EL GOBIERNO Y SUS RESPONSABILIDADES FRENTE AL CÓDIGO SEGURO

Nope Saavedra Oscar Alexander
oscarnope@gmail.com
Universidad Piloto de Colombia

Resumen – El artículo menciona la responsabilidad social del gobierno frente a la seguridad, para mitigar los ataques cibernéticos. Los temas tratados giran en torno al documento CONPES 3701, que define los principios rectores de la seguridad cibernética, por parte de las autoridades gubernamentales y a la estrategia integral de Seguridad Cibernética Nacional Colombiana, centrado en el desarrollo de código seguro.

Palabras clave – Código seguro, CONPES 3701, desarrollo de aplicaciones, OWASP.

Abstract - The article mentions the social responsibility of the government against security, to mitigate cyber attacks. The topics revolve around the document CONPES 3701, which defines the guiding principles of cybersecurity, by government authorities and the comprehensive strategy Colombian National Cybersecurity focused on developing secure code.

Keywords - Insurance Code, CONPES 3701, application development, OWASP.

I. INTRODUCCIÓN

El uso de las tecnologías de la información, ha crecido en el mundo y por ende en Colombia. Además, ha incrementado considerablemente los servicios en línea que ofrecen las organizaciones públicas y privadas para ser competitivas en el mundo real, generando que todo ciudadano tenga que interactuar con sistemas de información que brinda la era digital, para el desarrollo de sus actividades económicas, académicas, culturales y sociales.

Organizaciones públicas y privadas crean servicios en línea con el fin de mejorar a la competencia; así mismo, el gobierno no se escapa de la interacción con las empresas y ciudadanos para intercambiar información, comprar bienes o servicios e informar sobre alguna solicitud o reclamo. El cambio constante y rápido de la era digital, no permite que los ciudadanos se actualicen en materia informática para conocer los riesgos que conlleva el uso de la tecnología. Cada día se realizan ataques cibernéticos, que dañan la imagen de empresas porque exponen información confidencial, claves personales, información bancaria, esto baja la confianza de los ciudadanos para el uso de los servicios en línea que brinda una organización.

Estos ataques son realizados por hackers y, son mitigados por expertos en seguridad informática por medio de controles y políticas de seguridad de cada organización, aun así, no es suficiente, ya que se debe implementar una seguridad integral, que depende de muchas variables para cada organización: de la infraestructura que posee, del software y hardware que utiliza y además debe ir de la mano con las leyes existentes que permiten procesar penalmente y judicialmente a los hackers. Muchas veces las mismas organizaciones no capacitan a los usuarios en materia de seguridad, siendo así el eslabón más débil de la cadena y permitiendo que los hackers exploten esta debilidad, utilizando la ingeniería social.

II. CONPES 3701

El documento CONPES [1], [2] tiene en cuenta el papel que juega el gobierno en la sociedad, porque es el que tiene la capacidad de sostener y mantener la seguridad integral, además es el que define los lineamientos; debe siempre participar de forma activa y constante en los cambios que requiere la era digital, prestando importancia en estos aspectos:

- Abogados que conozcan sobre tecnología.

- Universidades con programas de actualización, dirigido a expertos en seguridad.
- Leyes que soporten y protejan a los usuarios en los delitos cibernéticos.
- Inversión en infraestructura tecnológica; software y hardware que brinde la protección necesaria ante ataques cibernéticos.
- La revisión constante para que se realicen auto ataques, que permitan descubrir nuevas vulnerabilidades y pongan en riesgo los activos más críticos.

El documento CONPES no proporciona lineamientos para los siguientes aspectos:

- Directriz que proporciona seguridad física.
- Requerimientos necesarios para codificación segura.
- Requerimientos necesarios para realizar auditorías de cumplimiento en software desarrollado por fábricas de software, que son proveedores de las entidades del gobierno.
- Validez de los archivos firmados digitalmente a nivel internacional, por entidades certificadoras internacionales.

