Revista Cubana de Ciencias Informáticas
Vol. x, No. x, Mes x-y, 2013
ISSN: 2227-1899 | RNPS: 2301
Pág. x-y
http://rcci.uci.cu

Tipo de artículo: Artículo original

Temática: Seguridad Informática

Recibido: dd/mm/aa | Aceptado: dd/mm/aa | Publicado: dd/mm/aa

Automatización de pruebas de seguridad en aplicaciones web basadas en CMS

Automation of security tests in CMS-based web applications

Leonardo Aguilera Blanco [1*], Henry Raúl González Brito [2]

[1]Leonardo Aguilera Blanco, Universidad de las Ciencias Informáticas (UCI), La Habana, Cuba, laguilera@uci.cu

[2]Henry Raúl González Brito, Universidad de las Ciencias Informáticas (UCI), La Habana, Cuba, henryraul@uci.cu

[*] Autor para correspondencia: laguilera@uci.cu

Resumen

Las tecnologías web han evolucionado hasta permitir que los desarrolladores puedan crear nuevas experiencias web. Los sistemas de gestión de contenidos facilitan la creación de contenido web por lo que, en la mayoría de los casos, los usuarios carecen de un conocimiento experto de la tecnología en sí, y mucho menos de los problemas de seguridad en ella. Para complicar el asunto, las vulnerabilidades de CMS son objetivos atractivos para posibles atacantes. Un análisis de seguridad de CMS populares de código abierto reveló importantes agujeros de seguridad. Estas vulnerabilidades dejan las aplicaciones y sus usuarios no expertos abiertos a la explotación. La mayoría de estas vulnerabilidades pueden ser corregidas durante las fases iniciales del desarrollo de software, sin embargo, son detectadas más adelante en su ciclo de vida por lo que retrasa la publicación de parches y produciendo un aumento de los costes de desarrollo. Por ello el objetivo de esta investigación fue formalizar un paquete de pruebas de seguridad mediante Gherkin a partir de las pruebas automatizables en los Sistemas de Gestión de Contenidos utilizando la metodología de Behavior Driven Development. Dicha metodología permite una mejor comunicación entre el equipo de desarrollo de software debido a que las pruebas de seguridad son redactadas utilizando los requisitos específicos de

software y en lenguaje natural, por lo que ambas partes pueden entender y mantener las pruebas sin poseer altos conocimientos de programación.

Palabras clave: Automatización, CMS, Pruebas de seguridad, Seguridad Informática

Abstract

Web technologies have evolved to allow developers to create new web experiences. Content management systems facilitate the creation of web content so, in most cases, users lack an expert knowledge of the technology itself, much less of the security problems in it. To complicate matters, CMS vulnerabilities are attractive targets for would-be attackers. A security scan of popular open source CMS revealed significant security holes. These vulnerabilities leave applications and their non-expert users open to exploitation. Most of these vulnerabilities can be corrected during the initial phases of software development, however, they are detected later in its life cycle, thus delaying the publication of patches and producing an increase in development costs. Therefore, the objective of this research was to formalize a security testing package using Gherkin from automatable tests in Content Management Systems using the Behavior Driven Development methodology. This methodology allows better communication between the software development team because the security tests are written using the specific software requirements and in natural language, so that both parties can understand and maintain the tests without having high programming knowledge.

Keywords: *Automatization, CMS, Informatic security, Security Testing*

Introducción

Las aplicaciones web son la base para la informatización de la sociedad moderna. En la actualidad no se concibe una organización que no tenga presencia en Internet mediante un portal web. Los Sistemas de Gestión de Contenidos (CMS) facilitan la creación de estas aplicaciones web, los cuales brindan un marco de trabajo, basado en arquitecturas, componentes para extender y personalizar funcionalidades y su comportamiento visual. Son creados a través de una comunidad de programadores, diseñadores y equipos de experiencia en estos sistemas, lo que evita que las empresas requieran contratar desarrolladores para realizar estas aplicaciones y poder concentrarse en su objeto social (Barnes, Goodwin y Vidgen 2001).

En la actualidad los CMS representan un alto porciento de las tecnologías utilizadas en Internet. Por ejemplo, W3Techs, BuiltWith y Statista muestran que el 70% de la web son CMS (BuiltWith® 2019; W3Techs 2019). Entre estos destacan los CMS Wordpress, Drupal y Joomla que constituyen el 40% de los CMS. Por tanto, los CMS a los que se hace referencia en la investigación son estos y no otros de carácter más específicos.

