
- También puede permitir el completo acceso al esquema de la base de datos, o cuentas de usuario, o incluso a nivel del sistema operativo

Inyección SQL – Demostración





1. Aplicación presenta un formulario web al atacante
2. Atacante envía un ataque en los datos del formulario
3. Aplicación dirige el ataque a la base de datos en una consulta SQL
4. Base de datos ejecuta el ataque y envía los resultados cifrados nuevamente a la aplicación
5. Aplicación descifra los datos normalmente y envía los resultados al atacante

A1 – Como evitar Fallas de Inyección

- Recomendaciones
  1. Evitar el interprete completamente
  2. Utilizar una interfase que soporte variables parametrizadas (Ej., declaraciones preparadas, o procedimientos almacenados),
     - Las variables parametrizadas permiten al interprete distinguir entre código y datos
  3. Decodificar y convertir todas las entradas del usuario a su forma mas simple antes de enviarlas al interprete
  - Siempre efectuar una validación 'positiva' de todas las entradas realizadas por el usuario
  - Seguir el principio de mínimo privilegio en las conexiones con bases de datos para reducir el impacto de una falla

- Referencias
  - Para mayor información:
    http://www.owasp.org/index.php/SQL_Injection_Prevention_Cheat_Sheet

A2 – Secuencia de Comandos en Sitios Cruzados (XSS)

Ocurre cada vez que…

- Datos no validados de un atacante son enviados al navegador de una victima

Los datos no validados pueden…

- Encontrarse ALMACENADOS en una base de datos
- Ser REFLEJADOS desde una entrada web (formulario, campo oculto, URL, etc…)
- Ser enviados directamente a un cliente JavaScript

Virtualmente todas las aplicaciones web tienen este problema

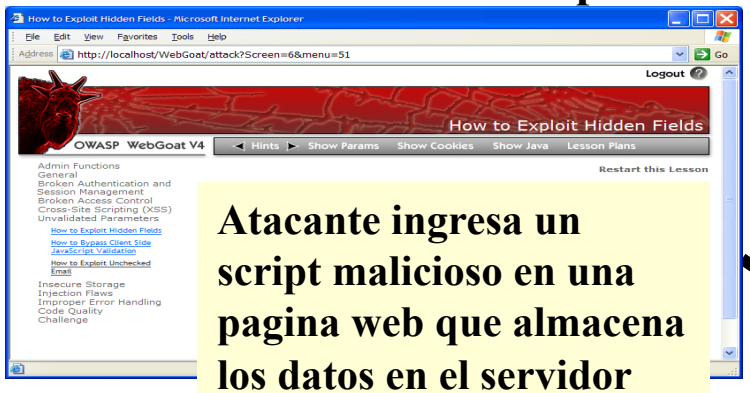
- Intentar esto en un navegador – javascript:alert(document.cookie)

Impacto Típico

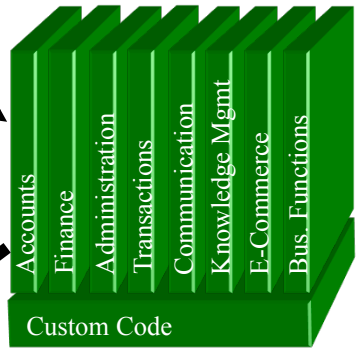
- Robar la sesión del usuario, robar datos sensibles, redireccionar un usuario hacia un sitio de malware o phishing
- Mas grave: Instalar un proxy XSS que permite a un atacante observar y dirigir todas las actividades de un usuario en el sitio vulnerable y forzarlo hacia otros sitios

XSS - Demostración

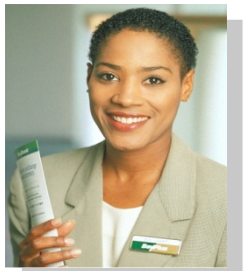
1 Atacante establece una trampa – actualizar perfil

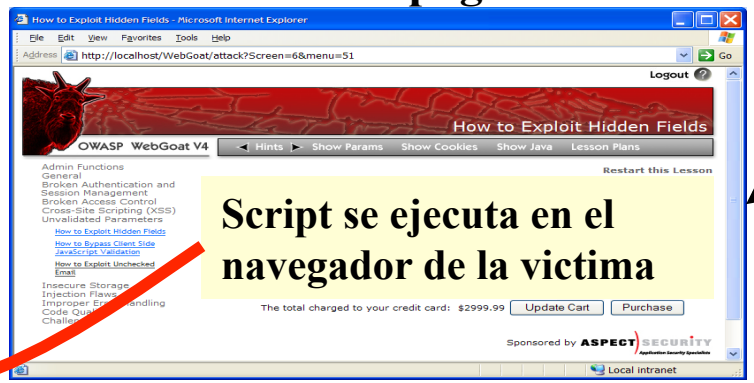


Aplicación con vulnerabilidad XSS Reflejado



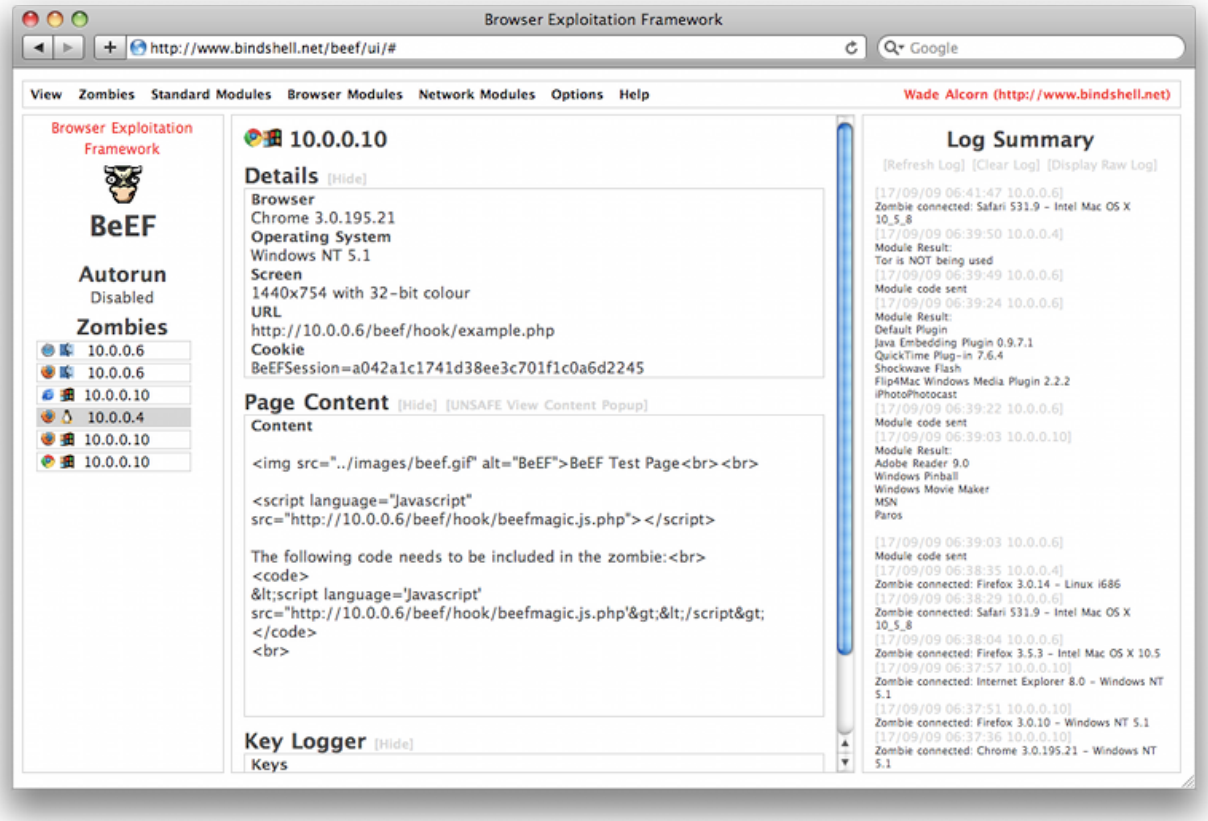
2 Victima visualiza la pagina – accede al perfil





3 Script silenciosamente envía la sesión de la victima al atacante

XSS - Demostración



BEEF Demo

A2 – Como evitar Fallas de XSS

- Recomendaciones
  - Eliminar la Falla
    - No incluir entradas suministradas por el usuario en la pagina de salida
  - Defenderse de la Falla
    - Recomendación Principal: Codificar todos los datos de entrada en la pagina de salida (Utilizar OWASP's ESAPI para dicha tarea):
      http://www.owasp.org/index.php/ESAPI
    - Siempre efectuar una validación 'positiva' de todas las entradas realizadas por el usuario
    - Cuando grandes cantidades de HTML son suministradas por el usuario, utilizar OWASP's AntiSamy para sanitizar dicho HTML
      Ver: http://www.owasp.org/index.php/AntiSamy

(AntiSamy)

- Referencias
  - Para ver como codificar datos en la pagina de salida:
    http://www.owasp.org/index.php/XSS_(Cross Site Scripting) Prevention Cheat Sheet

A3 – Perdida de Autenticación y Gestión de Sesiones

HTTP es un protocolo "sin estado"

- Significa que las credenciales tienen que viajar en cada pedido HTTP
- Debería utilizar SSL para todo contenido que requiere autenticación

Fallas en la gestión de sesiones

- SESSION ID es utilizado para mantener el estado ya que HTTP no puede
  - Para un atacante es igual de útil que poseer el usuario y contraseña
- SESSION ID es típicamente expuesto en la red, en el navegador, los logs, etc

Tener cuidado con las "puertas laterales"

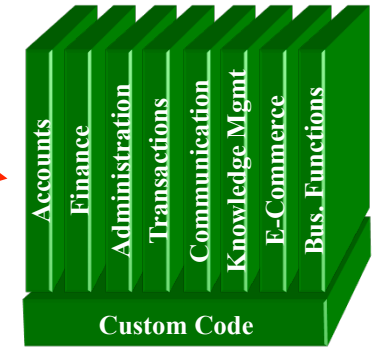
- Cambio de contraseña, recordar contraseña, olvidar la contraseña, pregunta secreta, desconexión del sitio, cambio de correo electrónico, etc.

