RETOS DE LA SEGURIDAD INFORMÁTICA EN SERVIDORES NODE JS

Arias Melo Yeison Hernando
yariasm@gmail.com
Universidad Piloto de Colombia

Resumen— Desde sus inicios en 1995, JavaScript ha evolucionado a lo largo de los años para convertirse hoy en día en uno de los lenguajes de programación más utilizados del lado del cliente para el mejoramiento de la interfaz de usuario y las páginas web dinámicas, sin embargo, en el año 2009 nació Node JS con el objetivo de aprovechar el gran potencial de JavaScript, y situarlo del lado del servidor para hacer competencia directa a otros lenguajes de programación como PHP, JSP, ASP.net entre otros. Este acontecimiento, además de brindar nuevas alternativas a los desarrollos web, genera nuevos desafíos en el ámbito de la seguridad informática. Este documento exhibe las características propias de Node JS y presenta algunas de las técnicas o estándares más utilizados para aumentar la seguridad informática en la utilización de este tipo de servidores.

Índice de Términos— Lenguajes de programación, riesgos, seguridad de la información, Servidores web.

Abstract — Since its inception in 1995, JavaScript has evolved over the years to become today one of the most widely used client-side programming languages for improving user interface and dynamic web pages, in 2009 Node JS was born with the objective of taking advantage of the great potential of JavaScript, and placing it on the server side to make direct competition to other programming languages like PHP, JSP, ASP.net among others. This event, in addition to providing new alternatives to web developments, generates new challenges in the field of computer security. This document shows the characteristics of Node JS and presents some of the techniques or standards most used to increase the computer security in the use of this type of servers.

Keywords— Programming languages, risks, information security, Web servers.

I. INTRODUCCIÓN

En la actualidad, y debido al auge y crecimiento de los desarrollos web orientados a satisfacer necesidades organizacionales, los retos en cuanto a seguridad informática son cada vez mayores y los riesgos asociados a ello van desde la caída de nuestra página web informativa, hasta la pérdida de cliente y declive de la organización gracias a la pérdida de credibilidad.

Muchas empresas dedicadas al desarrollo de aplicaciones web, están enfocadas en la velocidad de carga de las páginas, una interfaz de usuario que permita una fácil navegación, practicidad para la realización de comercio electrónico y el SEO, el cual es importante para el posicionamiento de la marca y el aumento del número de visitantes a los portales. Estos puntos son muy importantes, pero muchos de ellos no tienen en cuenta el aspecto más importante de todos, la seguridad informática.

Hoy en día existen numerosos lenguajes de programación y técnicas que permiten a los desarrolladores agilizar sus proyectos y le dan al usuario final una interfaz llamativa, rápida e intuitiva para el acceso a los productos y/o servicios de la empresa. Tenemos el caso de CSS, el lenguaje de diseño gráfico por excelencia en el mundo, que permite al diseñador web crear paginas realmente llamativas. Otro lenguaje de programación indispensable en la actualidad en el desarrollo web es JavaScript, el cual mejora significativamente la experiencia de los usuarios y permite la creación de páginas dinámicas.

JavaScript ha trascendido más allá del dinamismo orientado al usuario final (client-side) y hoy en día muchos desarrolladores lo utilizan de lado del servidor (server-side), esto con el fin de optimizar el tiempo de respuesta de los servidores y así aumentar la concurrencia de los sitios. Es evidente que la utilización de esta nueva tecnología llama mucho la atención de personas con fines ilegales que siempre buscan grietas de seguridad que permitan la intrusión, robo, eliminación o modificación de la información. Las técnicas que nombraremos en este documento buscan minimizar los riesgos en seguridad informática asociados a la utilización de esta nueva tecnología y brindan a los desarrolladores una idea más clara del porqué es

importante tener en cuenta la seguridad informática antes de iniciar un nuevo desarrollo web.

II. CRECIMIENTO DE JAVASCRIPT

El lenguaje de programación JavaScript, inicialmente llamado Mocha, surge gracias a la aparición de Internet, su desarrollo se realizó en la empresa Netscape en el año de 1995 y fue concedido para la implementación de un nuevo lenguaje de programación del lado del cliente, este lenguaje fue adaptado en el navegador Netscape Navigator.

JavaScript fue desarrollado por Rendan Eichen tomando como base varios lenguajes de programación, entre ellos Java, C y self, actualmente es una marca registrada de Oracle Corporation y es usada con licencia por los productos creados por Netscape Communications y entidades actuales como la Fundación Mozilla.