La mayoría de las aplicaciones presentan defectos desde su etapa de desarrollo, por tanto, es necesario reducir el número de vulnerabilidades más críticas, apoyados en Organizaciones Internacionales dedicadas a reunir estas estadísticas como OWASP, adicionalmente, existen librerías que ayudan a mitigar estas vulnerabilidades, como la librería de Microsoft Anti-Cross Site Scripting Library.

III. OWASP TOP 10

Es un documento de los diez riesgos de seguridad más importantes en aplicaciones web, según la organización OWASP (en inglés Open Web Application Security Project, en español Proyecto Abierto de Seguridad de Aplicaciones Web). Esta lista es publicada y actualizada cada tres años por dicha organización.

El objetivo de este proyecto según OWASP top 10 (2013), es crear conciencia acerca de la seguridad en aplicaciones, mediante la identificación de algunos de los riesgos más críticos que enfrentan las organizaciones. [3]

OWASP analiza el top 10 de las vulnerabilidades más habituales y que tienen mayor impacto en las aplicaciones WEB.

- Inyección: inyección de código, siendo las inyecciones SQL una de las más comunes.
- Pérdida de autenticación y gestión de sesiones: mal manejo de las sesiones en aquellas aplicaciones que utilizan autenticación.
- Secuencia de comandos en sitios cruzados (XSS): ocurre cuando existe validación pobre de la información ingresada por el atacante.
- Referencia directa insegura a objetos: puede derivar en un acceso no autorizado a información crítica, debido a errores en el diseño o desarrollo.
- Configuración de seguridad incorrecta: configuraciones no adecuadas que pueden impactar en la seguridad de la propia aplicación.
- Exposición de datos sensibles: protección incorrecta de datos críticos tales como, números de tarjetas de crédito, contraseñas, entre otros.
- Ausencia de control de acceso a las funciones: falta de controles desde el servidor, permitiendo que un posible atacante acceda a funciones que no debería.
- Falsificación de peticiones en sitios cruzados: permite a un atacante generar peticiones sobre una aplicación vulnerable, a partir de la sesión de la víctima.
- Uso de componentes con vulnerabilidades conocidas: explotación de librerías, frameworks y otros componentes vulnerables

por parte de un atacante, con el fin de obtener acceso o combinar otros ataques.

- Redirecciones y reenvíos no validados: uso de redirecciones de sitios web a otros sitios, utilizando información no confiable (untrusted), para redirigir a las víctimas a sitios de phishing o que contienen malware.

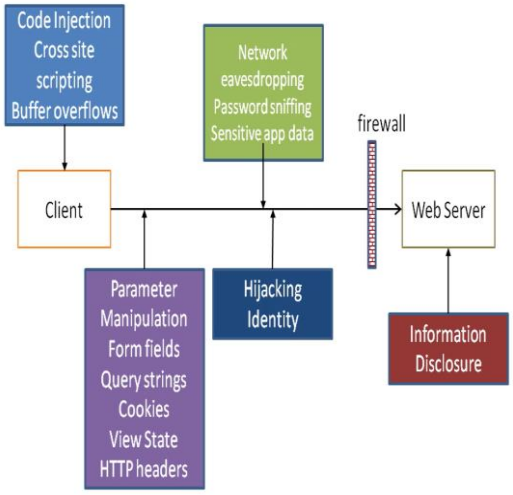

Fig. 1. Amenazas por Capas. Fuente: http://www.learnonline.com/lms/course/owp1/oasp-avoiding-havker-tricks/

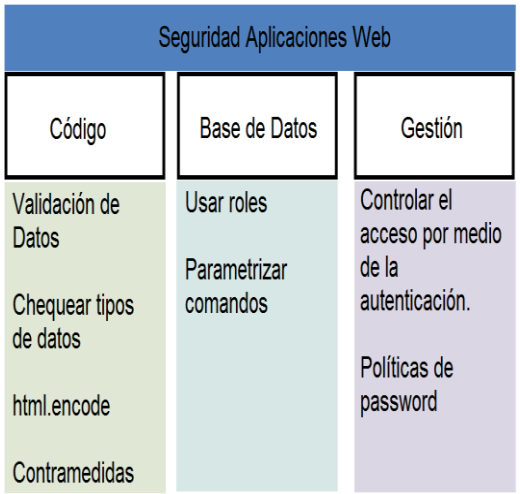

Fig. 2. Seguridad de Código. Fuente: http://ww.learnnowonline.com/lms/course/owp1/owasp-avoiding-hacker-tricks/