Impacto Típico

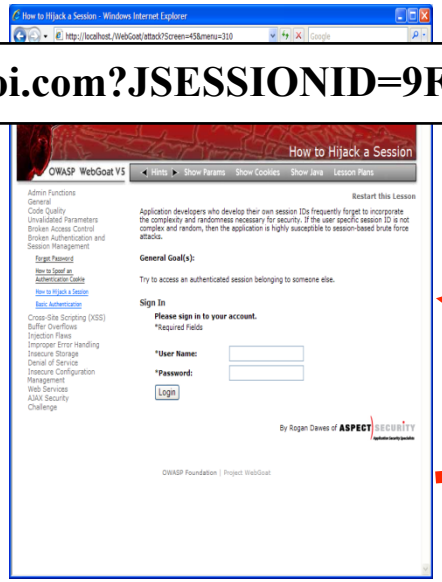
- Cuentas de usuario comprometidas o sesiones de usuario secuestradas

Perdida de Autenticación - Demostración

1 Usuario envia credenciales



www.boi.com?JSESSIONID=9FA1DB9EA...



2 Sitio utiliza reescritura de URLs

(ej., escribe sesion en URL)

3 Usuario hace clic en un link hacia http://www.hacker.com en un foro

4 Hacker verifica encabezados de referencia en los logs de www.hacker.com y encuentra la JSESSIONID del usuario

5 Hacker utiliza la JSESSIONID y toma posesión de la cuenta del usuario

A3 – Como evitar la Perdida de Autenticación y Gestión de Sesiones

- Verificar la arquitectura
  - Autenticación debería ser simple, centralizada y estandarizada
  - Utilizar el gestor de sesiones estándar provisto por el servidor de aplicaciones – no inventar uno propio!
  - Estar seguro que SSL protege tanto las credenciales como las sesiones de usuario todo el tiempo
- Verificar la implementación
  - No utilizar solamente análisis automático
  - Verificar el certificado SSL
  - Examinar todas las funciones relacionadas a autenticación
  - Verificar que "cierre de sesión" efectivamente destruya la sesión
  - Utilizar OWASP's WebScarab para testear la implementación
- Para mayor información:
  - http://www.owasp.org/index.php/Authentication_Cheat_Sheet

A4 – Referencia Directa Insegura a Objetos

Como proteger el acceso a los datos?

- Esto forma parte de realizar una "Autorización" apropiada, junto con Restringir el Acceso a URLs (A8)

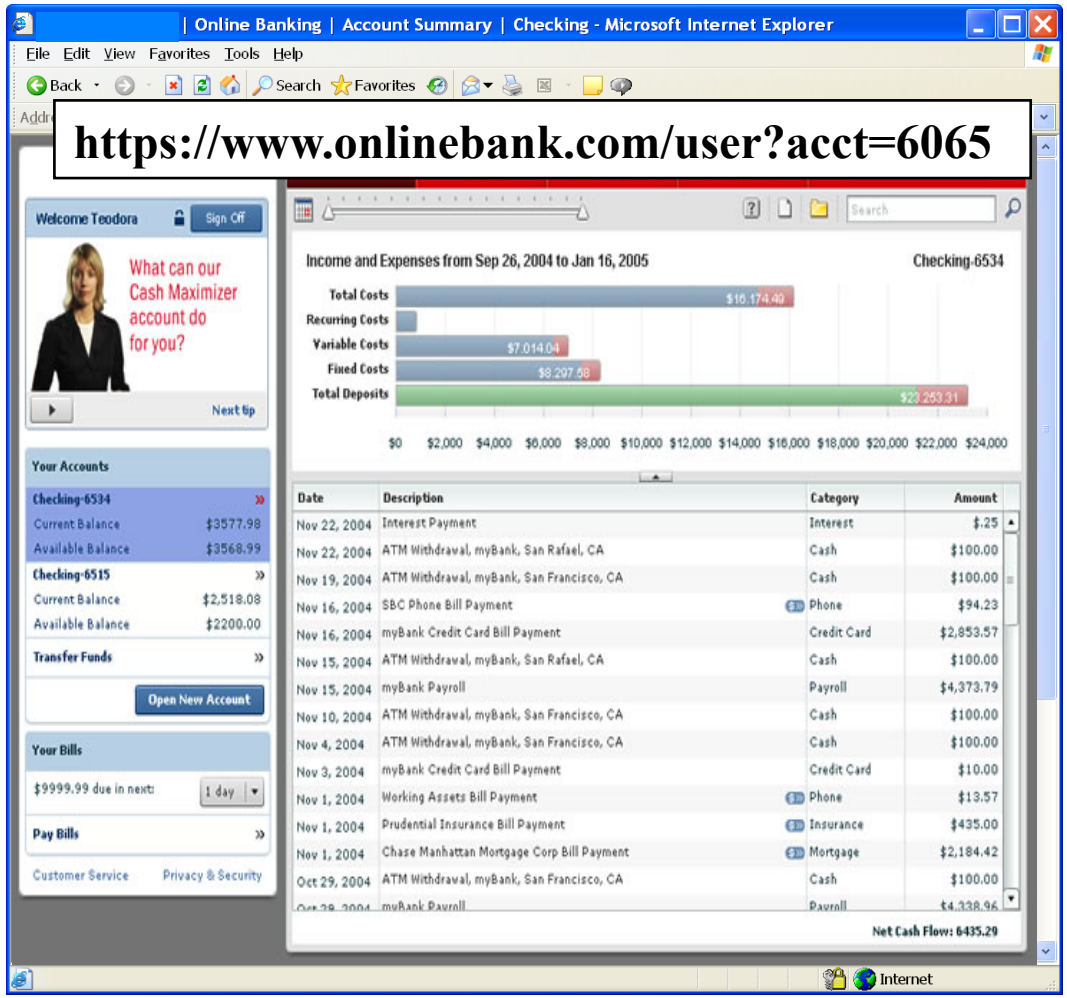
Un error común…

- Listar solamente los objetos 'autorizados' para el usuario actual,
- Ocultar las referencias a objetos en campos ocultos
- … y luego no imponer estas restricciones del lado del servidor
- Esto se denomina control de acceso en la capa de presentación, y no funciona efectivamente
- El atacante sencillamente modifica los valores de los parámetros

Impacto Típico

- Usuarios son capaces de acceder ficheros o datos sin autorización

Referencia Directa Insegura a Objetos - Demostración



- Atacante identifica su numero de cuenta 6065

  ?acct=6065

- Lo modifica a un numero parecido

  ?acct=6066

- Atacante visualiza los datos de la cuenta de la victima

A4 – Como evitar Referencias Directas Inseguras a Objetos

- Eliminar la referencia directa a objetos
  - Reemplazarla con un valor temporal de mapeo (ej. 1, 2, 3)
  - ESAPI proporciona soporte para mapeos numéricos y aleatorios
    - IntegerAccessReferenceMap & RandomAccessReferenceMap

http://app?file=Report123.xls
http://app?file=1

http://app?id=9182374
http://app?id=7d3J93

Report123.xls

Acct:9182374

- Validar la referencia directa al objeto
  - Verificar que el valor del parámetro se encuentra adecuadamente formateado
  - Verificar que el usuario se encuentra autorizado a acceder el objeto determinado
    - Restricciones en los parámetros funcionan muy bien!
  - Verificar que el modo de acceso al objeto solicitado se encuentra autorizado (ej., lectura, escritura, modificación)

A5 – Falsificación de Petición en Sitios Cruzados (CSRF)

Falsificación de Petición en Sitios Cruzados

- Es un ataque donde el navegador de la victima es engañado para que emita un comando a una aplicación web vulnerable
- La vulnerabilidad es causada debido a que los navegadores incluyen automáticamente información de autenticación del usuario (ID de sesión, dirección IP, credenciales de dominio Windows, …) en cada pedido HTTP

Imagínese…

- Que pasaría si un atacante pudiera mover su ratón y lograra que usted haga clic en enlaces de su aplicación bancaria?