Debido a su extensa utilización son objetivos atractivos para los ciberdelincuentes, que buscan aprovechar las vulnerabilidades presentes, las cuales son fallas o debilidades en el diseño, la implementación, el funcionamiento o la gestión de un sistema, que pueden ser explotados con la finalidad de violar su política de seguridad. Esto ha producido un incremento en los incidentes de ciberseguridad a nivel mundial que involucran esta tecnología, con el agravante que cuando se descubre una vulnerabilidad de seguridad, estos son masivamente explotados a través de las botnets controladas por grupos de ciberdelincuentes (Petrică et al. 2017; Walden et al. 2009; Shteiman 2014), llegándose a cuantificar decenas de miles de portales web comprometidos diariamente. Ejemplo de ellos son los casos que tuvieron lugar con las vulnerabilidades CVE-2017-5487 (WordPress), CVE-2014-3704, CVE-2018-7600, CVE-2018-7600, CVE-2018-7602 (Drupal) y CVE-2015-8562, CVE-2017-8917 (Joomla) (Franklin, Wergin y Booth 2014).

En el desarrollo de aplicaciones web se realizan pruebas de seguridad para detectar estas vulnerabilidades (González Brito y Montesino Perurena 2018; Laverdière y Merlo 2018), sin embargo, estas aplicaciones debido a su naturaleza centralizada se encuentran en constantes cambios desde que se inicia el ciclo de desarrollo de software (Rodas-Silva et al. 2019). Para los CMS cuyo objetivo es la entrega ágil de portales web realizar estas pruebas manteniendo la frecuencia de cambios producidos no resulta posible debido a lo explicado a continuación:

- El proceso de detectar vulnerabilidades de forma automatizada en los CMS es deficiente ya que es realizado a través de herramientas genéricas que solo encuentran problemas comunes, omitiendo problemas específicos, los cuales, en ocasiones, constituyen los más costosos debido a que son los más intrínsecos con el funcionamiento lógico de la aplicación, como es el caso de los paneles de administración o la participación de APIs (Application Program Interface) de terceros para el consumo de datos para el negocio (Mohammed 2016).
- Estas herramientas generan reportes de seguridad que para su comprensión es necesario cierto grado de conocimientos en temas de seguridad informática, y en las aplicaciones web basadas en CMS se agudizan estos problemas debido a que muy a menudo son utilizados por personal con muy pocos conocimientos técnicos, por lo que el personal de la aplicación web debe desviar sus esfuerzos en investigarlos (Felderer et al. 2016).
- Para realizar pruebas más específicas deben ser aplicadas manualmente y debido al nivel de complejidad de estas pruebas requieren de personal especializado que comúnmente no forma parte del equipo de desarrollo, por lo cual deben contratar este servicio o consultar con otro departamento para realizar las pruebas de seguridad, siendo las vulnerabilidades detectadas en etapas tardías del ciclo de desarrollo de software, aumentando el costo de tiempo y recursos en ser reparada (Weidman 2014).

Por estas cuestiones el proceso de pruebas de seguridad es extenso, costoso y resulta complejo de automatizar debido a que no es posible realizar una herramienta automatizada para cada aplicación web específica. Por lo que es necesario utilizar pruebas de seguridad con métodos que garanticen un continuo flujo de trabajo que ocurra de forma paralela al desarrollo del software.

Behavior Driven Development (BDD) es una metodología de desarrollo conducido que utiliza pruebas unitarias para conducir el desarrollo (Menon 2016). Esta metodología fortalece la colaboración incrementando la participación del equipo con el ciclo de desarrollo y un lenguaje natural utilizado para escribir los escenarios de pruebas que mejora la visibilidad del proyecto, el valor del negocio y son satisfechas las necesidades del cliente debido a que todo el desarrollo se concentra alrededor de cómo lo visualizaría el usuario (Irshad, Britto y Petersen 2021; Rahman y Gao 2015).