En 1996, Microsoft decidió implementar su propia versión de JavaScript en su navegador Internet Explorer llamado JScript. Las diferencias entre éstas implementaciones condujeron a problemas y es por esto que nace el estándar ECMA-262. Este estándar dicta la base del lenguaje ECMAScript a través de su sintaxis, tipos, sentencias, palabras clave y reservadas, operadores y objetos, y sobre la cual se pueden construir distintas implementaciones. La versión JavaScript 1.3 fue la primera implementación completa del estándar ECMAScript.

III. NACIMIENTO DE ECMASCRIPT 6

Con base en la proposición realizada por la fundación Netscape Communications Corporation en 1996, para la realización del estándar para JavaScript, se inició el desarrollo de la primera versión de ECMAScript. Este lenguaje define un lenguaje de tipos dinámicos inspirado en otros lenguajes como Java y C. Soporta algunas características de la POO (programación orientada a objetos) mediante objetos basados en prototipos y pseudoclases.

ECMAScript versión 6 publicada en junio de 2015, editada por Allen Wirfs-Brock, añade diferentes novedades orientadas a dar mayor dinamismo en las plataformas web, algunas de estas novedades son:

- Arrow Functions: Conocidas como expresiones lambda en otros lenguajes de programación como Java y C#, permite la utilización del operador '=>' para realizar abreviaciones a funciones.
- Clases: Anteriormente las clases en ECMAScript se creaban de forma diferente, en esta nueva versión la sintaxis es más parecida a otros lenguajes igualmente orientados a objetos, esto con el fin de darle mayor facilidad de escritura a los desarrolladores acostumbrados a lenguajes como Java.
- Let y Const: La utilización de la función Let permite declarar variables para un bloque en particular, al terminar el bloque la variable deja de existir, esto permite evitar errores lógicos cuando alteramos una variable que no deberíamos.
- La utilización de la función Const permite declarar variables como constantes y previene que una variable declarada cambie de valor, evitando así modificaciones inesperadas.
- Template Strings: Esta nueva función, similar a la interpolación en otros lenguajes como Ruby, permite a los desarrolladores contener valores en variable de tipo string por medio de la utilización de un par de caracteres back-tick (`). Esta nueva función también permite abarcar múltiples líneas para el establecimiento de una variable string.
- Generadores: Los generadores son un tipo de función especial útiles para retornan una serie de valores con un algoritmo definido por el usuario.

IV. BENEFICIOS DE JAVASCRIPT

JavaScript se utiliza principalmente del lado del cliente, esto significa que se ejecuta en nuestros computadores más no en el servidor donde se encuentra la página web, esta característica le permite crear efectos y dinamismo a las páginas web, los navegadores modernos como Google

Chrome interpretan el código JavaScript integrado en cada sitio web, este código debe estar ajustado al estándar actual de ECMAScript.

Algunas de los beneficios más importantes de JavaScript utilizado del lado del cliente son los siguientes:

- Validación de formularios: uno de los principales beneficios de la utilización de JavaScript es el poder validar formularios en sitios web, muy útil si se desea que, por ejemplo, un determinado campo de texto debe tener un correo electrónico valido o simplemente deba ser requerido.
- Eventos sobre objetos en DOM: los objetos del DOM ('Document Object Model' o 'Modelo de Objetos del Documento') pueden ser interpretados y manipulados por JavaScript y agregar funcionalidades dependiendo de sus eventos, como, por ejemplo, al hacer clic encima de un botón o seleccionar una opción dentro de un listado. Estos eventos permiten a los desarrolladores un sinfín de utilidades con el fin de dar mayor dinamismo a la página web.
-
- Dinamismo: teniendo en cuenta que JavaScript puede ser ejecutado desde el ordenador del cliente, precisamente esta característica permite que una determinada acción no tenga que esperar una respuesta del servidor para ser ejecutada, esto permite, por ejemplo, cambiar toda la apariencia de una página web adaptándola a las necesidades del cliente.

a. Top de lenguajes de programación en 2016

El pasado 26 de julio de 2016, el Instituto de Ingeniería Eléctrica y Electrónica (IEEE) a través de IEEE Spectrum, publicó un estudio sobre los lenguajes de programación más populares del 2015. Para la realización de este estudio, IEEE Spectrum desarrolló una aplicación web la cual toma datos de diferentes fuentes, entre ellas: GitHub, IEEE Xplore o CareerBuilder, es por esto que aseguran que los resultados tienen unas bases fiables.

