Aplicaciones web vulnerables a propósito

Fernando Román Muñoz, Iván Israel Sabido Cortes, Luis Javier García Villalba

froman@ucm.es, javiergv@fdi.ucm.es

Grupo de Análisis, Seguridad y Sistemas (GASS)
Departamento de Ingeniería del Software e Inteligencia Artificial (DISIA)
Facultad de Informática, Despacho 431, Universidad Complutense de Madrid (UCM)
Calle Profesor José García Santesmases, 9, Ciudad Universitaria, 28040 Madrid, España

Resumen: *Existen varios tipos de aplicaciones web vulnerables, por ejemplo las aplicaciones desarrolladas para alguna utilidad, sobre las que se han detectado vulnerabilidades, como sistemas de facturación online o sistemas CMS; o las aplicaciones vulnerables a propósito, desarrolladas para realizar pruebas a herramientas de seguridad web o capacitación a desarrolladores o profesionales de la seguridad. Para la evaluación correcta de las herramientas de análisis de vulnerabilidades e impartición de formación en detección de vulnerabilidades web, se necesita de una aplicación o conjunto de aplicaciones web vulnerables consensuadas con las que probar y un conjunto de vulnerabilidades predefinido para las aplicaciones web. En el presente trabajo se hace un análisis y valoración de las aplicaciones vulnerables a propósito existentes, con el objetivo de seleccionar y probar las que más tipos de vulnerabilidades incluyan y que mejor se puedan ampliar con nuevas. Las pruebas para este trabajo se han realizado con 2 escáneres automatizados de vulnerabilidades web.*

Palabras clave: Aplicaciones Web Vulnerables, Escáneres Automatizados de Vulnerabilidades Web, Seguridad Web, Vulnerabilidades.

Abstract: *There are two main kinds of vulnerable web applications, usual applications developed for some aim and applications vulnerable by design. In this paper an analysis and assessment of vulnerable web applications is conducted in order to select the applications that includes the larger set of types of vulnerabilities. Then those applications are improved with more types of web vulnerabilities that vulnerable web applications do not include. Lastly, the new vulnerable web applications have been analyzed to check if web vulnerability scanners are able to detect the new added vulnerabilities, those vulnerabilities that vulnerable by design web applications do not used to include. The results show that the tools are not very successful in detecting those vulnerabilities, less than well-known vulnerabilities.*

Keywords: Vulnerable Web Applications, Automated Vulnerability Scanners Web, Web Security Vulnerabilities.

1 Introducción

La seguridad en aplicaciones web es un aspecto importante para la protección de los activos. Estos activos pueden ser elementos como un servidor, información almacenada en la base de datos o hasta la reputación de la empresa o gobierno. Una aplicación web debe cumplir tres aspectos importantes para su buen funcionamiento: integridad, disponibilidad y confiabilidad. Estos tres aspectos hacen necesaria la utilización de herramientas para la detección de vulnerabilidades en el desarrollo de la aplicación y una mejor capacitación por parte de los programadores, para el desarrollo seguro de éstas.

Las herramientas más populares para la detección de vulnerabilidades en aplicaciones web son los escáneres automáticos de vulnerabilidades web. Existen tanto comerciales como de software libre. Sin embargo estas herramientas, además de contar con muchas fortalezas, también cuentan con muchas limitaciones, debido principalmente a que las tasas de detección de vulnerabilidades puede variar [1]. Para poder evaluar las fortalezas y limitaciones de estas herramientas, se hace uso de aplicaciones web que contienen vulnerabilidades.

En el presente trabajo se hace un análisis de las distintas aplicaciones web vulnerables existentes. Estas aplicaciones han sido tomadas de distintos trabajos de instituciones.