Se facilita la predicción de errores y brinda una mayor confianza en los desarrolladores en el proceso de pruebas. Al mejorar la calidad del producto se reduce el costo y riesgo del proyecto (Rahman y Gao 2015). Cucumber es una herramienta de pruebas y un framework que soporta BDD, describe las acciones de la aplicación utilizando un Domain Specific Language (DSL) con una gramática simple a través de Gherkin (dos Santos y Vilain 2018).

Por lo que, en correspondencia con lo investigado anteriormente, el objetivo de la investigación fue formalizar un paquete de pruebas de seguridad mediante Gherkin a partir de las pruebas automatizables en CMS.

Materiales y métodos

Para la realización de la investigación se emplearon los siguientes métodos de investigación:

- Analítico-Sintético: se utiliza para analizar teorías, documentos y aplicaciones utilizadas en la implementación de la aplicación de pruebas automatizadas en CMS.
- Experimentación: se emplea al comprobar, recreando las condiciones de uso de la aplicación a través de pruebas, su funcionamiento y valor para cumplir los requisitos.

Resultados y discusión

La automatización de las pruebas en aplicaciones web se realiza a través de tecnologías que permiten escribir interacciones programáticas hacia el sistema bajo prueba (SUT, siglas en inglés) para facilitar codificación de los casos de pruebas. Uno de estos sistemas es el proyecto Selenium de código abierto, el cual se ha convertido el

estándar para la automatización de pruebas en aplicaciones web. Esta tecnología interactúa con la aplicación web a través del navegador y el Document Object Model (DOM) (Wang et al. 2018).

Utilizando Gherkin estas pruebas son redactadas utilizando lenguaje natural para luego ser interpretadas por cualquier persona y pueda modificar los datos sin poseer altos conocimientos sobre programación, volviendo las pruebas más sencillas de mantener. Otra ventaja de utilizar estos paquetes de pruebas es que son directamente creados a partir de los requisitos de seguridad para las aplicaciones web, por lo que son fácilmente verificables y brindan mayor comunicación entre el negocio y la producción del sistema. Además, son reutilizables para todas aquellas aplicaciones que se asemejan al SUT.

Las pruebas de seguridad deben ser redactadas por especialistas de seguridad informática para luego ser traducidas en lenguajes de programación, convirtiendo las mismas en pruebas automatizadas. A continuación se presentan las pruebas de seguridad redactadas en lenguaje Gherkin para aplicaciones web basadas en Sistemas de Gestión de Contenidos.

Eliminar los archivos residuales del proceso de instalación

Las instalaciones de los componentes de un CMS tienen un conjunto de archivos que describen las versiones y características de estos, así como algunos elementos de configuración predefinidos. Aunque su contenido es de interés para los administradores, no es adecuado que estén desplegados en el servidor de producción y mucho menos si el portal está publicado en Internet porque dentro de las informaciones brindadas se expresan claramente tecnologías, ramas, números de versiones y definitivamente no contienen ningún tipo de código ejecutable o brindan funcionalidad alguna. A continuación se presenta una prueba de seguridad utilizando Gherkin para automatizar la comprobación de estos archivos. Esta prueba se realiza con el objetivo impedir la exposición de información sobre la tecnología base, mediante la eliminación de archivos residuales del proceso de instalación.

Scenario Outline: Reconocimiento a través de archivos residuales

Given Realizo la conexión al CMS *<cms>* con archivo *<archivo>*

When Compruebo el contenido del archivo

Then Determino la tecnología a la cual pertenece el archivo residual

Examples:

cms	archivo
Wordpress	readme.html
Wordpress	license.txt
Wordpress	licencia.txt
Wordpress	wp-config-sample.php
Drupal	INSTALL.txt
Drupal	README.txt
Drupal	LICENSE.txt
Drupal	core/CHANGELOG.txt
Joomla	README.txt
Joomla	LICENSE.txt
Joomla	htaccess.txt
Joomla	robots.txt.dist
Joomla	web.config.txt
END	EOF

Supresión del metadato Generator.

El metadato Generator es el principal responsable de informar la versión del núcleo de los CMS, información muy valiosa para los ciberdelincuentes, los cuales lo utilizan para acotar el rango de exploit que son necesarios utilizar para evadir las medidas de seguridad. Esta prueba se realiza con el objetivo de impedir la exposición de información sobre la tecnología base mediante la eliminación de los metadatos correspondientes a recursos HTML.