Impacto Típico

- Iniciar Transacciones (transferencia de fondos, desconectar el usuario, cierre de cuenta, etc)
- Acceder datos sensitivos
- Cambiar detalles de la cuenta

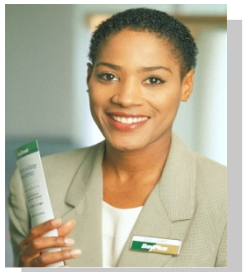
CSRF - Demostración

1

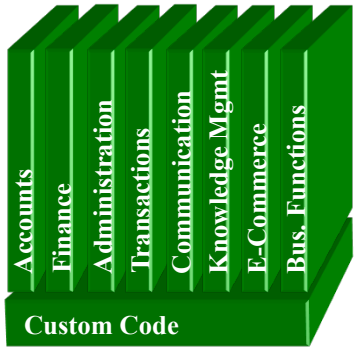
Atacante establece una trampa en algún sitio web en la Internet (o simplemente a través de un correo electrónico)

2

Mientras que la victima se encuentra conectada al sitio vulnerable, visualiza el sitio del atacante



Aplicación con vulnerabilidad CSRF



3

Sitio vulnerable recibe un pedido legitimo de la victima y ejecuta la acción solicitada

CSRF - Demostración

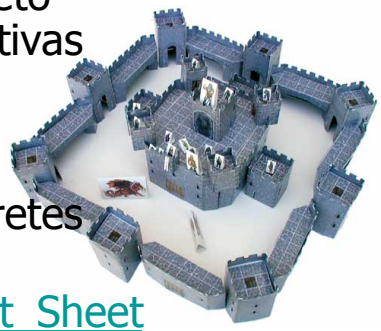
OWASP CSRF Tester

A5 – Como evitar Fallas de CSRF

- Agregue un token secreto, no enviado automáticamente, a todos los pedidos HTTP sensitivos
  - Esto hace imposible al atacante falsificar el pedido HTTP
    - (al menos que exista una vulnerabilidad XSS en la aplicación)
  - Los tokens deben ser lo suficientemente fuertes o aleatorios

- Opciones
  - Almacenar una token único en la sesión y agregarlo en todos los formularios y enlaces
    - **Campo Oculto:** `<input name="token" value="687965fdfaew87agrde" type="hidden"/>`
    - **URL de uso unico:** `/accounts/687965fdfaew87agrde`
    - **Token en el formulario:** `/accounts?auth=687965fdfaew87agrde ...`
  - Tener cuidado de no exponer el token en el encabezado de referencia
    - Se recomienda el uso de campos ocultos
  - Se puede tener un token único por cada función
    - Use un hash del nombre de una función, id de sesión, y un secreto
  - Se puede solicitar una autenticación secundaria para funciones sensitivas

- No permita que atacantes almacenen ataques en su sitio
  - Codificar todas las entradas en la salida
  - Esto "inactiva" todos los enlaces/pedidos en la mayoría de los interpretes



Para mayor información: www.owasp.org/index.php/CSRF_Prevention_Cheat_Sheet

A6 – Configuración Defectuosa de Seguridad

Las aplicaciones web dependen de cimientos seguros

- Desde el sistema operativo hasta el servidor de aplicaciones
- No olvidarse de todas las librerías utilizadas!!

Es su código fuente un secreto?

- Piense en todos los lugares donde se encuentra su código fuente
- Una seguridad eficaz no requiere que su código fuente sea secreto

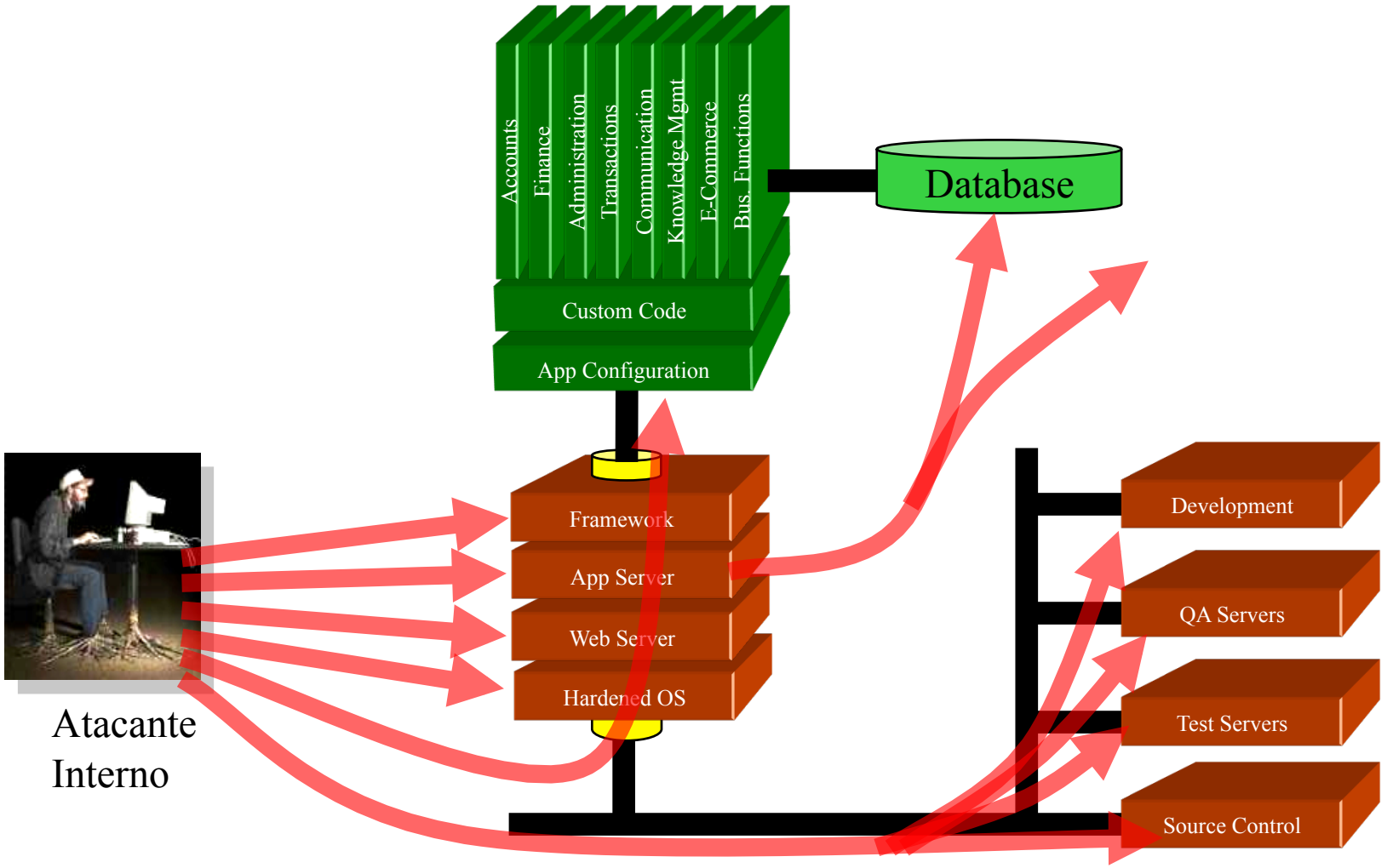
La CS debe ser extendida a todas las partes de la aplicación

- Por ejemplo, todas las credenciales deberían cambiar en el ambiente de producción

Impacto Típico

- Instalación de código malicioso debido a un parche faltante en el OS o servidor
- Falla de XSS debido a un parche faltante en el framework de la aplicación
- Acceso no autorizado a cuentas por defecto, funcionalidad de la aplicación, etc debido a una defectuosa configuración del servidor

Configuración Defectuosa de Seguridad - Demostración



A6 – Como evitar una Configuración Defectuosa de Seguridad

- Verificar la gestión de configuración de sus sistemas
  - Uso de guías de securizacion
    - Automatizar tareas es MUY UTIL aquí
  - Mantener actualizadas todas las plataformas
  - Aplicar parches en todos los componentes
    - Esto incluye librerías de software, no solo OS y servidor de aplicaciones
  - Analizar los efectos de estos cambios (en un entorno de prueba)

- Puede "volcar" la configuración de la aplicación?
  - Desarrolle reportes en sus procesos
  - La regla es simple: Si no se puede verificar, no es seguro

- Verificar la implementación
  - Un simple escaneo puede encontrar problemas de configuración genéricos y parches faltantes

A7 – Almacenamiento Criptográfico Inseguro

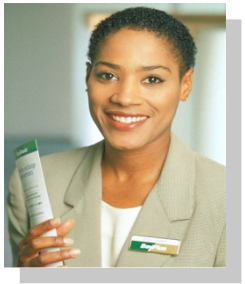
Ocurre cuando…

- No se identifican todos los datos sensitivos
- No se identifican todos los lugares donde estos datos son almacenados
  - Base de datos, ficheros, carpetas, archivos de log, backups, etc.
- No se protege esta información en todas sus ubicaciones