Según este estudio, JavaScript es el octavo lenguaje de programación más popular en el mundo, por detrás de Java, quien posee el primer lugar, C, Python, PHP entre otros, y por delante de lenguajes de programación importantes como Ruby o Matlab. El Top diez de los lenguajes de programación más populares es el siguiente:

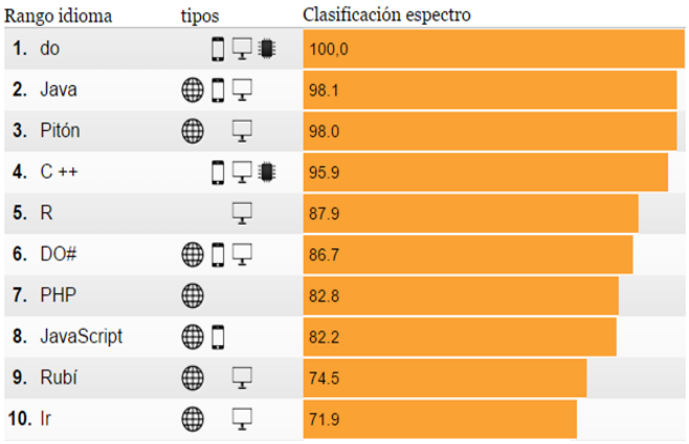
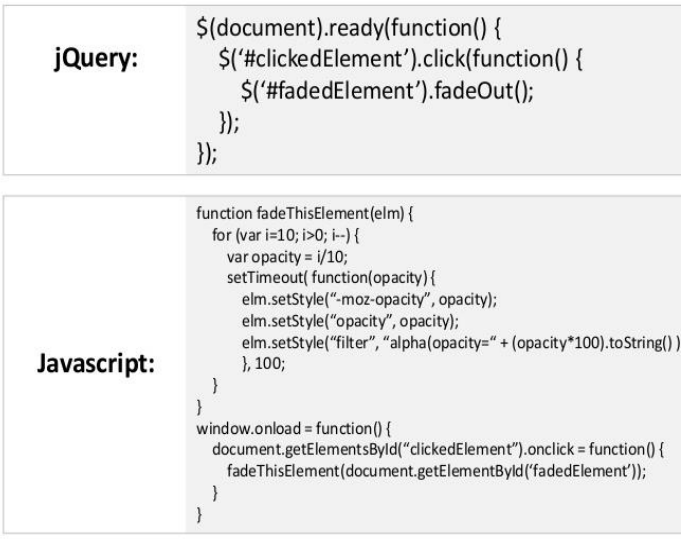


Fig. 1. Top 10 Lenguajes de programación más populares en 2015. [5]

V. NACIMIENTO DE JQUERY

Además de esto, JQuery permite integrar todo el potencial de JavaScript con menos código, simplificando la labor de los desarrolladores, a continuación, vemos un ejemplo del grado de simplicidad que permite a JQuery realizar tareas complejas con menos líneas en comparación con el



código nativo de JavaScript.

Fig. 2. Comparación del código nativo de JavaScript contra el código de JQuery. [24]

VI. RENDIMIENTO DE JQUERY

En términos de velocidad, JQuery es bastante rápido utilizando los navegadores modernos en una computadora de mediano o alto rendimiento, sin embargo, tanto JavaScript como JQuery poseen un rendimiento deficiente en navegadores y computadoras antiguas o desactualizadas. Para tener acceso al DOM, JavaScript suele ser más eficiente, pero puede llevar a errores que, por medio de la utilización de JQuery, podemos solucionar más rápidamente, esto significa que se codifica mucho más rápido y el tratamiento de errores es más eficiente con JQuery.

	Test	Ops/sec
jQuery ID Selector	`$("#testid");`	2,370,440 ±8.46% 100% slower
getElementById	`document.getElementById("testid")`	1,359,782,093 ±0.19% fastest

Fig. 3. Comparativa de velocidad al utilizar código nativo de JavaScript y utilizar código de JQuery. [25]

VII. VENTAJAS DE JQUERY

Las ventajas más notables derivadas de la utilización de JQuery, en comparación de la utilización de código nativo de JavaScript, son las siguientes:

- Compatibilidad con los diferentes navegadores del mercado: un ejemplo sencillo es el uso de la función. attr(), ya que si se usan las alternativas nativas el código podría generar errores en ciertos navegadores.
- Simplificación en operaciones generalmente complicadas: JQuery posee un sinnúmero de funciones que facilitan la escritura de código y disminuyen la probabilidad de errores, un ejemplo de ello es el uso de la función $.ajax(), la cual es muy compleja en código nativo de JavaScript.
- Manejo de elementos del DOM: tareas tan simples como el asignar eventos a elementos del DOM o seleccionar valores de los objetos pueden ser muy complicadas y diferentes entre navegadores, si no se tiene el conocimiento se puede escribir mal el código y ralentizar la página, JQuery brinda funciones que simplifican estas tareas y lo mejor es que funcionan para múltiples navegadores.
- Acceso a funciones futuras: funciones como. indexOf y bind son utilizadas en JavaScript nativo, sin embargo, estas funciones aún no están soportadas por algunas versiones de los navegadores, por medio de la utilización de JQuery es posible utilizarlas permitiendo su ejecución en navegadores no actualizados.

VIII. INTERACTIVIDAD MEDIANTE AJAX

Ajax nace mediante una iniciativa de Microsoft para la implementación de scripting remoto, la cual es una tecnología que permite a los scripts que se ejecutan en un navegador web intercambiar información con el servidor. Sin embargo, el término AJAX nace en el año 2005 a través del arquitecto de información Jesse James Garrett, con el objetivo base de poder cargar información de forma asincrónica sin necesidad de recargar el navegador web.

Esta técnica mejora notablemente la interactividad, velocidad y usabilidad en las aplicaciones web ya que es posible realizar cambios sobre las páginas web mediante la comunicación asíncrona con el servidor en segundo plano.

Ajax es una técnica válida para múltiples plataformas ya que está basada en estándares abiertos como JavaScript y Document Object Model (DOM).

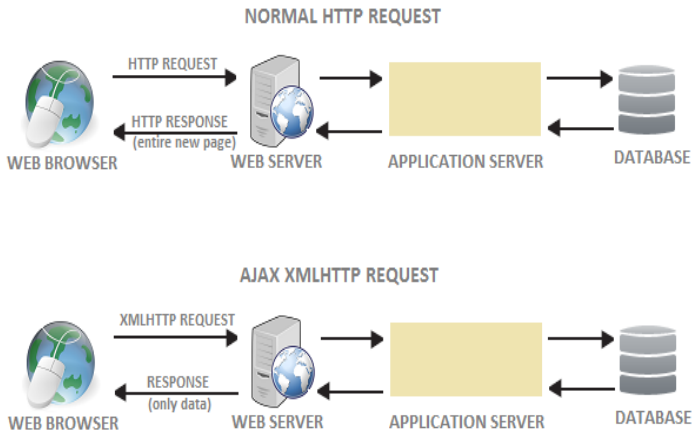


Fig. 4. Solicitud vía HTTP normal versus una solicitud Ajax.

IX. DESVENTAJAS Y RIESGOS DE JAVASCRIPT

Siendo JavaScript un lenguaje principalmente utilizado del lado del cliente, los fragmentos de código son descargados desde el servidor y ejecutados en el computador del cliente, esto significa que algún tipo de código malicioso también puede ser ejecutado en el computador del cliente con el fin de explotar alguna vulnerabilidad de seguridad de la información conocida en una aplicación web, navegador o hasta en el mismo sistema operativo. Al pasar de los años se han implementado numerosos estándares de seguridad con el fin de restringir la ejecución de código por parte de los navegadores, sin embargo, aún se puede ejecutar código que dañe, robe o destruya información del lado del cliente.

Otra desventaja encontrada con la utilización de JavaScript es el inevitable aumento de código en las páginas web, ya que, cuando un motor de búsqueda, por ejemplo, Google, intenta indexar un sitio web lo primero que encuentra son cientos o hasta miles de líneas de código JavaScript, dificultando la indexación y disminuyendo la eficiencia en el mismo. Para solucionar este inconveniente, se sugiere a los desarrolladores separar el código JavaScript en archivos de script con extensión (*.js) para que los motores de búsqueda puedan leer el contenido de calidad el sitio web, aumentando el posicionamiento en los buscadores.

En términos de seguridad informática, otro inconveniente es que el código JavaScript proveniente de los servidores son totalmente legibles para cualquier persona, esto significa que los atacantes podrían determinar el funcionamiento de ciertos eventos o funciones y manipularlos con el fin de extraer información o cambiar información enviadas desde o hacia el servidor.

Cross site scripting es un tipo de inseguridad informática en aplicaciones web que, junto con SQL Injection, son las más comunes en el mundo. XSS (como es denominado Cross-site scripting) permite a una tercera persona inyectar en páginas web visitadas por el usuario código JavaScript evitando medidas de control como la política del mismo origen.