Scenario Outline: Reconocimiento a través del metadato Generator

Given Realizo la conexión al CMS *<cms>* con Generator *<generator>*

When Busco el Generator en el contenido del html

Then Determino la tecnología a la cual pertenece el generator

Examples:

cms	generator
Wordpress	<meta name="generator"
Drupal	generatorDrupal
Joomla	generatorJoomla
END	EOF

Supresión lógica de funciones XML-RPC

XML-RPC es un protocolo desarrollado en los años noventa para realizar llamadas a funciones remotas (RPC o Remote Procedure Call) codificadas a través de XML. Su incorporación en WordPress data desde sus mismos orígenes y se emplea a modo de API para la ejecución de diversas funciones como mecanismo paralelo de gestión, pero también tiene la opción de instalarse mediante componentes en otros CMS. Dos funciones en específico han sido reportadas en años recientes como vector explotado por los ciberdelincuentes, estas son:

- **system.multicall:** Su objetivo es recibir y ejecutar como parámetro lotes de llamadas de funciones XML-RPC, las cuales pueden alcanzar cifras de 300, 400 e incluso algunos autores han reportado mil funciones en una sola petición HTTP. Los ciberdelincuentes lo utilizan para realizar ataques de fuerza bruta amplificados mediante el empleo de funciones que requieran autenticación como wp.getUsersBlogs.

- **pingback.ping:** Es una función que permite enviarle una notificación a una aplicación web cuando es referenciada en los contenidos publicados. La aplicación web referenciada tratará de descargar el contenido desde la cual la están referenciando. Esta petición puede ser manipulada de manera tal que un adversario obligue a la aplicación web a enviarle un *pingback* a una tercera, logrando un ataque de denegación de servicios si repite este proceso con decenas de aplicaciones web vulnerables a modo de botnet temporal.

Para comprobar el servicio XML-RPC es necesario enviar una petición HTTP enviando un archivo XML en donde se comprueban los métodos system.multicall y pingback.ping para conocer si se encuentran disponibles, en caso de lo estén, la prueba de seguridad es fallida.

Scenario Outline: Comprobar métodos del XML-RPC

Given Realizo la conexión al XML-RPC

When Busco los métodos *<metodo>* a desactivar

Then Determino si están activos

Examples:

metodo
system.multicall
pingback.ping
pingback.extensions.getPingbacks
END

Cambiar la ubicación de archivos de configuración.

Algunos archivos en los CMS contienen datos de configuración con credenciales de usuarios, contraseñas, claves sal, puertos, direcciones IP de servidores y otras informaciones de interés para un posible adversario, las cuales pueden servir también para escalar privilegios y afectar otros sistemas que compartan, por ejemplo, el mismo servidor de base de datos. Los adversarios, durante el diseño de malware, al conocer la ubicación por defecto de estos archivos, podrán aprovecharse mejor de vulnerabilidades como las de inclusión de recursos y de directorio transversal para exponerlos. Con esta prueba se puede impedir la exposición de información interna del portal mediante el bloqueo de la indexación de los directorios.

Scenario Outline: Reconocimiento a través de archivos de configuración

Given Realizo la conexión al CMS *<cms>* hacia el archivo *<archivo>*

When Compruebo que existe el archivo

Then Determino la tecnología a la cual pertenece el archivo de configuración

Examples:

cms	archivo
Wordpress	wp-config.php
Wordpress	wp-admin
Wordpress	wp-content
Drupal	sites/default/settings.php
Joomla	configuration.php
END	EOF

Evitar la navegación de directorios.

Los servidores web mal configurados permiten la exploración del sistema de archivos a través del navegador web. En este caso un adversario puede analizar la estructura del portal y encontrar información valiosa como la versión del servidor web, copias de seguridad que no fueron procesadas, entre otros datos útiles para la búsqueda posterior de exploits. Se realiza una prueba de seguridad en donde se acceden a directorios específicos en los que se listan la estructura del CMS debido a que no cuentan con un archivo *index*.