Impacto Típico

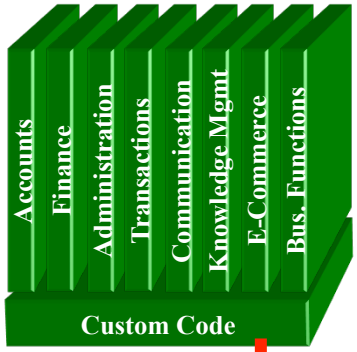
- Atacantes acceden o modifican información privada o confidencial
  - Ej., tarjetas de crédito, registros médicos, datos financieros
- Atacantes extraen secretos a ser usados en otros ataques
- Mala Imagen para la Compañía, clientes insatisfechos, y perdida de confianza
- Gastos para corregir el incidente, tales como análisis forense, enviar cartas de disculpas, reemisión de tarjetas de crédito, etc
- El Negocio es demandado o multado

Almacenamiento Criptográfico Inseguro - Demostración



1

Victima ingresa su tarjeta de credito en un formulario web



El gestionador de errores almacena los datos de la TC porque el gateway no se encuentra disponible

2

4

Atacante interno obtiene 4 millones de tarjetas de credito

Logs son accesibles a todos los miembros de IT por motivos administrativos

3

A7 – Como evitar un Almacenamiento Criptográfico Inseguro

- Verificar la arquitectura
  - Identificar todos los datos sensitivos
  - Identificar todos los lugares donde estos datos son almacenados
  - Utilice encripcion para contrarrestar las amenazas, no solo para 'encriptar' datos
- Proteger la información con mecanismos apropiados
  - Encripcion en ficheros, base de datos, etc
- Utilizar los mecanismos correctamente
  - Usar únicamente algoritmos públicos reconocidos (como AES, RSA, y SHA-256)
  - No utilizar algoritmos considerados débiles (como MD5 o SHA1)
  - Nunca transmitir claves privadas por canales inseguros
  - No almacenar información innecesaria. Ej, código CVV de tarjeta de crédito (req. PCI DSS)
- Verificar la implementación
  - Un algoritmo estándar es utilizado, y es el algoritmo apropiado para dicha situación
  - Todas las llaves, certificados, y contraseñas se encuentran debidamente almacenadas y protegidas
  - Los métodos para distribuir las llaves son seguros y efectivos
  - Mas difícil: Analizar el código de encripcion por posibles fallas

A8 – Falla de Restricción de Acceso a URL

Como proteger el acceso a URLs (páginas)?

- Esto forma parte de realizar una "autorización" apropiada junto con prevenir Referencias Directas Inseguras a Objetos (A4)

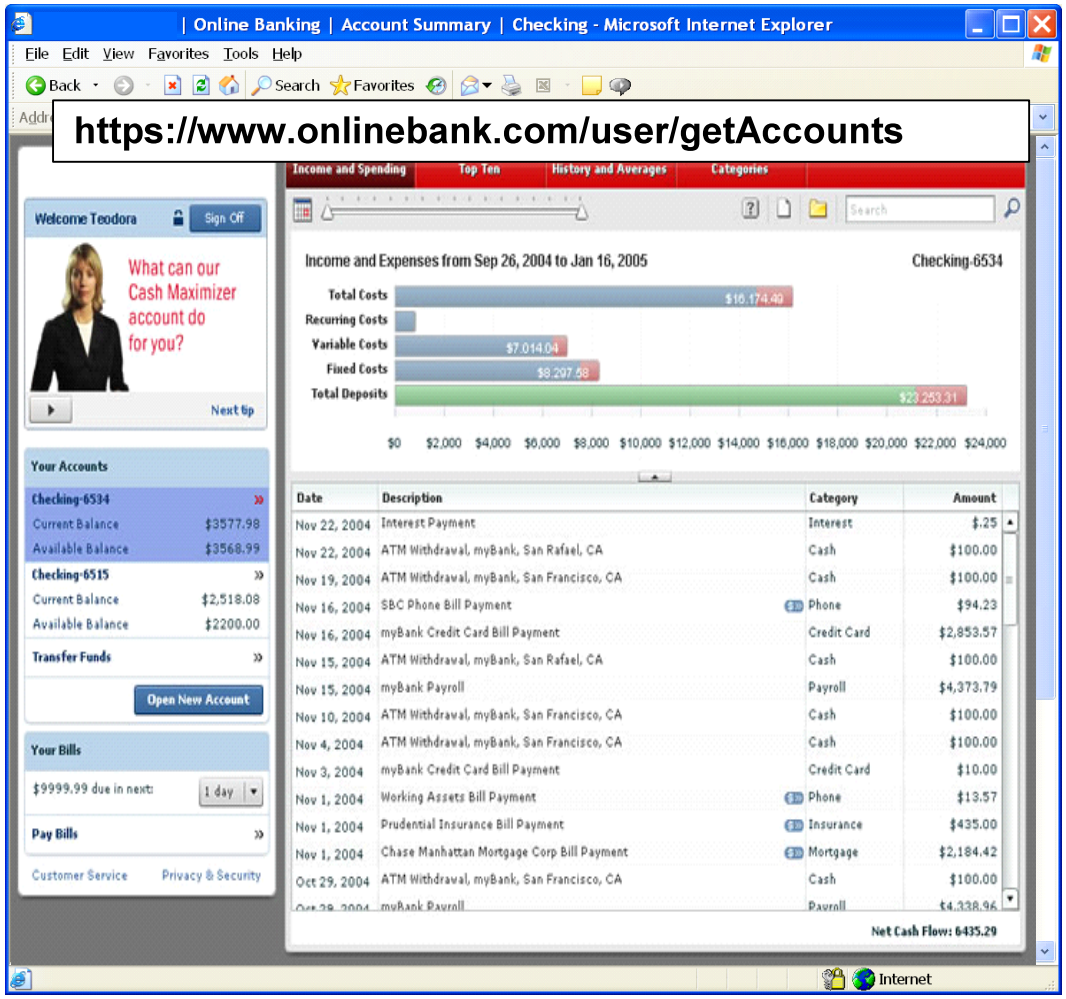
Un error común…

- Listar solamente enlaces y opciones de menu autorizados
- Esto se denomina control de acceso en la capa de presentación, y no funciona efectivamente
- El atacante simplemente modifica la URL para acceder a paginas no autorizadas

Impacto Típico

- Atacantes invocan funciones y servicios a los cuales no se encuentran autorizados
- Acceso a otras cuentas de usuario y datos
- Realizar acciones privilegiadas (admin)

Falla de Restricción de Acceso a URL - Demostración



- Atacante identifica que la URL indica su perfil

  /user/getAccounts

- La modifica apuntando a otra carpeta (perfil)

  /admin/getAccounts, o

  /manager/getAccounts

- Atacante visualiza otras cuentas

A8 – Como evitar Fallas de Restricción de Acceso a URL

- Por cada URL, un sitio necesita realizar 3 cosas
  - Restringir el acceso solo a usuarios autenticados (en caso que no sea publica)
  - Imponer los permisos de usuario o rol (en caso que sea privado)
  - Deshabilitar completamente los pedidos a paginas desautorizadas (ej., ficheros de config, ficheros de log, codigo fuente, etc.)
- Verificar la arquitectura
  - Utilizar un modelo simple y positivo en cada capa
  - Asegurarse que existe un mecanismo en cada capa
- Verificar la implementación
  - No utilizar análisis automático para detectar fallas
  - Verificar que cada URL en la aplicación se encuentra protegida por
    - Un filtro externo, como Java EE web.xml o un producto comercial
    - O controles internos en el codigo – Usar ESAPI's isAuthorizedForURL()
  - Verificar que la configuracion del servidor deshabilite pedidos a paginas no autorizadas (ej. .conf)
  - Utilizar WebScarab para 'falsificar' pedidos no autorizados