INYECCIÓN

Las fallas de inyección, tales como SQL, OS, y LDAP, ocurren cuando datos no confiables son enviados a un intérprete como parte de un comando o consulta. Los datos hostiles del atacante pueden engañar al intérprete en ejecutar comandos no intencionados o acceder datos no autorizados.

OBJETIVO

Verificar si la aplicación es vulnerable a una inyección, confirmando que se separa la información no confiable del comando o consulta.

VULNERABILIDAD

Causar pérdida o corrupción de datos, pérdida de responsabilidad, o negación de acceso.

Fig. 3. Inyección. Fuente Adaptada: OWASP

PÉRDIDA DE AUTENTICACIÓN Y GESTIÓN DE SESIONES

Las funciones de la aplicación relacionadas a autenticación y gestión de sesiones son frecuentemente implementadas incorrectamente, permitiendo a los atacantes contraseñas, claves, token de sesiones, o explotar otras fallas de implementación para asumir la identidad de otros usuarios.

OBJETIVO

Proteger los activos de la gestión de sesiones, como credenciales y los identificadores (ID) de sesión.

VULNERABILIDAD

Permitir que algunas o todas las cuentas sean atacadas.
Una vez que el ataque resulte exitoso, el atacante podría realizar cualquier acción.
Las cuentas privilegiadas son objetivos prioritarios.

Fig. 4. Perdida de Autenticación y Gestión de Sesiones. Fuente Adaptada: OWASP

SECUENCIAS DE COMANDOS CRUZADOS

Las fallas XSS ocurren cada vez que una aplicación toma datos no confiables y los envía al navegador web sin una validación y codificación apropiada. XSS permite a los atacantes ejecutar secuencia de comandos en el navegador de la víctima, los cuales pueden secuestrar las sesiones de usuario, destruir sitios web, o dirigir al usuario hacia un sitio malicioso.

OBJETIVO

Asegurar que todas las entradas de datos ingresadas por los usuarios, son codificadas adecuadamente.

VULNERABILIDAD

Ejecutar secuencias de comandos en el navegador de la víctima, para secuestrar las sesiones de usuario, alterar la apariencia del sitio web, insertar código hostil, redirigir usuarios, secuestrar el navegador de la víctima utilizando malware.

Fig. 5. Secuencias de Comandos Cruzados.
Fuente Adaptada: OWASP

REFERENCIA DIRECTA INSEGURA A OBJETOS

Una referencia directa a objetos ocurre cuando un desarrollador, expone una referencia a un objeto de implementación interno, tal como un fichero, directorio, o base de datos sin un chequeo de control de acceso u otra protección, los atacantes pueden manipular estas referencias para acceder datos no autorizados.

OBJETIVO

Verificar que todas las referencias a objetos tienen las protecciones apropiadas.

VULNERABILIDAD

Comprometer toda la información que pueda ser referida por parámetros.

Fig. 6. Referencia Directa Insegura a Objetos. Fuente Adaptada: OWASP

CONFIGURACIÓN DE SEGURIDAD INCORRECTA

Una buena seguridad requiere tener definida e implementada una configuración segura para la aplicación, marcos de trabajo, servidor de aplicación, servidor web, base de datos y plataforma. Todas estas configuraciones deben ser definidas, implementadas, y mantenidas ya que por lo general no son seguras; por defecto, esto incluye mantener todo el software actualizado, incluídas las librerías de código utilizadas por la aplicación.

OBJETIVO

Verificar que la aplicación cuenta con el apropiado fortalecimiento en seguridad a través de todas las capas que la componen.

VULNERABILIDAD

Dar acceso no autorizado a algunas funcionalidades o datos del sistema.