Scenario Outline: Reconocimiento a través de la navegación por directorios

Given Realizo la conexión al CMS *<cms>* hacia la ruta *<ruta>*

When Compruebo que existe el contenido del HTML

Then Determino si permite la navegación por directorios

Examples:

cms	ruta
Wordpress	wp-content
Wordpress	wp-admin

Revista Cubana de Ciencias Informáticas
Vol. x, No. x, Mes x-y, 2013
ISSN: 2227-1899 | RNPS: 2301
Pág. x-y
http://rcci.uci.cu

| Wordpress | wp-includes |

| Drupal | sites/default |

| Joomla | configuration |

| END | EOF |

Evitar la enumeración de credenciales de usuarios

La enumeración de usuarios es un proceso mediante el cual los adversarios comprueban la existencia o simplemente recolectan los nombres de credenciales de acceso para luego emplearlas en el lanzamiento de ataques de fuerza bruta de tipo diccionario contra los canales de autenticación habilitados, ya sean paneles y formularios de autenticación web o distintos tipos de APIs.

El principal riesgo se produce cuando se presentan funcionalidades que permiten automatizar la enumeración de las credenciales, por ejemplo:

- La existencia de identificadores de usuarios secuenciales o fácilmente predecibles y utilizados como parámetros de la URL.
- Respuestas con mensajes o códigos HTTP diferenciados cuando el nombre de usuario introducido es incorrecto y cuando la contraseña es incorrecta.
- Utilización de nombres frecuentes tales como administrador, webmaster, editor, redactor, etc.

A través de esta prueba se disminuyen las posibilidades de éxito de los ataques de diccionario, mediante el bloqueo de la automatización del proceso de recolección de credenciales de usuarios.

Scenario Outline: Enumeración de usuarios a través de la respuesta del panel de autenticación

Given Realizo la conexión hacia la ruta de autenticación *<ruta>*

When Compruebo el mensaje fallido de autenticación

Then No debe mostrarme el mensaje *<mensaje>*

Examples:

| cms | mensaje | ruta |

Grupo Editorial “Ediciones Futuro”Universidad de las Ciencias Informáticas. La Habana, Cuba
rcci@uci.cu

Revista Cubana de Ciencias Informáticas
Vol. x, No. x, Mes x-y, 2013
ISSN: 2227-1899 | RNPS: 2301
Pág. x-y
http://rcci.uci.cu

| Wordpress | Invalid username | wp-login.php |

| Wordpress | The password you entered for the username usernametest123123 is incorrect | wp-login.php |

| END | EOF | EOF |

Scenario Outline: Enumeración de usuarios a través de la ruta Author

Given Realizo la conexión hacia el sitio web del CMS *<cms>* en la ruta *<ruta>* con el rango *<rango>*

Then Debe mostrarme el titulo *<titulo>*

Examples:

| cms | ruta | titulo | rango |

| Wordpress | ?author= | Page not found | 1-10 |

| END | EOF | EOF | END |

Incorporar encabezados de respuesta HTTP de seguridad

El protocolo HTTP contiene campos de encabezados para proveer mejoras de seguridad de las transacciones que se establecen entre el agente de usuario y el servidor web. A pesar de ello, en muchos casos los servidores y aplicaciones web no los incorporan por defecto en las respuestas enviadas:

- Strict-Transport-Security: Instruye al agente de usuario para que la conexión se realice a través de HTTPS en lugar de HTTP. Existen servidores web que tienen habilitado ambas conexiones, dejando en manos del usuario la responsabilidad de indicar la forma de hacerlo.
- X-Frame-Options: Especifica que no se pueden embeber las aplicaciones web en una etiqueta HTML frame. Esto garantiza en gran medida la protección ante ataques de secuestros de clic, los cuales se utilizan por ejemplo para el robo de credenciales de usuarios.
- X-XSS-Protection: Contribuye a evitar los ataques de tipo XSS mediante la activación de filtros que poseen los navegadores web.

Grupo Editorial “Ediciones Futuro”Universidad de las Ciencias Informáticas. La Habana, Cuba
rcci@uci.cu

Revista Cubana de Ciencias Informáticas
Vol. x, No. x, Mes x-y, 2013
ISSN: 2227-1899 | RNPS: 2301
Pág. x-y
http://rcci.uci.cu

- X-Content-Type-Options: Instruye al navegador web que no cargue hojas de estilo ni los scripts dañinos basados en la confusión de tipos MIME (Multipurpose Internet Mail Extensions).

Saltar esta prueba exitosamente permite reducir el riesgo de manipulación y mal uso de transacciones HTTP mediante la inclusión de campos de encabezado de seguridad en las respuestas del servidor web.