XSS es un vector de ataque que puede ser utilizado para robar información confidencial, secuestrar sesiones de usuarios, y comprometer el navegador, subyugando la integridad del sistema.

En el siguiente gráfico, creado por Web Hacking Incident Database for 2011 (WHID) se puede apreciar claramente que, aunque existen muchos métodos de ataque diferentes a las páginas web, SQL Injection y XSS (Cross-site scripting) son los más populares. Para agregar a esto, muchos otros métodos de ataque, tales como divulgaciones de información, spoofing de contenido y credenciales robadas podrían ser efectos secundariamente luego de efectuar un ataque XSS.

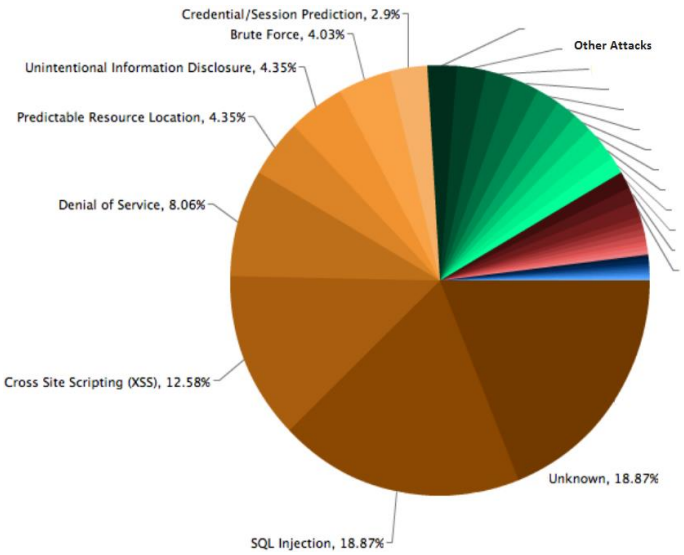


Fig. 5. Tipos de ataques informáticos a aplicaciones web. [16]

X. SEGURIDAD DE LA INFORMACIÓN EN JAVASCRIPT

JavaScript representa un gran potencial para los desarrolladores maliciosos que escriben scripts para ejecutarse en un cliente mediante la web. Los navegadores están preparados para evitar y contener este riesgo de dos maneras.

- Primera: los scripts deben ser ejecutados en una sandbox (palabra que del inglés significa caja de arena) en la cual solo se pueden ejecutar acciones relacionadas a internet, no tareas de programación de propósito general como crear archivos.
- Segunda: que los scripts están limitados por la regla del mismo origen: los scripts de una web no tienen acceso a información como nombres de usuario, contraseñas o cookies enviadas desde otra web.

La mayoría de los errores de software en JavaScript relacionados con la seguridad de la información son grietas en una de estas reglas. También hay que tener en cuenta que muchas vulnerabilidades presentes en los aplicativos web son errores de implementación por parte de los desarrolladores de los navegadores y de los desarrolladores de los aplicativos.

XI. NODE JS

Node JS es un entorno de ejecución para JavaScript construido en el motor de JavaScript V8 de Chrome, es empleado principalmente para la capa del servidor y está creado con el objetivo de ser útil en la creación de programas de red altamente escalables, como, por ejemplo, servidores web.

Node JS asegura ser altamente fiable y confiable, ya que todo el código escrito en Node JS que funcione bien en el servidor, funcionará bien en el lado de los clientes sea cual sea el sistema operativo utilizado, dado que la ejecución de código en el servidor quedará aislada en el servidor.

XII. VENTAJAS DE NODE JS

La principal característica de Node JS se encuentra en la programación asincrónica, la filosofía detrás de Node JS se centra en hacer aplicaciones que no bloqueen la línea de ejecución de código con respecto a entradas y salidas, de modo que los ciclos de procesamiento queden disponibles mientras se llevan a cabo las diferentes acciones.

Teniendo en cuenta que la base de Node JS es JavaScript, tenemos que hablar igualmente de la programación orientada a eventos, pero en Node JS funciona un poco diferente, aunque sigue el mismo concepto, por ejemplo, en JavaScript del lado del cliente tenemos objetos como ‘document’ o ‘body’, pero en Node JS no existen, pues estamos en el lado del servidor. Al contrario, los eventos que podremos encontrar del lado del servidor son por ejemplo un ‘uncaughtError ‘, que se produce cuando se encuentra un error por el cual un proceso ya no pueda continuar.