Fig. 7. Configuración de Seguridad Incorrecta. Fuente Adaptada: OWASP

EXPOSICIÓN DE DATOS SENSIBLES

Muchas aplicaciones web no protegen adecuadamente datos sensibles tales como números de tarjetas de crédito o credenciales de autenticación. Los atacantes pueden robar o modificar tales datos para llevar a cabo fraudes, robos de identidad u otros delitos. Los datos sensibles requieren de métodos de protección adicionales, tales como el cifrado de datos, así como también de preocupaciones especiales en un intercambio de datos en el navegador.

OBJETIVO

Determinar el conjunto de datos sensibles que requerirán protección extra.

VULNERABILIDAD

Comprometer todos los datos que deberían estar protegidos.

Fig. 8. Exposición de Datos Sensibles. Fuente Adaptada: OWASP

INEXISTENTE CONTROL DE ACCESO A NIVEL DE FUNCIONALIDADES

La mayoría de aplicaciones web verifican los derechos de acceso a nivel de función antes de hacer visible la función en la misma interfaz de usuario. A pesar de esto, las aplicaciones necesitan verificar el control de acceso en el servidor cuando se accede a cada función. Si las solicitudes de acceso no se verifican, los atacantes podrán realizar peticiones sin la autorización apropiada.

OBJETIVO

Verificar que la aplicación restringe adecuadamente el acceso a nivel de funcionalidades.

VULNERABILIDAD

Permitir el acceso no autorizado de los atacantes a funciones del sistema.

Fig. 9. Inexistente Control de Acceso a Nivel de Funcionalidades. Fuente Adaptada: OWASP

FALSIFICACIÓN DE PETICIONES EN SITIOS CRUZADOS (CSRF)

Un ataque CSFR obliga al navegador de una víctima autenticada, a enviar una petición HTTP falsificado, incluyendo la sesión del usuario y cualquier otra información de autenticación incluida automáticamente, a una aplicación web vulnerable. Esto permite al atacante forzar al navegador de la víctima, para generar pedidos que la aplicación vulnerable, piensa son peticiones legítimas provenientes de la víctima.

OBJETIVO

Verificar el comportamiento de la aplicación, con la ausencia de un token en cada enlace y formulario.

VULNERABILIDAD

Cambiar cualquier dato que el usuario esté autorizado a cambiar.

Fig. 10. Falsificación de Peticiones en Sitios Cruzados. Fuente Adaptada: OWASP

USO DE COMPONENTES CON VULNERABILIDADES CONOCIDAS

Algunos componentes tales como las librerías, los frameworks y otros módulos de software casi siempre, funcionan con todos los privilegios. Si se ataca un componente vulnerable, esto podría facilitar la intrusión en el servidor o una perdida seria de datos. Las aplicaciones que utilicen componentes con vulnerabilidades conocidas, debilitan las defensas de la aplicación y permiten ampliar el rango de posibles ataques e impactos.

OBJETIVO

Revisar si un componente utilizado tiene una vulnerabilidad conocida ya detectada y verificar si el fallo, tiene impacto en la aplicación.

VULNERABILIDAD

Cualquier aplicación tiene este tipo de problema debido a que la mayoría de los equipos de desarrollo no se enfocan en asegurar que sus componentes / bibliotecas se encuentren actualizadas.

Fig. 11. Uso de Componentes con Vulnerabilidades Conocidas. Fuente Adaptada: OWASP

REDIRECCIONES Y REENVÍOS NO VÁLIDOS

Las aplicaciones web frecuentemente redirigen y reenvían a los usuarios, hacia otras páginas o sitios web, y utilizan datos no confiables, para determinar la página de destino, sin una validación apropiada, los atacantes pueden redirigir a las víctimas hacia sitios de phishing o malware, o utilizar reenvíos para acceder páginas no autorizadas.

OBJETIVO

Verificar las redirecciones utilizadas en el código y validar la URL destino con una lista blanca previamente.

VULNERABILIDAD

Intentar instalar código malicioso o engañar a los usuarios, para que revelen contraseñas u otra información sensible.