Para obtener un conjunto representativo de todas las aplicaciones existentes se les aplican 5 criterios de selección, con lo que se obtienen 5 aplicaciones. Como siguiente paso, se hace un análisis del conjunto representativo para obtener a las 2 aplicaciones que contienen la mayoría de los tipos de vulnerabilidades de varias de las clasificaciones más conocidas, como la Guía de Pruebas de OWASP [2] o la clasificación de amenazas de WASC [3]. Para finalizar se agregan 8 nuevos tipos de vulnerabilidades a una de las aplicaciones para contar con una mayor cantidad. Esta aplicación servirá como apoyo para comprobar las capacidades reales de detección de las herramientas de análisis y para impartir formación en detección de vulnerabilidades web.

El artículo se estructura en 6 secciones, siendo la primera la presente introducción. En la sección II se explican los conceptos sobre las aplicaciones vulnerables. En la sección III se enumeran los trabajos previos relevantes. En la sección IV se seleccionan las aplicaciones vulnerables que se van a utilizar. En la sección V se muestran los resultados. Por último, se resumen las conclusiones y el trabajo futuro en la sección VI.

2 Fundamentos

2.1. Vulnerabilidades en aplicaciones web

Una vulnerabilidad es una debilidad en los requisitos del sistema de seguridad, el diseño, ejecución o funcionamiento, que se podría accionar accidentalmente o

explotar intencionalmente y que da lugar a una violación a la política de seguridad del sistema [4].

Las vulnerabilidades en las aplicaciones web pueden ser aprovechadas para obtener algún beneficio o comprometer la reputación de la empresa. Estos ataques pueden ser realizados del lado del cliente como por ejemplo "cross site scripting", del lado del servidor como la inyección de código, o del lado de la capa de datos, en este caso el ataque podría ser una inyección SQL.

La mayoría de las vulnerabilidades se deben a los errores cometidos por los programadores al desarrollar una aplicación web, esto puede ser debido al poco conocimiento en seguridad o a la restricción de tiempo para la entrega de la aplicación.

Existen instituciones que se dedican a clasificar los diferentes tipos de vulnerabilidades, ataques y daños que pueden ocasionar, así como a elaborar consejos de seguridad para evitarlas. Entre estas instituciones tenemos el "Open Web Application Security Project" (OWASP) con su top ten de vulnerabilidades más comunes y su manual de pruebas para aplicaciones web y el "Web Application Security Consortium" WASC con su clasificación de amenazas, entre otras.

2.2. Herramientas de detección

Las herramientas de detección automatizada se han hecho populares como apoyo a los profesionales en la búsqueda de vulnerabilidades en aplicaciones web. Estas herramientas analizan de manera automática o semiautomática la aplicación web, realizando ataques maliciosos en busca de vulnerabilidades que puedan ponerla en riesgo.

Para que la herramienta pueda iniciar el análisis, se le debe proporcionar su URL o URLs y eventualmente las credenciales necesarias. Estás herramientas suelen contar con 3 módulos principales [5]:

1. Un rastreador, que se encarga de iniciar el rastreo de la aplicación partiendo de las URLs proporcionadas, recupera las páginas y archivos accesibles de ésta e identifica todas las entradas que pueden encontrarse en formularios, al igual que los parámetros que se envían al servidor de la aplicación.
2. Un módulo de ataque que se encarga de generar valores que pueden explotar algún tipo de vulnerabilidad, estos valores son generados para cada entrada que ha encontrado el rastreador y para cada tipo de vulnerabilidad.
3. Un módulo de análisis, el cual se encarga de analizar las respuestas retornadas por el servidor o la aplicación de los distintos ataques en busca de algún patrón que pudiera dar como válido el ataque realizado, o para retroalimentar a los otros módulos.

A pesar de que estas herramientas son un gran apoyo para los profesionales de la seguridad, aún generan muchos falsos positivos, esto hace necesario el tener que evaluar las herramientas existentes para conocer sus capacidades y limitaciones. Para poder realizar esta evaluación es necesario definir una lista con las vulnerabilidades web que pueden encontrarse en las aplicaciones actuales y desarrollar una aplicación web que cuente con las vulnerabilidades de la lista [6].