A9 – Protección Insuficiente en la Capa de Transporte

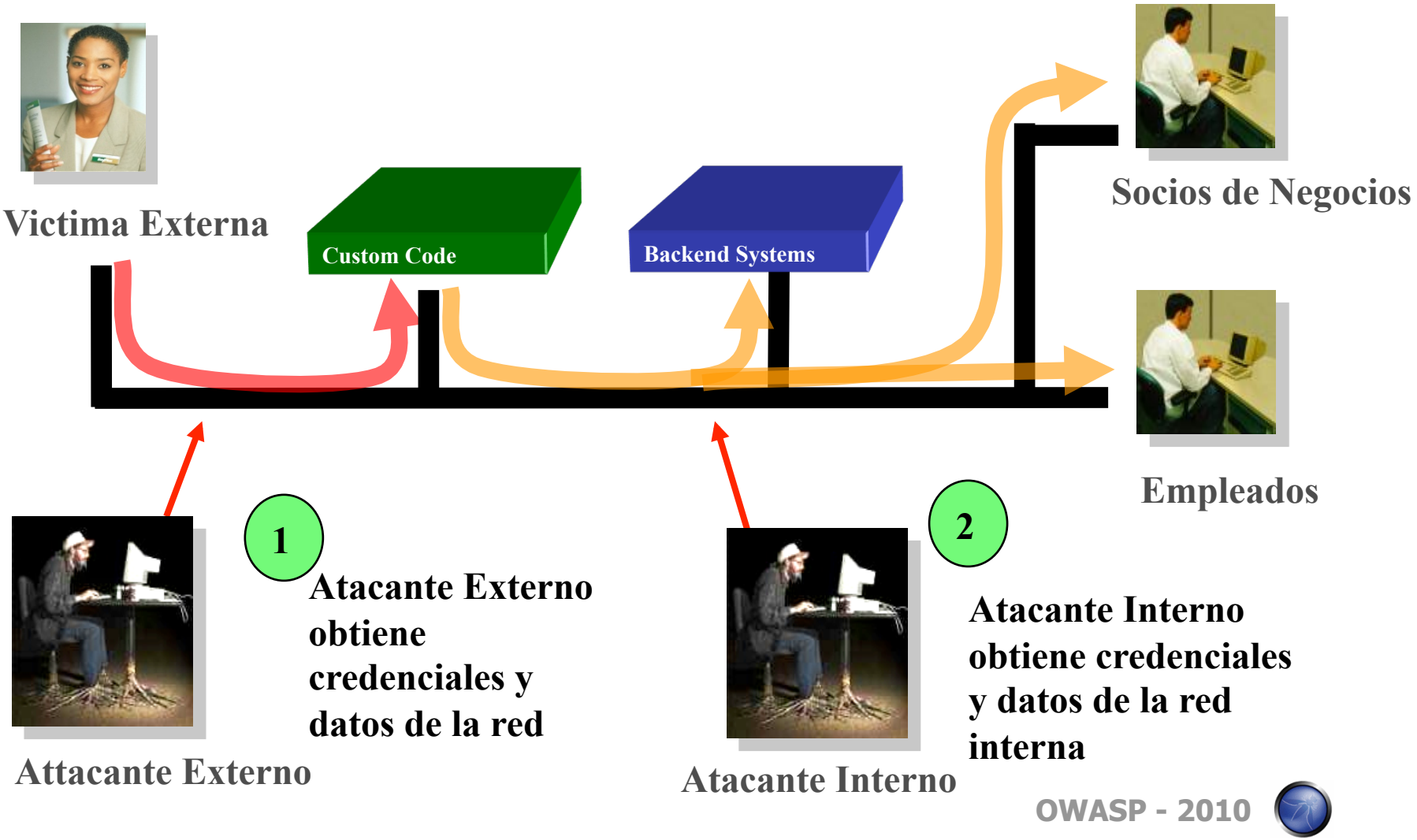
Ocurre cuando…

- No se identifican todos los datos sensitivos
- No se identifican todos los lugares donde estos datos son enviados
  - En la web, bases de datos, socios de negocios, comunicaciones internas
- No se protege esta información en todas sus ubicaciones

Impacto Típico

- Atacantes acceden o modifican información privada o confidencial
  - Ej., tarjetas de crédito, registros médicos, datos financieros
- Atacantes extraen secretos a ser usados en otros ataques
- Mala Imagen para la Compañía, clientes insatisfechos, y perdida de confianza
- Gastos para corregir el incidente, tales como análisis forense, enviar cartas de disculpas, reemisión de tarjetas de crédito, etc
- El Negocio es demandado o multado

Protección Insuficiente en la Capa de Transporte - Demostración



A9 – Como evitar Protección Insuficiente en la Capa de Transporte

- Proteger con mecanismos apropiados
  - Utilizar TLS en todas las conexiones con datos sensitivos
  - Encriptar mensajes individualmente antes de transmitirlos
    - Ej., XML-Encryption
  - Firmar digitalmente los mensajes antes de transmitirlos
    - Ej., XML-Signature
- Utilizar los mecanismos correctamente
  - Verificar los certificados SSL antes de usarlos
  - Utilizar SSL en cualquier comunicación autenticada o al transmitir información sensible
  - Verificar que la comunicación entre componentes (ej., servidor web y base de datos) también utiliza un canal seguro.
  - Para mayor información: http://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet

A10 – Redirecciones y Destinos No Validados

Las redirecciones en aplicaciones web son muy comunes

- Frecuentemente incluyen parametros suministrados por el usuario en la URL destino
- Si no son validados, el atacante puede enviar a la victima a un sitio de su elección

Los destinos (llamados transferencias en .NET) son comunes

- Internamente envian el pedido a una nueva pagina en la misma aplicacion
- Algunas veces los parametros definen la pagina de destino
- Si no son validados, el atacante puede utilizar destinos no validados para 'evitar' controles de autenticación

Impacto Típico

- Redireccionar a una victima hacia un sitio de phishing o malware
- El pedido del atacante es ejecutado, pasando por alto los controles de seguridad

Redirecciones No Validadas - Demostración

1 Atacante envia un ataque a la victima a traves de correo

Agencia Tributaria

GOBIERNO DE ESPAÑA

Notificación de Impuestos

Después de los cálculos del último informe anual de su actividad fiscal hemos Determinado que usted es elegible para Recibir un reembolso de impuestos de 186.80 euros.

Por favor, envíe la solicitud de reembolso de impuestos y nos permiten 3 días para procesarla.

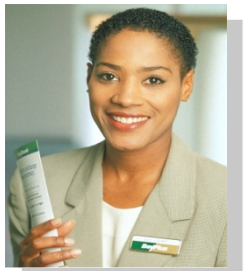
Para acceder al formulario para su reembolso de impuestos, haga clic aquí

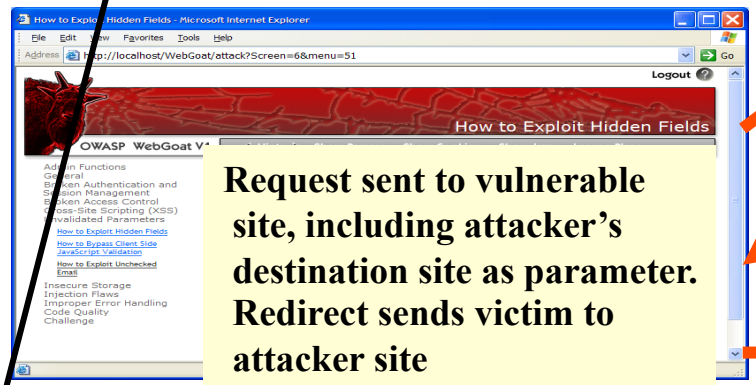
Un reembolso se puede retrasar para una variedad de razones. Por ejemplo, la presentación de los registros inválidos o aplicándose después del plazo.

Documento de Referencia: (92054568).

© Agencia Tributaria

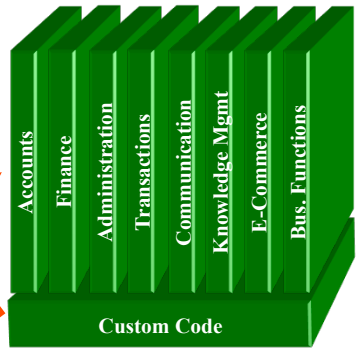
2 Victima hace clic en el enlace con parametros invalidados





Request sent to vulnerable site, including attacker's destination site as parameter. Redirect sends victim to attacker site

3 Aplicación redirecciona a la victima hacia el sitio del atacante



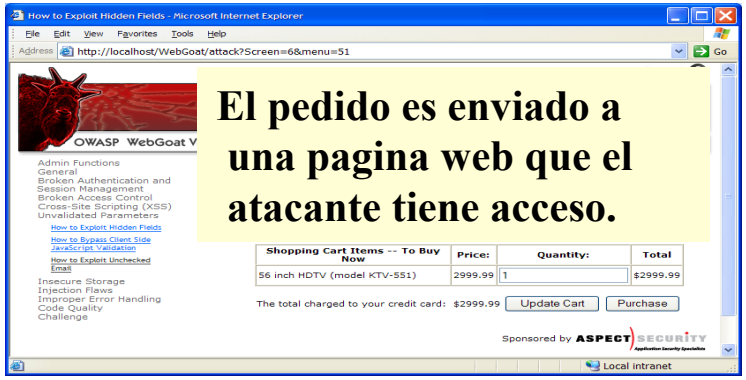
4 Sitio maligno instala malware en la victima

http://www.agenciatributaria.es/taxrefund/claim.jsp?year=2010&… &dest=www.sitiomaligno.com

Destinos No Validados - Demostración

1

Atacante envia un ataque a la pagina web vulnerable



2

La aplicación autoriza el pedido y lo procesa

Filtro

```
public void doPost( HttpServletRequest request,
HttpServletResponse response) {
 try {
    String target = request.getParameter( "dest" ) );
       ...
       request.getRequestDispatcher( target ).forward
       (request, response);
   }
   catch ( ...
```

3

La pagina destino falla al validar el parámetro, enviando al atacante directamente hacia una pagina no autorizada, salteando los controles de acceso.

```
public void sensitiveMethod
( HttpServletRequest request,
HttpServletResponse response) {
 try {
        // Do sensitive stuff here.
        ...
   }
   catch ( ...
```

http://www.ejemplo.com/destino.jsp?fwd=admin.jsp

A10 – Como evitar Redirecciones y Destinos No Validados

- Existen varias opciones
  1. Intentar evitar el uso de redirecciones y destinos (es difícil, verdad?)
  2. No utilizar parámetros provistos por usuarios para definir la URL destino
  3. Si se deben utilizar dichos parámetros, entonces:
     a) Validar cada parámetro para asegurarse que es valido y se encuentra autorizado para el usuario actual, o
     b) Preferido – Utilizar mapeos del lado del servidor para 'traducir' la opción provista al usuario en la verdadera pagina de destino
  - Defensa en profundidad: Para las redirecciones, validar la URL destino luego que es calculada para asegurarse que efectivamente sea un sitio externo autorizado
  - ESAPI puede realizar todo esto!!
    - See: SecurityWrapperResponse.sendRedirect(URL)
    - http://owasp-esapi-java.googlecode.com/svn/trunk_doc/org/owasp/esapi/filters/SecurityWrapperResponse.html#sendRedirect(java.lang.String)

- Algunas ideas para proteger los destinos no validados
  - Idealmente, se invocaría al controlador de acceso para asegurarse que el usuario se encuentra autorizado antes de realizar la transferencia (con ESAPI, es fácil)
  - Asegurarse que los usuarios que pueden acceder la pagina de origen se encuentran TODOS autorizados a acceder la pagina destino.