Fig. 12. Redirecciones y Reenvíos no Válidos. Fuente Adaptada: OWASP

La primera amenaza crítica de las vulnerabilidades OWASP TOP 10 son de tipo inyección, vulnerabilidad que se puede mitigar desde el código de programación. El segundo tipo de vulnerabilidad más crítico corresponde al manejo de sesiones, por errores en el diseño, donde puede haber escalamiento de privilegios.

La forma de mitigar la amenaza más crítica del tipo inyección, es utilizar librerías existentes, que ayudan a minimizar el tiempo invertido en desarrollo y de esta forma se puede estandarizar su uso en el software, en lugar de crear métodos personalizados para cada aplicativo o entidad.

IV. MICROSOFT ANTI-CROSS SITE SCRIPTING LIBRARY

Microsoft Anti-Cross Site Scripting Library V4.3 (AntiXSS V4.3) es una biblioteca de codificación diseñada para ayudar a los desarrolladores, a proteger sus aplicaciones basadas en la Web ASP.NET de ataques XSS [4].

Consiste en una serie de métodos para proteger cualquier entrada del usuario al aplicativo web, protegiéndolo de ataques cross-site scripting, siendo una de las formas más comunes de atacar un sitio web.

La lista de métodos que incorpora son:

HtmlEncode, HtmlFormUrlEncode, y HtmlAttributeEncode
Codifica los datos para su uso en páginas HTML.

XmlEncode y XmlAttributeEncode
Codifica datos XML.

UrlEncode y UrlPathEncode
Codifica contenido del URL, incluyendo los parámetros de cadena de consulta.

CssEncode
Codifica las cadenas de entrada utilizados en valores de elementos CSS.

[5]

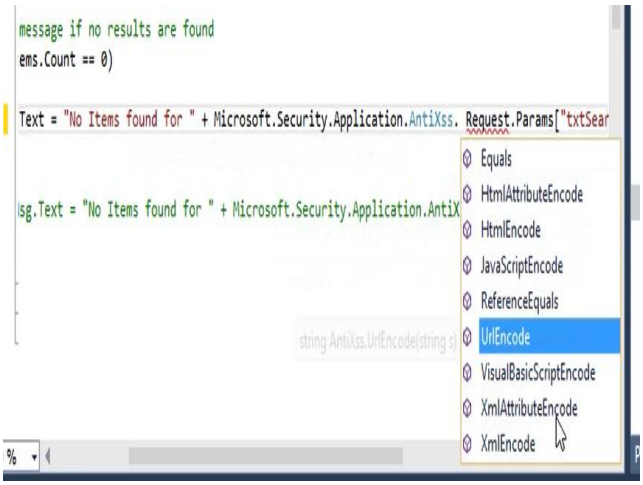

Fig. 13. Métodos librería Microsoft. Fuente: El Autor.

```
public static string AddDirectRef(string directRef)
{
    var rng = new RNGCryptoServiceProvider();
    var buff = new byte[32];
    rng.GetBytes(buff);

    var indirectRef = HttpServerUtility.UrlTokenEncode(buff);

    var map = (Dictionary<string, string>) HttpContext.Current.Session["Map"];

    if (map == null)
        map = new Dictionary<string, string>();

    map.Add(directRef, indirectRef);
    map.Add(indirectRef, directRef);

    HttpContext.Current.Session["Map"] = map;
    return indirectRef;

}
```

Fig. 14. Métodos HttpServerUtility. Fuente: El Autor.

Existen herramientas que ayudan a detectar vulnerabilidades en el código estático; aún así, es necesario realizar revisiones manuales, con el código en ejecución, para detectar el mayor número de vulnerabilidades posibles.

Algunas de las herramientas que pueden ayudar a encontrar vulnerabilidades en el código estático, son descritas a continuación:

V. MICROSOFT MINIFUZZ FILE FUZZER

Herramienta que busca encontrar errores a través del ingreso de valores no válidos, con el fin de verificar el manejo de excepciones.