A continuación, encontramos una tabla de rendimiento de Node JS en comparación con PHP, uno de los lenguajes de programación orientado a servidores más utilizados en el mundo, en él podemos apreciar el rendimiento de ambos lenguajes al momento de ejecutar una secuencia de comandos equivalente.

Performance		
Number of iterations	Node.JS	PHP
100	2.00	0.16
10,000	3.00	9.52
1,000,000	13.00	1117.21
10,000,000	138.00	10461.29

Fig. 6. Comparación entre Node JS y PHP. [20]

Se puede apreciar que Node JS supera enormemente al lenguaje de programación PHP cuando el número de iteraciones es considerablemente alto, esto da cuenta del gran potencial que tiene Node JS y nos da una idea sobre el futuro en cuanto al desarrollo de aplicaciones web.

XIII. DESVENTAJAS DE NODE JS

Uno de los mayores inconvenientes en la utilización de Node JS es su inestabilidad entre versiones, ya que cuando se libera una nueva versión, suele cambiar en formas que rompen la

compatibilidad con su versión predecesora, lo que requiere que se apliquen cambios frecuentes en el código con el fin de mantener todo funcionando en las versiones actuales.

Otra desventaja actual de este lenguaje es la falta de librerías en general, si se desea, por ejemplo, implementar una librería para el procesamiento de imágenes, una interfaz de bases de datos madura o un analizador de XML, debido a que JavaScript no posee muchos años de experiencia del lado del servidor, este tipo de librerías todavía no existen o no están lo suficientemente maduras para brindar estabilidad en producción.

Debido a que Node JS es un entorno de ejecución reciente, su nivel de madurez lo hace susceptible a opiniones subjetivas debido a que es una cuestión bastante abierta.

XIV. SEGURIDAD EN NODE JS

Node JS se ha caracterizado por tener una seguridad envidiable con respecto a otros lenguajes de programación orientados al lado de servidor, sin embargo, se han detectado algunos problemas de seguridad que los desarrolladores deben tener en cuenta:

a. Propagación de malware a través del gestor de paquetes de Node JS

El gestor de paquetes de Node JS, llamado NPM, se implementó por defecto a partir de la versión 0.6.3 de Node JS y es el encargado de manejar las dependencias para una aplicación, Además, permite a los usuarios instalar aplicaciones Node.js que se encuentran en el repositorio dado. La propagación de malware a través de este gestor fue anunciada por Sam Saccone, ingeniero de Google, quien afirmó que un fallo detectado en el gestor de paquetes podría hacer tambalear todo el ecosistema JavaScript si lo utiliza un ciberdelincuente experimentado.

Sin embargo, para que esta vulnerabilidad pueda ser explotada, inicialmente el desarrollador deberá descargar el paquete que está equipado con el código malicioso, posterior a ello, el código malicioso deberá ser subido al servidor de producción y de esta manera ser ejecutado en el equipo servidor y descargado por los usuarios.

XV. SEGURIDAD EN NODEJS CON EL PROYECTO OWASP NODEGOAT

El proyecto OWASP NodeGoat proporciona un entorno para conocer cómo se funcionan los principales riesgos de seguridad de OWASP a las aplicaciones web desarrolladas con Node JS y cómo solucionarlas de manera efectiva.

Al ser ligero y eficiente, el desarrollo web con Node JS se está convirtiendo rápidamente en una plataforma de elección para la creación de aplicaciones web modernas, rápidas y escalables e intensivas en datos. Sin embargo, el desarrollo de aplicaciones web estables y resistentes en esta plataforma es muy dependiente de los desarrolladores debido a su configuración predeterminada mínima y opciones de arquitectura. El objetivo del proyecto OWASP NodeGoat es actuar como un recurso de aprendizaje que demuestra cómo los riesgos de seguridad Top 10 de OWASP se aplican a las aplicaciones web desarrolladas con Node JS y cómo abordarlas eficazmente. Incluye una aplicación web vulnerable y una guía tutorial acompañada.

XVI. PROYECTO OWASP TOP 10

El proyecto de OWASP top 10 está destinado a educar a los desarrolladores, diseñadores, arquitectos, gerentes y organizaciones sobre las consecuencias de las vulnerabilidades de seguridad más importantes en aplicaciones web.

El Top 10 provee técnicas básicas sobre cómo protegerse en estas áreas de alto riesgo y también provee orientación sobre los pasos a seguir.