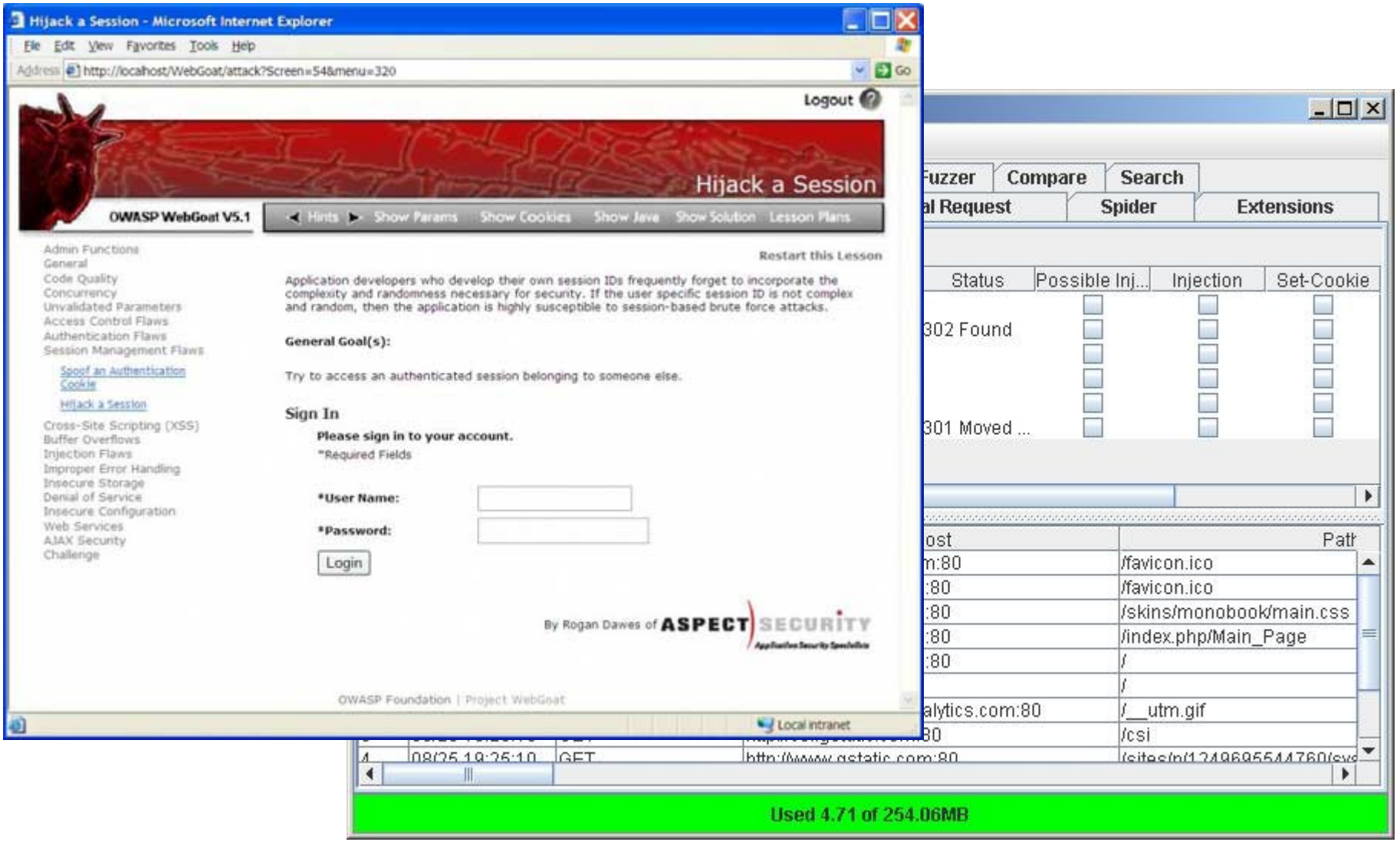
Un acercamiento por fases a la Seguridad de Aplicaciones

1. Crear conciencia sobre estas cuestiones en los equipos de desarrollo.
   - Top 10/WebScarab/Webgoat Training

2. Proveer a los equipos de desarrollo con herramientas & documentacion
   - OWASP ESAPI
   - OWASP Dev & Code Review Guides

3. Crear un equipo independiente en seguridad de aplicaciones
   - OWASP ASVS (Auditoria)
   - OWASP Testing Guide (PenTest)

OWASP WebGoat & WebScarab



El problema de las vulnerabilidades

- Cada vulnerabilidad surge de….

- Controles faltantes
  - Falta de Encripcion
  - Falla al realizar control de accesos

- Controles rotos
  - Algoritmo de encripcion debil
  - Validacion de entrada solo del lado cliente

- Controles ignorados
  - Encripcion mal implementada
  - No utilizar codificacion de salida

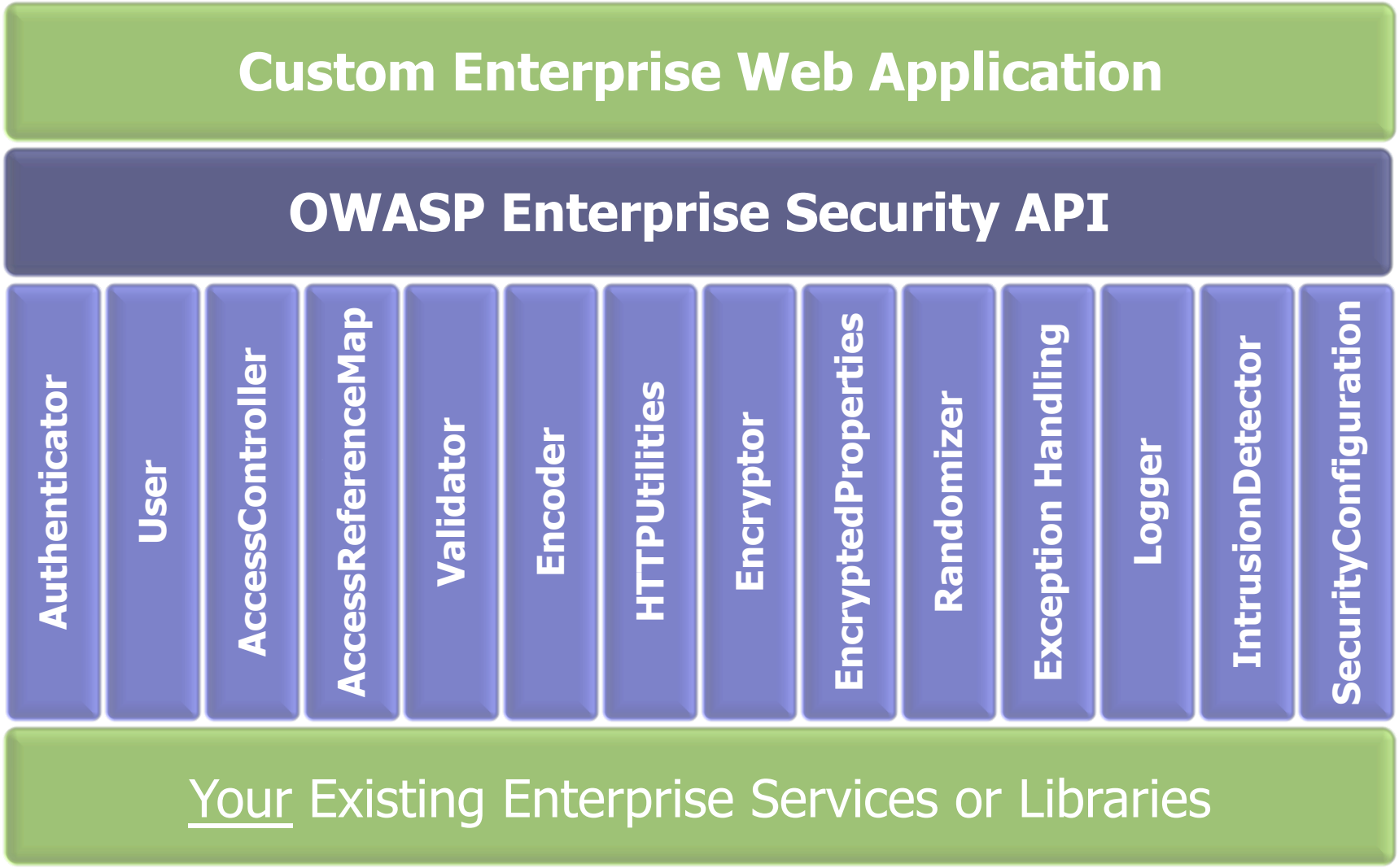
Imagine una API Empresarial de Seguridad