El proceso de esta herramienta, consiste en que si tenemos un programa que lee un archivo, el texto sería:

1. El programa lee un archivo.
2. La herramienta crea un archivo corrupto.
3. El programa acepta el archivo corrupto.
4. Si el programa genera un error, no controlado, la vulnerabilidad ha sido explotada.

En herramientas de desarrollo como JAVA o .NET, las excepciones en el código se controlan con la siguiente sintaxis:

```
Try
{
        //Secuencia de instrucciones
        Return data;
} catch (e Exception)
{
        return null;
}
Finally
{
        fs.close();
}
```

Fig. 15. Sintaxis para el Manejo de Excepciones. Fuente: El Autor

El manejo de excepciones permite capturar errores de tipo buffer – overflow, esta es una vulnerabilidad identificada en el documento OWASP TOP 10.

Las combinaciones posibles de valores que utiliza la herramienta, para que la aplicación genere errores, son algunos de estos tipos: numérico, urls, valores negativos, valores muy grandes, caracteres de tipo HTML, solicitudes SQL, ingreso de comillas.

VI. BINSCOPE BINARY ANALYZER

Herramienta que busca vulnerabilidades en archivos binarios que pueden afectar archivos de tipo DLL, COM. Las validaciones que realiza son:

Asegura que los encabezados de los archivos binarios sean correctos.

Verifica los compiladores y enlazadores de archivos binarios, en el caso de VISUAL STUDIO.NET, se analiza el FRAMEWORK .NET utilizado, junto con su número de versión. En algunos casos, se puede encontrar que el archivo binario, fue generado con un FRAMEWORK BETA y no un RELEASE oficial. La versión BETA es una versión de prueba, puesta a disposición del público en general para encontrar fallas. Cuando esta versión es definitiva, se genera la versión RELEASE que es la que superó las pruebas de calidad y es puesta en producción.

Verifica que la DLL no sea un malware, que tiene como objetivo dañar programas que se encuentran en el computador.

VII. CODE ANALYSIS TOOL (CAT.NET)

Herramienta para realizar el análisis de código, utilizada para escanear vulnerabilidades de aplicaciones .NET desarrolladas en lenguaje C#, VB.NET y J #. Se integra con el IDE de VISUAL STUDIO.NET, revisa métodos, clases y variables declaradas dentro del código, para detectar vulnerabilidades identificadas en el documento OWASP TOP 10; ya que se integra con el IDE, es posible navegar al Código Fuente dando clic sobre la lista de hallazgos encontrados.

VIII. CONCLUSIONES

Encontrar defectos en el código en la fase de desarrollo ayuda a minimizar los costos, al contrario, si se encuentran en una fase de implementación, se pueden hacer cambios a nivel de diseño que van a desfasar las fechas de entrega; por ende, el impacto en el código puede ser mucho mayor.

Los defectos encontrados pueden impactar la imagen del negocio o activos intangibles que perjudiquen la imagen de las organizaciones y reduzcan la confianza de los usuarios que tienen de la organización.

Las buenas prácticas contemplan las amenazas más críticas y el apoyo de metodologías como OWASP, herramientas estáticas que ayudan a reducir estas vulnerabilidades con el fin de crear controles en el código que las contrarrestan.

REFERENCIAS

[1] DNP, "CONPES". [Online]. Disponible: https://www.dnp.gov.co/CONPES/Paginas/CONPES-Elaboracion.aspx

[2] BID, "Ciberseguridad ¿Estamos preparados en América Latina y el Caribe?". [Online]. Disponible: https://publications.iadb.org/handle/11319/7449

[3] Wikipedia, "Owasp Top 10". [Online]. Disponible: https://es.wikipedia.org/wiki/OWASP_Top_10

[4] Microsoft, "Microsoft Anti-Cross Site Scripting Library V4.3". [Online]. Disponible: https://www.microsoft.com/en-sg/download/details.aspx?id=43126

[5] Microsoft, "Microsoft Adds AntiXSS Tool to ASP.NET 4.5". [Online]. Disponible: http://devproconnections.com/aspnet/microsoft-adds-antixss-tool-aspenet-45