Seguridad en aplicaciones construidas por desarrolladores con visión de atacante

Figueroa Marulanda, Viviana Farina
farinamarulanda@gmail.com
Universidad Piloto de Colombia

Resumen—Este artículo pretende dar un enfoque de la importancia de aplicar un poco de la llamada malicia indígena a la hora de implementar aplicaciones, simplemente teniendo en cuenta sugerencias de codificación para estar siempre un paso delante de los posibles atacantes y a su vez proporcionar un mecanismo de defensa a las organizaciones contra los ciberataques, todo esto encaminado a acabar con la imagen de generadores de vulnerabilidad que se han creado alrededor de los desarrolladores.

Índice de Términos—*AMENAZA-APLICACIONES-ATAQUES INFORMATICOS*

Abstract—The aim of this paper is to provide an approach to the importance of thinking as an attacker when developing applications, considering simply coding tips to stay one step ahead of the hackers and in turn to provide a defense mechanisms to cyber attacks, all this aimed to mitigate the image of vulnerability generators, that has been created around the developers.

Index Terms- ATTACKS -COMPUTER APPLICATIONS- THREAT

I. INTRODUCCIÓN

Es común escuchar la frase “la cadena se rompe por su eslabón más débil” y en muchos casos esta es aplicable a los desarrolladores ya que en el concepto de quienes intervienen en el mundo TI, este ha sido uno de los puntos más frágiles en la seguridad de la información y lamentablemente la historia narra que algunos de los agujeros que han dado paso a robo de millones de tarjetas de crédito, han causado daños financiero y comprometido información sensible que terminan en la perdida reputación para las empresa, se han presentado por aplicaciones web que a pesar de estar concebidas para dar soporte a procesos de negocio o para estar expuestas a Internet, en su construcción se descuido importantes factores de seguridad.

II. ANTECEDENTES DE LA SEGURIDAD EN APLICACIONES

OWASP (acrónimo de Open Web Application Security Project, en inglés ‘Proyecto abierto de seguridad de aplicaciones web’) es un proyecto de código abierto dedicado a determinar y combatir las causas que hacen que el software sea inseguro.

La comunidad OWASP está formada por empresas, organizaciones educativas y particulares de todo mundo. Juntos constituyen una comunidad de seguridad informática que trabaja para crear artículos, metodologías, documentación, herramientas y tecnologías que se liberan y pueden ser usadas gratuitamente por cualquiera.

OWASP por estar libre de presiones corporativas facilita que OWASP proporcione información imparcial, práctica y redituable.

Entre los documentos con más éxito de OWASP se encuentra la guía OWASP y el ampliamente

adoptado documento de autoevaluación OWASP Top 10. Las herramientas OWASP más usadas incluyen el entorno de formación WebGoat, la herramienta de pruebas de penetración WebScarab y las utilidades de seguridad para entornos .NET OWASP DotNet. OWASP cuenta con unos 50 capítulos locales por todo el mundo y miles de participantes en las listas de correo del proyecto. OWASP ha organizado la serie de conferencias AppSec para mejorar la construcción de la comunidad de seguridad de aplicaciones web.

Periódicamente OWASP, tal como se muestra en (fig. 1), publica la lista de riesgos de seguridad más críticos de aplicaciones web, la cual se basa en información proveniente de 8 firmas especializadas en seguridad de aplicaciones. Se tienen en cuenta alrededor de 500,000 vulnerabilidades alrededor de cientos de organizaciones y miles de aplicaciones. Estas vulnerabilidades son priorizadas de acuerdo al nivel de explotación, detección e impacto estimado.

OWASP Top 10 – 2010 (Previous)	OWASP Top 10 – 2013 (New)
A1 – Injection	A1 – Injection
A3 – Broken Authentication and Session Management	A2 – Broken Authentication and Session Management
A2 – Cross-Site Scripting (XSS)	A3 – Cross-Site Scripting (XSS)
A4 – Insecure Direct Object References	A4 – Insecure Direct Object References
A6 – Security Misconfiguration	A5 – Security Misconfiguration
A7 – Insecure Cryptographic Storage – Merged with A9 →	A6 – Sensitive Data Exposure
A8 – Failure to Restrict URL Access – Broadened into →	A7 – Missing Function Level Access Control
A5 – Cross-Site Request Forgery (CSRF)	A8 – Cross-Site Request Forgery (CSRF)
<buried in A6: Security Misconfiguration>	A9 – Using Known Vulnerable Components
A10 – Unvalidated Redirects and Forwards	A10 – Unvalidated Redirects and Forwards
A9 – Insufficient Transport Layer Protection	Merged with 2010-A7 into new 2013-A6

Fig. 1. https://www.owasp.org/images/5/5f/OWASP_Top_10_-_2013_Final_-_Espa%C3%B1ol.pdf Las 10 vulnerabilidades más críticas de la aplicaciones web según OWASP.