- Todos los controles de seguridad que un desarrollador necesita

- Estandar
- Centralizada
- Organizada
- Integrada
- Alta Calidad
- Intuitiva
- Comprobada (prox. por la NSA)

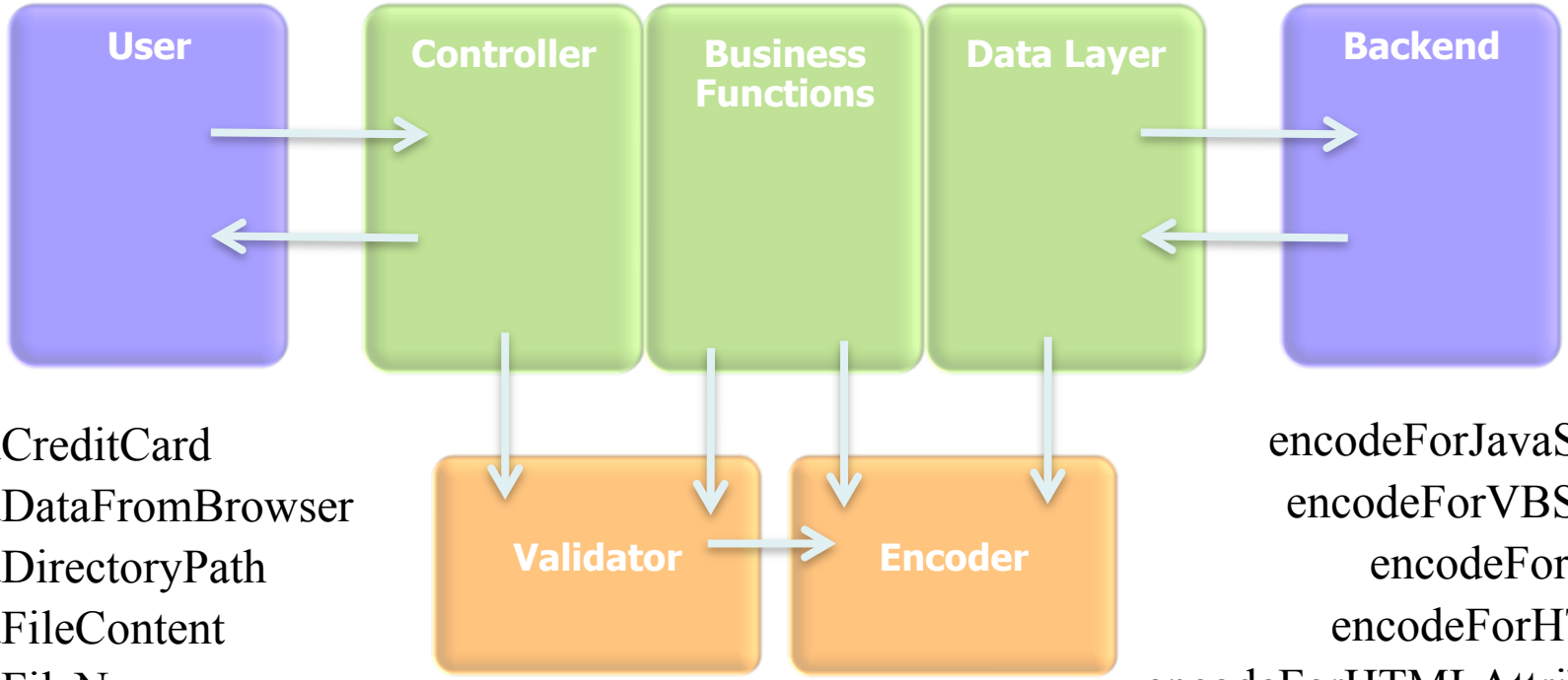
- Resuelve los problemas de controles faltantes y averiados

OWASP (ESAPI)



ESAPI Homepage: http://www.owasp.org/index.php/ESAPI

Manejo de Validación y Codificación



isValidCreditCard
isValidDataFromBrowser
isValidDirectoryPath
isValidFileContent
isValidFileName
isValidHTTPRequest
isValidListItem
isValidRedirectLocation
isValidSafeHTML
isValidPrintable
safeReadLine

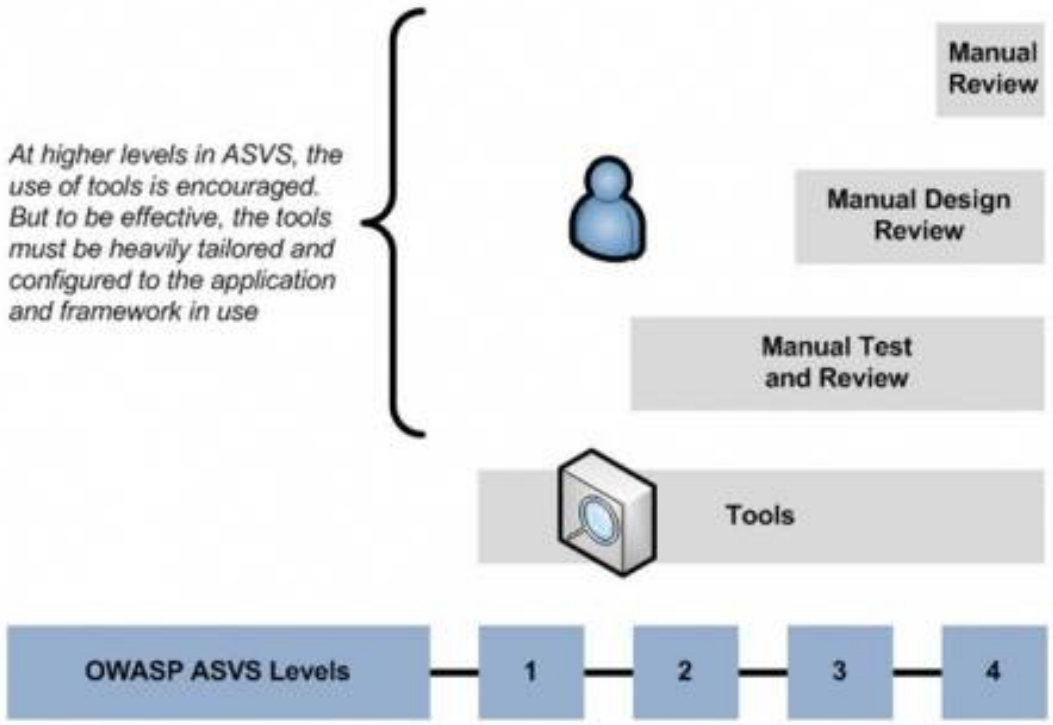
Canonicalization
Double Encoding Protection
Sanitization
Normalization

encodeForJavaScript
encodeForVBScript
encodeForURL
encodeForHTML
encodeForHTMLAttribute
encodeForLDAP
encodeForDN
encodeForSQL
encodeForXML
encodeForXMLAttribute
encodeForXPath

OWASP Top Ten Coverage

OWASP Top Ten	OWASP ESAPI
A1. Cross Site Scripting (XSS)	Validator, Encoder
A2. Injection Flaws	Encoder
A3. Malicious File Execution	HTTPUtilities (upload)
A4. Insecure Direct Object Reference	AccessReferenceMap
A5. Cross Site Request Forgery (CSRF)	User (csrftoken)
A6. Leakage and Improper Error Handling	EnterpriseSecurityException, HTTPUtils
A7. Broken Authentication and Sessions	Authenticator, User, HTTPUtils
A8. Insecure Cryptographic Storage	Encryptor
A9. Insecure Communications	HTTPUtilities (secure cookie, channel)
A10. Failure to Restrict URL Access	AccessController

OWASP ASVS



- Esta pensado como un estándar para verificar la seguridad de aplicaciones web
- Debería ser independiente de la aplicación
- Debería ser independiente del SDLC
- Debería definir requerimientos que puedan ser aplicados en distintas aplicaciones web sin ninguna interpretación especial