2.3. Listado de tipos de vulnerabilidades

Para este trabajo se ha considerado la lista de vulnerabilidades de la clasificación realizada en [6], la cual es una clasificación basada en el mapeo de vulnerabilidades y clasificaciones existentes. En ese trabajo previo, inicialmente se relacionan algunos de los principales conceptos del desarrollo inseguro de software: vulnerabilidades, amenazas, debilidades, riesgos, controles características de auditoría o patrones de ataque. A continuación se describen las clasificaciones de tipos de vulnerabilidades que existen actualmente, tanto en aplicaciones Web como en cualquier tipo de aplicaciones, indicando las principales características de cada una de ellas. Más adelante se analizan las relaciones que entre las clasificaciones de vulnerabilidades han realizado diversas instituciones. A partir de la información obtenida se consigue un único mapeo entre las clasificaciones. Este mapeo se usa a continuación para, después de seleccionar sólo las vulnerabilidades en aplicaciones Web, obtener un listado que contiene todas las vulnerabilidades Web en las clasificaciones actuales. Esta nueva clasificación incluye las vulnerabilidades actuales en las aplicaciones web y se conforma de una lista de 63 vulnerabilidades, la cual la hace una de las más completas y actualizadas. La clasificación obtenida proporciona una visión más completa de las vulnerabilidades web, con la que los investigadores pueden realizar pruebas a las herramientas de seguridad para comprobar sus capacidades y limitaciones, y para los profesionales de la seguridad el poder llevar a cabo una mejor búsqueda de vulnerabilidades y consideraciones para asegurar la aplicación.

2.4. Aplicaciones vulnerables

Existen varios tipos de aplicaciones web vulnerables. Se cuenta con aplicaciones desarrolladas para alguna utilidad, en las cuales se han detectado vulnerabilidades, y aplicaciones desarrolladas con vulnerabilidades a propósito para realizar pruebas o formación con ellas. Estas últimas también pueden dividirse en aplicaciones web desarrolladas por los fabricantes de herramientas de detección, y las desarrolladas en trabajos independientes.

De esta forma se tienen tres fuentes de aplicaciones vulnerables:

1. Aplicaciones desarrolladas con algún propósito, que tienen alguna vulnerabilidad conocida.
2. Aplicaciones desarrolladas por los fabricantes de herramientas de detección de vulnerabilidades web.
3. Aplicaciones desarrolladas para comprobar las características de estas herramientas o para impartir formación.

3 Trabajos relacionados

Se han desarrollado varios trabajos de investigación en donde se intenta evaluar las capacidades y limitaciones de las distintas herramientas de detección. En [7] se hace uso

de versiones de aplicaciones como wordpress, Drupal y phpBB2, en las cuales se han detectado vulnerabilidades. En estas aplicaciones puede darse el caso de haber otros tipos vulnerabilidades que no hayan sido detectadas o dadas a conocer. En otros trabajos como el realizado en la Universidad de Santa Bárbara [5], se desarrolla una aplicación propia llamada Wackopicko, la cual cuenta con diferentes retos para el rastreo y vulnerabilidades para la prueba de herramientas automatizadas, tanto comerciales como de código libre. En dicho trabajo consideran que una aplicación vulnerable debe contar con los siguientes requisitos: (1) debe tener claramente definidas las vulnerabilidades, (2) debe ser fácilmente personalizable para poder agregar nuevas vulnerabilidades y (3) debe representar a las aplicaciones actuales en términos de funcionalidad y tecnología.

En [8] se hace uso de una aplicación con 5 tipos de vulnerabilidades desarrollada en Drupal y al igual que en el trabajo mencionado anteriormente, se hacen algunas consideraciones para realizar una mejor evaluación de las herramientas. En [9], consideran el uso de niveles con diferentes mecanismos de defensa para incrementar la seguridad. Con estos niveles se puede evaluar la complejidad de los ataques generados para evadir los mecanismos de defensa que siguen siendo vulnerables.