Scenario Outline: Encabezados de respuesta de seguridad

Given Realizo la conexión hacia el sitio web

Then Debe mostrarme el encabezado `<encabezado>`

Examples:

encabezado
X-Frame-Options
Strict-Transport-Security
X-Content-Type-Options
X-XSS-Protection
END

Gestionar el archivo robots.txt

Un archivo robots.txt solo debe hacer referencia a rutas y recursos públicos que no deba ser indexado por los bots. No tiene sentido establecer indicaciones para aquellos protegidos por contraseña y otros mecanismos de seguridad, a los cuales los bots no pueden llegar. El archivo robots.txt constituye “solamente” una invitación a no realizar determinadas acciones, no es ni pretende ser un mecanismo de control de acceso a los recursos de una aplicación web.

Esto garantiza que los datos del archivo robots.txt no comprometan la seguridad, mediante la aplicación de configuraciones efectivas alineadas a las características del portal.

Scenario Outline: Detección de rutas a través del archivo robots.txt

Given Realizo la conexión hacia el sitio web del CMS `<cms>` hacia el archivo "robots.txt"

Grupo Editorial “Ediciones Futuro”Universidad de las Ciencias Informáticas. La Habana, Cuba
rcci@uci.cu

Revista Cubana de Ciencias Informáticas
Vol. x, No. x, Mes x-y, 2013
ISSN: 2227-1899 | RNPS: 2301
Pág. x-y
http://rcci.uci.cu

Then No debe mostrarme la ruta `<ruta>`

Examples:

cms	ruta
Wordpress	readme.html
Wordpress	license.txt
Wordpress	licencia.txt
Drupal	INSTALL.txt
Drupal	README.txt
Drupal	LICENSE.txt
Drupal	CHANGELOG.txt
Joomla	README.txt
Joomla	LICENSE.txt
Joomla	htaccess.txt
Joomla	robots.txt.dist
Joomla	web.config
Wordpress	wp-content
Wordpress	wp-admin
Wordpress	wp-includes
Drupal	sites/default
Joomla	configuration
END	EOF

Grupo Editorial “Ediciones Futuro”Universidad de las Ciencias Informáticas. La Habana, Cuba
rcci@uci.cu

Conclusiones

Se presentaron un conjunto de pruebas de seguridad que permiten detectar fallas y vulnerabilidades comunes en las aplicaciones web de manera automatizadas a través del lenguaje Gherkin. Con la propuesta de pruebas de seguridad los desarrolladores de aplicaciones web basadas en Sistemas de Gestión de Contenidos pueden:

- Eludir los ataques automatizados más comunes.
- Disminuir la superficie de ataque.
- Minimizar los ataques de fuerza bruta contra paneles de autenticación.
- Aumentar la seguridad de las transacciones HTTP.
- Evitar la enumeración de usuario.
- Impedir el reconocimiento de la estructura del portal.

Las pruebas de seguridad propuestas han sido aplicadas y probadas en aplicaciones publicadas en Internet, las cuales, han permitido detectar con éxito debilidades existentes en las aplicaciones y ayudar a prevenir su explotación por ciberatacantes.

Como trabajo futuro se propone el desarrollo de una aplicación que permita solucionar aquellas pruebas de seguridad fallidas, realizando las principales recomendaciones de seguridad según la tecnología correspondiente y repare la vulnerabilidad detectada de manera automatizada.

Referencias

BARNES, S., GOODWIN, S. y VIDGEN, R., 2001. Web content management. *BLED Proceedings*, pp. 47.

BUILTWITH®, 2019. *CMS Usage Distribution in the Top 1 Million Sites* [en línea]. 2019. S.l.: s.n. Disponible en: https://trends.builtwith.com/cms.

DOS SANTOS, E.C. y VILAIN, P., 2018. Automated acceptance tests as software requirements: An experiment to compare the applicability of fit tables and Gherkin language. *Lecture Notes in Business Information Processing*. S.l.: s.n., ISBN 9783319916019. DOI 10.1007/978-3-319-91602-6_7.

FELDERER, M., BÜCHLER, M., JOHNS, M., BRUCKER, A.D., BREU, R. y PRETSCHNER, A., 2016. Security Testing: A Survey. *Advances in Computers* [en línea], vol. 101, pp. 1-51. [Consulta: 1 febrero 2020]. ISSN 0065-2458. DOI 10.1016/BS.ADCOM.2015.11.003. Disponible en: https://www.sciencedirect.com/science/article/pii/S0065245815000649.