OWASP ASVS & Top 10

V3 - Session Management Verification Requirements

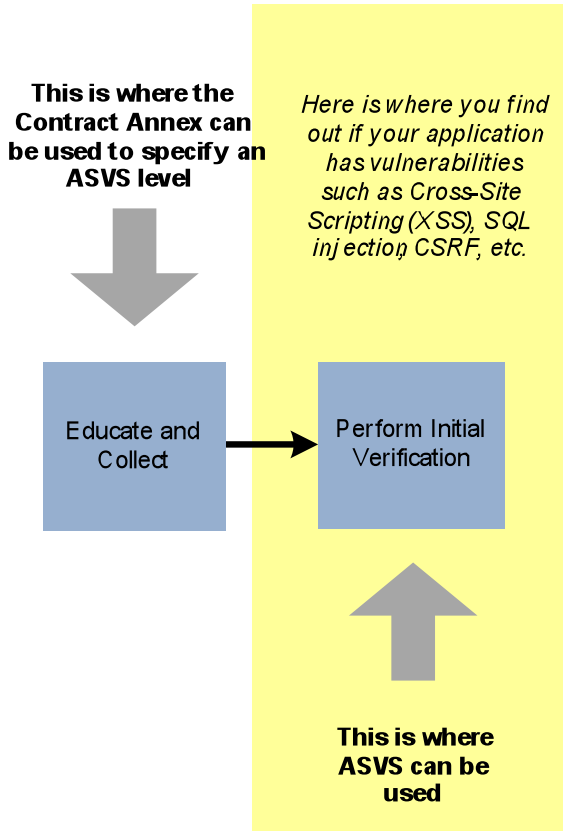
The Session Management Verification Requirements define a set of requirements for safely using HTTP requests, responses, sessions, cookies, headers, and logging to manage sessions properly. The table below defines the corresponding verification requirements that apply for each of the four verification levels.

Table 3 - OWASP ASVS Session Management Requirements (V3)

Verification Requirement		Level 1A	Level 1B	Level 2A	Level 2B	Level 3	Level 4
V3.1	Verify that the framework's default session management control implementation is used by the application.	✓		✓	✓	✓	✓
V3.2	Verify that sessions are invalidated when the user logs out.	✓		✓	✓	✓	✓
V3.3	Verify that sessions timeout after a specified period of inactivity.	✓		✓	✓	✓	✓
V3.4	Verify that sessions timeout after an administratively-configurable maximum time period regardless of activity (an absolute timeout).					✓	✓

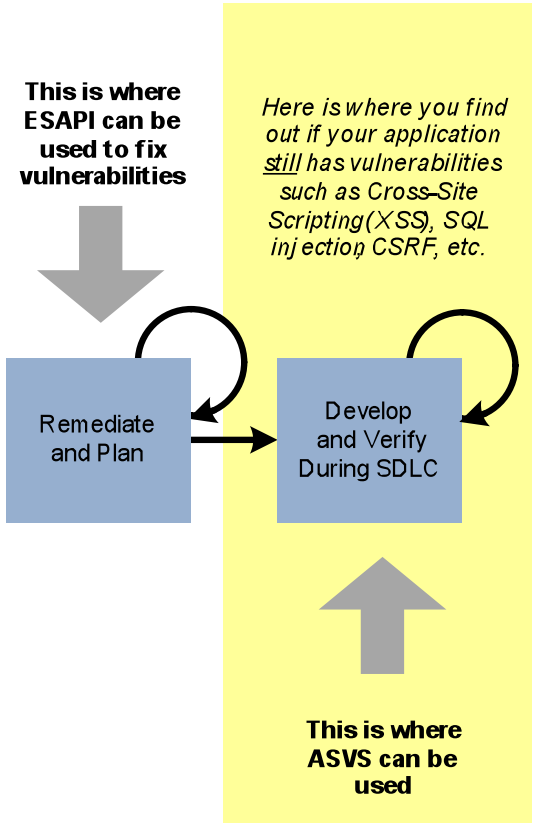
Como comienzo a utilizar ASVS?

- Comprador & Vendedor: acuerdan que requerimientos técnicos de seguridad serán verificados estableciendo un nivel de 1 a 4
- Realizar una verificación inicial de la aplicación

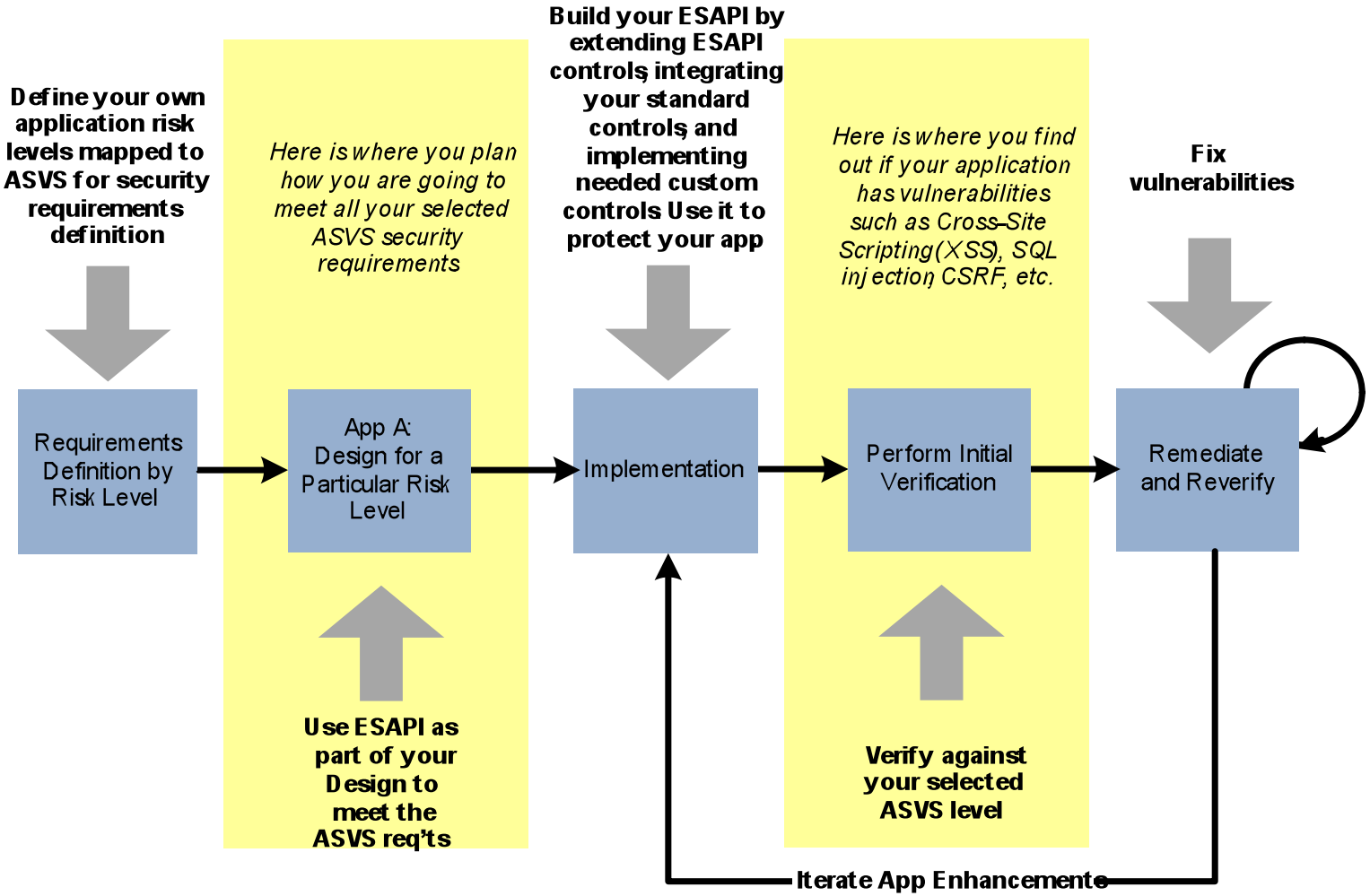


Como comienzo a utilizar ASVS? (continuacion)

- Desarrollar y ejecutar una estrategia de remediación,
- Verificar nuevamente luego que se han realizado los arreglos
- Desarrollar una estrategia que agregue un proceso de verificación en el SDLC.



Como Integrar ASVS en el SDLC
(Outsourcing no es necesario)



Resumen: Como resuelvo las problemáticas de seguridad en aplicaciones?

- Desarrollar código seguro
  - Seguir las mejores practicas en la Guía de OWASP: "Como construir aplicaciones web seguras"
    - http://www.owasp.org/index.php/Guide
  - Utilizar el Estándar OWASP de Verificación de Seguridad en Aplicaciones (ASVS) como una guía para determinar los requerimientos para que una aplicación sea segura
    - http://www.owasp.org/index.php/ASVS
  - Utilizar componentes estándares de seguridad que se adapten a la organización o entorno
    - Utilizar OWASP's ESAPI como una base para definir los componentes estándares de seguridad
    - http://www.owasp.org/index.php/ESAPI

- Revisar las aplicaciones
  - Formar un equipo de expertos para revisar las aplicaciones
  - Revisar las aplicaciones siguiendo las guias OWASP
    - Guia de Revision de Codigo OWASP: http://www.owasp.org/index.php/Code_Review_Guide
    - Guia de Testeos OWASP: http://www.owasp.org/index.php/Testing_Guide

Preguntas?

Muchas gracias!

Fcerullo@owasp.org