Este proyecto tiene en cuenta los lenguajes de programación más comunes en la actualidad como Java, .NET, Python, PHP, Node JS entro otros.

El proyecto OWASP Top 10, se basa en 8 conjuntos de datos de 7 firmas especializadas en seguridad de aplicaciones, incluyendo 4 empresas

consultoras y 3 proveedores de herramientas. Estos datos abarcan más de 500.000 vulnerabilidades a través de cientos de organizaciones y miles de aplicaciones. Las vulnerabilidades del Top 10 son seleccionadas y

priorizadas de acuerdo a estos datos de prevalencia, en combinación con estimaciones consensuadas de explotabilidad, detectabilidad e impacto.

A continuación, veremos una breve descripción sobre las 10 vulnerabilidades descritas en el proyecto OWASP top 10. [21]

a. A1- Inyección

Las vulnerabilidades de inyección, como por ejemplo SQL Injection, suceden cuando datos no confiables son enviados a un intérprete como parte de un comando o consulta. Los datos hostiles del atacante pueden engañar al interprete en ejecutar comandos no intencionados o acceder datos no autorizados.

b. A2- Pérdida de autenticación y gestión de sesiones

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

c. A3- Secuencia de comandos en sitios cruzados (XSS)

Las vulnerabilidades XSS suceden cada vez que una aplicación web toma datos no confiables y los envía al navegador web sin una validación adecuada. 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 no deseado.

d. A4- Referencia directa insegura a objetos

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

e. A5- Configuración de seguridad Incorrecta

Una buena seguridad de la información requiere tener definida e implementada una configuración segura para la aplicación web, 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, incluidas las librerías de código utilizadas por la aplicación.

f. A6- 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 precauciones especiales en un intercambio de datos con el navegador.

g. A7- Ausencia de control de acceso a funciones

La mayoría de aplicaciones web verifican los derechos de acceso a nivel de función antes de hacer visible en la misma interfaz de usuario. A pesar de esto, las aplicaciones web requieren 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 apropiadas.

h. A8- Falsificación de peticiones en sitios cruzados (CSRF)

Un ataque CSRF obliga al navegador de una víctima autenticada a enviar una petición HTTP falsificada, incluyendo la sesión del usuario y cualquier otra información de autenticación incluida automáticamente, a una aplicación web vulnerable.

i. A9- Utilización de componentes con vulnerabilidades conocidas

Algunos componentes tales como librerías, 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.

j. A10- Redirecciones y reenvíos no validados

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 desino. 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.

XVII. CONCLUSIONES

Las tecnologías relacionadas con aplicaciones web están avanzando a pasos agigantados en los últimos años, los desarrolladores hoy en día, teniendo como base la experiencia de los usuarios, se están inclinando en tecnologías que permitan una mayor velocidad, mejor interfaz gráfica, dinamismo y eficiencia en el manejo de los recursos de los servidores. No obstante, una variable realmente importante como la seguridad se está dejando en un segundo plano, algo que está cambiando, pero aún requiere de una mayor campaña de sensibilización para que los desarrolladores entiendan y tomen medidas adecuadas.

Una de las herramientas que más furor ha causado en los últimos años es JavaScript, un lenguaje de programación que permite integrar mayor dinamismo a las aplicaciones web.

Debido a la necesidad de contar con aplicaciones asincrónicas, que permitan una mayor velocidad y experiencia al usuario final, fue desarrollado Node JS, un entorno en tiempo de ejecución basado en el motor V8 de Google que está revolucionando los aplicativos webs.

El OWASP (‘Open Web Application Security Project’ o ‘Proyecto abierto de seguridad de aplicaciones web’), teniendo en cuenta el crecimiento de Node JS, ha desarrollado el proyecto OWASP NodeGoat, por medio del cual se proporciona a los desarrolladores un entorno para conocer cómo se aplican los principales riesgos de seguridad de OWASP a las aplicaciones web desarrolladas con Node JS y cómo solucionarlas de manera efectiva.