III. EXPOSION DE LAS APLICACIONES

[1]Las aplicaciones exposición masiva a internet, debería ser desarrolladas considerando las vulnerabilidades que implica ser utilizada por cualquier persona en cualquier lugar del mundo, multiplicando las posibilidades de que de estos aplicativos sean explotadas por cualquiera de los siguientes tipos de ataques:

Denegación de Servicio: tienen como objetivo saturar los recursos de la víctima de forma tal que se inhabilita los servicios brindados por la misma.

[2]Desbordamientos de búfer: Uno de los primeros errores de seguridad de la historia de la informática del que se sacó provecho fue un desbordamiento de búfer y hoy en día continúan siguen siendo uno de los puntos débiles más peligrosos y comunes.

Un buffer de datos es un espacio de memoria en disco o en un instrumento digital reservada para el almacenamiento temporal de información, mientras que está esperando ser procesada, pero es muy importante que los programadores, cuando tengan que escribir datos en los búferes, no sobrepasen la capacidad permitida de los mismos. Si la cantidad de datos que se escriben supera el espacio del búfer asignado a tal efecto, se produce un desbordamiento del búfer. Si esto ocurre, los datos se escriben en partes de la memoria que pueden estar asignadas a otros fines.

En el peor de los casos, el desbordamiento de búfer contiene código malintencionado que entonces se ejecuta. Los intentos de explotar este tipo de vulnerabilidad pueden dar lugar a problemas como el bloqueo de la aplicación o que un atacante inserte y ejecute código malintencionado en el proceso de la aplicación.

Virus y gusanos: éstos, son los tipos más conocidos de software maligno que existen y se distinguen por la manera en que se propagan. Se definen como un programa que al ejecutarse se propaga a si mismo explotando vulnerabilidades en una red de para infectar otros equipos.

[1] http://www.deloittedce.cl/cde/wp-content/uploads/2014/03/Desarrollo-Seguro-Certificaci%C3%B3n-BCS-101-buguroo1.pdf

[2] http://msdn.microsoft.com/es-es/library/aa292190(v=vs.71).aspx

Backdoor o puerta trasera: Es un método para eludir los procedimientos habituales de autenticación al conectarse en una computadora o sistema. Los crackers suelen usar puertas traseras para asegurar el acceso remoto a una computadora, permaneciendo ocultos ante posibles inspecciones, utilizando troyanos, gusanos u otros métodos.

Cross-site scripting (XSS) son un tipo de ataque, en el que los scripts maliciosos son inyectados en los sitios web. Ataques XSS ocurren cuando un atacante utiliza una aplicación web para enviar código malicioso.

Un atacante puede usar XSS para enviar un script malicioso a un usuario desprevenido. El navegador del usuario final no tiene manera de saber que el guión no debe ser de confianza, y se ejecutará el script. Porque piensa que llegó el guión de una fuente confiable, el script malicioso puede acceder a las cookies, tokens de sesión u otra información sensible retenido por el navegador y se utiliza con ese sitio. Estos scripts pueden incluso volver a escribir el contenido de la página HTML.

SQL injection: es un método de infiltración de código intruso que se vale de una vulnerabilidad informática presente en una aplicación en el nivel de validación de las entradas para realizar consultas a una base de datos.

Los ataques por fuerza bruta suelen utilizarse para superar sistemas criptográficos como los protegidos con contraseñas. Los ciberdelincuentes utilizan programas informáticos para probar una gran cantidad de contraseñas y descifrar el mensaje o acceder al sistema

El origen de la vulnerabilidad radica en el incorrecto chequeo y/o filtrado de las variables utilizadas en un programa que contiene, o bien genera, código SQL.

IV. FACTORES LLEVAN A QUE EN UN DESARROLLADO SE PASA POR ALTO LA SEGURIDAD.

- Desde la óptica de un mundo utópico bajo los lineamientos de PMI, un proyecto de elaboración de software se planifica para que cada paso se ejecute con el tiempo necesario; pero la realidad es que en la mayoría de estos siempre llegan desfasados a la etapas de análisis, diseño e implementación, obligando esto a realizar desarrollos para “ayer", priorizando los resultados en términos de funcionalidad dejando relegada a un último puesto la seguridad de las aplicaciones.

- Los desarrolladores normalmente piensan que su aplicación solo va a ser utilizada por usuarios de buenas intenciones, pocas veces se detienen a pensar como atacante y a auditar su implementación desde ese punto de vista.

- Regularmente no se llevan a cabo estrictos ciclos de calidad, de principio a fin, basados en criterios de programación segura por encima de todo.