FRANKLIN, J., WERGIN, C. y BOOTH, H., 2014. CVSS implementation guidance. *National Institute of Standards*

and Technology, NISTIR-7946,

GONZÁLEZ BRITO, H.R., 2018. Configuraciones internas para el fortalecimiento de la seguridad en WordPress. *VIII Congreso Internacional de Tecnologías y Contenidos Multimedia. INFORMÁTICA 2018*. Havana: s.n., pp. 1-9.

GONZÁLEZ BRITO, H.R. y MONTESINO PERURENA, R., 2018. Capacidades de las metodologías de pruebas de penetración para detectar vulnerabilidades frecuentes en aplicaciones web. *Revista Cubana de Ciencias Informáticas* [en línea], vol. 12, no. 4, pp. 52-65. Disponible en: http://rcci.uci.cu/?journal=rcci&page=article&op=view&path%5B%5D=1742.

IRSHAD, M., BRITTO, R. y PETERSEN, K., 2021. Adapting Behavior Driven Development (BDD) for large-scale software systems. *Journal of Systems and Software*, ISSN 01641212. DOI 10.1016/j.jss.2021.110944.

LAVERDIÈRE, M. y MERLO, E., 2018. Detection of protection-impacting changes during software evolution. *2018 IEEE 25th International Conference on Software Analysis, Evolution and Reengineering (SANER)*. S.l.: s.n., pp. 434-444. DOI 10.1109/SANER.2018.8330230.

MENON, P.R., 2016. Behavior-driven development using specification by example: An approach for delivering the right software built in right way. *Emerging Innovations in Agile Software Development*. S.l.: s.n., ISBN 9781466698598.

MOHAMMED, R., 2016. Assessment of Web Scanner Tools. *International Journal of Computer Applications*, vol. 133, no. 5, pp. 1-4. DOI 10.5120/ijca2016907794.

PETRICĂ, G., AXINTE, S., BACIVAROV, I.C., FIROIU, M. y MIHAI, I., 2017. Studying cyber security threats to web platforms using attack tree diagrams. *2017 9th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*. S.l.: s.n., pp. 1-6. DOI 10.1109/ECAI.2017.8166456.

RAHMAN, M. y GAO, J., 2015. A reusable automated acceptance testing architecture for microservices in behavior-driven development. *Proceedings - 9th IEEE International Symposium on Service-Oriented System Engineering, IEEE SOSE 2015*. S.l.: s.n., ISBN 9781479983551. DOI 10.1109/SOSE.2015.55.

RODAS-SILVA, J., GALINDO, J.A., GARCÍA-GUTIÉRREZ, J. y BENAVIDES, D., 2019. Selection of Software Product Line Implementation Components Using Recommender Systems: An Application to Wordpress. *IEEE Access*, vol. 7, pp. 69226-69245. DOI 10.1109/ACCESS.2019.2918469.

SHTEIMAN, B., 2014. Why CMS platforms are breeding security vulnerabilities. *Network Security*, vol. 2014, no. 1, pp. 7-9. ISSN 13534858. DOI 10.1016/S1353-4858(14)70006-6.

W3TECHS, 2019. *Usage of Content Management Systems for Websites* [en línea]. 2019. S.l.: s.n. Disponible en: https://w3techs.com/technologies/overview/content_management/all.

WALDEN, J., DOYLE, M., WELCH, G.A. y WHELAN, M., 2009. Security of open source web applications. *2009 3rd International Symposium on Empirical Software Engineering and Measurement, ESEM 2009*. S.l.: s.n., pp. 545-553. ISBN 9781424448418. DOI 10.1109/ESEM.2009.5314215.

WANG, D., LIU, L., LIN, J., ZHAO, W. y DU, X., 2018. Detecting XSS Vulnerability Based on DOM State Transition. *Beijing Gongye Daxue Xuebao/Journal of Beijing University of Technology*, ISSN 02540037. DOI 10.11936/bjutxb2017060016.

WEIDMAN, G., 2014. *Penetration Testing: A Hands-On Introduction to Hacking*. S.l.: s.n. ISBN 9788578110796.