REFERENCIAS

[1] Pedro Gutierrez (2010). ¿JavaScript tiene futuro en el lado del servidor?. [Online]. Disponible en: http://www.genbetadev.com/respuestas/javascript-tiene-futuro-en-el-lado-del-servidor

[2] librosweb.es. Breve historia de JavaScript. [Online]. Disponible en: http://librosweb.es/libro/javascript/capitulo_1/breve_historia.html

[3] javascript.com (Actualizado en 2016). Documentación oficial de JavaScript. [Online]. Disponible en: https://www.javascript.com/

[4] Jorge (Dic, 2011). La evolución de JavaScript. [Online]. Disponible en: http://tratandodeentenderlo.blogspot.com.co/2011/12/la-evolucion-de-javascript.html

[5] Nick Diakopoulos, Stephen Cass (2016). Interactive: The Top Programming Languages 2016. [Online]. Disponible en: http://spectrum.ieee.org/static/interactive-the-top-programming-languages-2016

[6] Neha Tayal (2014). JavaScript vs jQuery: A Quick Overview and Comparison. [Online]. Disponible en: http://www.lucemorker.com/blog/javascript-vs-jquery-quick-overview-and-comparison

[7] Eugenio duarte (2012). ¿Qué Es JavaScript? Ventajas Y Desventajas. [Online]. Disponible en: http://blog.capacityacademy.com/2012/10/19/que-es-javascript-ventajas-y-desventajas/

[8] Acunetix (2011). Cross Site Scripting Attack. [Online]. Disponible en: https://www.acunetix.cz/websitesecurity/cross-site-scripting/

[9] Wikipedia (Actualizado el 11 jun 2013). Que es JScript. [Online]. Disponible en: https://es.wikipedia.org/wiki/JScript

[10] Carlos Azaustre (2015). aprende ecmascript 6 (es6 o es2015), el nuevo estándar de javascript. [Online]. Disponible en: https://carlosazaustre.es/blog/ecmascript-6-el-nuevo-estandar-de-javascript/

[11] Enrique Munguía (2015). Novedades en EcmaScript6. [Online].
Disponible en:
https://devcode.la/tutoriales/novedades-ecmascript6/

[12] Wikipedia (Actualizado el 2 sep 2016). Que es Ajax. [Online].
Disponible en:
https://es.wikipedia.org/wiki/AJAX

[13] Miguel Angel Alvarez (2013). Características destacables de NodeJS. [Online].
Disponible en:
http://www.desarrolloweb.com/articulos/caracteristicas-nodejs.html

[14] nodejs.org (Actualizado a 2016). Documentación oficial de Node JS. [Online].
Disponible en:
https://nodejs.org/es/

[15] Chad dotson (2014). Node.js vs Python vs PyPy – A Simple Performance Comparison – Updated. [Online].
Disponible en:
http://www.cdotson.com/2014/12/node-js-vs-python-vs-pypy-a-simple-performance-comparison-updated/

[16] Technology Timely (2011). Web Hacking Threats. [Online].
Disponible en:
https://techtimely.wordpress.com/2011/04/22/web-hacking-threats/

[17] softwero.com (2014). Las Desventajas de Usar Node.js en el Servidor. [Online].
Disponible en:
http://www.softwero.com/2014/08/las-desventajas-de-usar-nodejs-en-el.html

[18] Adrián Crespo (2016). El gestor de paquetes de Node.js permite propagar malware. [Online].
Disponible en:
http://www.redeszone.net/2016/03/28/gestor-paquetes-node-js-permite-propagar-malware/

[19] OWASP (sep, 2016). OWASP Node js Goat Project. [Online].
Disponible en:
https://www.owasp.org/index.php/OWASP_Node_js_Goat_Project

[20] Abhishek Jaiswal (2014). Node.JS vs Traditional Scripting Languages. [Online].
Disponible en:
http://www.c-sharpcorner.com/UploadFile/2072a9/node-js-vs-traditional-scripting-languages/

[21] OWASP (2013). OWASP Top 10 2013. [Online].
Disponible en:
https://www.owasp.org/images/5/5f/OWASP_Top_10_-_2013_Final_-_Espa%C3%B1ol.pdf

[22] codigoverde.com (2015). Los 7 retos de la Seguridad Informática y cómo enfrentarlos. [Online].
Disponible en:
http://codigoverde.com/los-7-retos-de-la-seguridad-informatica-y-como-enfrentarlos/

[23] socialmood (2016). ¿Qué es el SEO y por qué lo necesito? [Online].
Disponible en:
https://www.40defiebre.com/guia-seo/que-es-seo-por-que-necesito/

[24] Codecademy (2013). Lesson 3 Jquery. [Online].
Disponible en:
http://www.slideshare.net/bobrenjc93/lesson-203-18-sep131500ay

[25] mannheim-design.de(2015). Sder performance-vergleich: jquery vs. Javascript. [Online].
Disponible en:
http://mannheim-design.de/der-performance-vergleich-jquery-vs-javascript-2/