V. SUGERENCIAS PARA UN CÓDIGO MÁS SEGURO

Consejos para tener en cuenta en la etapa de análisis y diseño:

- Acordar la criticidad de la aplicación a desarrollar, para tener en cuenta el número de usuarios a la cuales está dirigida la aplicación y cómo será el manejo de sesiones para administrar la concurrencia y evitar en lo posible los ataques de denegación de servicio.

- Definir el perfil de la población objetivo: usuarios internos, externos, invitados o grupos.

- Especificar cuál va a ser la infraestructura de hardware y de red que va a soportar la aplicación para asegurar entre otras cosas la correcta y ágil administración de parches para el software de los servidores expuestos a internet para evitar sorpresas.

- Detallar niveles de seguridad a implementar ya que no se manejaran los mismos si se trata una aplicación bancaria o gubernamental a si es una aplicación interna de control de ingreso de empleados de una pymes.

Consejos en la etapa de Desarrollo:

- Cifrar la información sensible que viaja por la aplicación especialmente si se tienen integraciones con otras aplicaciones o fuentes, así como también el manejo de contraseñas.

- Evitar código quemado en y validaciones visibles en los HTML.

- Uso de validación, ya sean propias de la herramienta de desarrollo o sean personalizadas que normalmente se implementan para verificar que los datos almacenados en ellas son correctos, son útiles también como salvaguarda ante un SQL injection.

- Ocultar o personalizar los mensajes de errores: tanto en la etapa de implementación como en las pruebas se pueden detectar cuales excepciones presentan en la aplicación y bajo qué circunstancias, lo cual permite para controlarlas a través de mascaras que sean entendibles al usuario final y con ello evitar la presentación de los mensajes por defectos de la herramienta de desarrollo que proporciona a un posible atacante información vital del código web como rutas, configuraciones y extensiones.

- Limpiar los campos de capturas de datos que permiten la interacción con el usuario, así como los POST, GET, SERVER, COOKIE y REQUEST, ya que estos son vulnerables ataques mediante XSS o SQL injection permitiendo con ello el robo de datos e incluso pueden alterar las aplicaciones.

- Inicialización de variables, cada vez que creemos una variable debemos inicializarla con un valor, con el fin de evitar que los hacker utilicen una variable no inicializada para realizar sus ataques.

- Funciones de escape, empleándolas evitaremos en gran medida los ataques mediante SQL Injection.

- En lo posible deshabilitar la opción de “ver código fuente” del menú emergente de las paginas.

- Obvie redirecciones y reenvíos no validados. Errores en el tratamiento de redirecciones y uso de datos no confiables como destino.

- Evite el uso de componentes con vulnerabilidades conocidas tales como librerías, frameworks y otros módulos de software, que en muchas ocasiones funcionan con todos los privilegios. Si se ataca un componente vulnerable esto podría facilitar la intrusión en el servidor o una pérdida de datos.

- No utilizar [3]QueryStrings en las URL de la aplicación. Que permite la alteración de los parámetros que se le envían al servidor.

- Utilice los [4]captchas son una barrera de seguridad contra scripts de fuerza bruta para

[3] QueryString es un método para pasar valores entre páginas a través de la URL

probar masivamente contraseñas sobre un formulario, así como para verificar que quien está enviando el formulario es una persona y no una máquina con un script automático.

Aplicando estos sencillos consejos, se puede conseguir que el código sea un poco más seguro frente a un posible ataque de hackers.

VI. CONCLUSIONES

La sensibilización de la seguridad informantica no solo debe ir dirigida al usuario final en esta se debe involucrar al desarrollador para que tenga presente la óptica de hacker y con esto logre prever en gran medida los posibles ataques de los puede ser victima la aplicación que construye.

Se deben definir niveles de seguridad requeridos según la criticidad de la aplicación y su población objetivo.

La planificación de las actividades de implementación debe reales para realizar un proceso detallado de las técnicas de seguridad que se van a aplicar durante esta etapa.

La necesidad de una constante retroalimentación técnica de los tipos de ataque que salen a la luz cómo prevenirlos.

VII. REFERENCIAS

[1] https://www.owasp.org/index.php/Main_Page .
[2] http://www.pedroventura.com/php/ataques-xss-como-prevenirlos-en-php/
[3] http://msdn.microsoft.com/es-es/library
[4] https://www.linux-magazine.es/issue/62/008-009_InseguridadesLM62.pdf

Autora

Viviana Farina Figueroa Marulanda
Ingeniera de Sistemas
Estudiante Especialización en Seguridad Informática
Universidad Piloto de Colombia

[4] Captcha, Completely Automated Public Turing test to tell Computers and Humans Apart, que en español se puede traducir como "Prueba de Turing pública y automática para diferenciar máquinas y humanos".