Como puede verse en los distintos trabajos realizados, las vulnerabilidades que han sido consideradas suelen ser muy pocas y las más comunes y conocidas, tales como "SQL injection", "cross site scripting" (reflejado, almacenado y DOM), inclusión de archivos, gestión de sesiones, "cross site request forguery", "path transversal", restricción de acceso insuficientes, protección insuficiente en la capa de transporte, inyección de comandos, contraseñas débiles, manipulación de parámetros e inyección javascript, entre otras. Aunque en cada trabajo se menciona que las herramientas no son capaces de detectar muchas de las vulnerabilidades implementadas, estos tampoco consideran el evaluar las aplicaciones con una lista de tipos de vulnerabilidades más extensa o bien definida. Considerando lo anterior y a que en cada trabajo se consideran distintos tipos de vulnerabilidades, se hace necesario contar con una lista definida de tipos de vulnerabilidades y con una o varias aplicaciones que contengan a estas [6]. Para realizar una mejor valoración de las herramientas y contar con aplicaciones que consideren la tecnología actual, realistas y con buena documentación, se han tomado como filtro para la selección de estas, 5 criterios los cuales se presentan en el siguiente capítulo.

4 Selección de aplicaciones

4.1. Aplicaciones Web vulnerables a propósito

Como primer objetivo del trabajo se ha seleccionado un conjunto lo suficientemente representativo de las aplicaciones web vulnerables que existen actualmente. Para esto, se ha hecho una recopilación de proyectos de distintas organizaciones y sitios o trabajos de recopilación de aplicaciones vulnerables conocidos en el ámbito de la seguridad web. Estas aplicaciones tienen como objetivo probar herramientas y apoyar en la enseñanza en seguridad web.

Para seleccionar un conjunto de aplicaciones lo suficientemente representativo, se ha realizado un análisis considerando los siguientes criterios:

- Deben tener claramente definidas las vulnerabilidades.
- Deben ser fácilmente personalizables para poder agregar nuevas vulnerabilidades.
- Deben representar a las aplicaciones actuales en términos de funcionalidad y tecnología.
- Deben contar con varias vulnerabilidades de distinto tipo.
- Deben contar con buena documentación.

Como aplicaciones vulnerables relevantes tenemos las que se indican a continuación:

- *WebGoat*: Cuenta con varios tipos de vulnerabilidades y sigue actualizándose constantemente. Está desarrollada en Java e incluye distintos tipos de tecnologías como JavaScript, XML, etc. Es mantenida como un proyecto de OWASP y se ha desarrollado para poder agregar nuevas vulnerabilidades. Su objetivo es la enseñanza en seguridad y la evaluación de herramientas automatizadas. Cuenta con una buena documentación de las vulnerabilidades que están en la aplicación así como video tutoriales.

- *Mutillidae II*: Cuenta con varios tipos de vulnerabilidades y constantemente se actualiza. Maneja tecnología actual y se le pueden agregar nuevas vulnerabilidades. No se basa en una temática, por ejemplo un blog, sin embargo los ejemplos si pueden ser considerados como pequeñas aplicaciones reales. Cuenta con distintos niveles de seguridad. Está desarrollado en PHP y tiene como base de datos MySql. Su objetivo es la enseñanza en seguridad web. Cuenta también con una buena documentación de las vulnerabilidades que están en la aplicación así como video tutoriales.

- *Damn Vulnerable Web Application*: Cuenta con varios tipos de vulnerabilidades. Maneja tecnología actual y es desarrollada para poder agregar nuevas vulnerabilidades. Su funcionalidad no es como las aplicaciones reales ya que ha sido pensada para la enseñanza de seguridad web. Al igual que Mutillidae cuenta con niveles de seguridad. Cuenta con una buena documentación de las vulnerabilidades que están en la aplicación y está desarrollada en PHP y como base de datos hace uso de MySql.

- *WackoPicko*: Está desarrollada en PHP, su funcionalidad se basa en un portal de ventas de fotos. Cuenta con varios tipos de vulnerabilidades y con tecnología actual. Es difícil agregar nuevas vulnerabilidades sin salirse de su funcionalidad. Cuenta con una buena documentación de las vulnerabilidades que están en la aplicación.

- *The ButterFly Security Project*: Este proyecto tiene como objetivo dar una idea de las vulnerabilidades en aplicaciones Web comunes. Es desarrollado en PHP y MySql y cuenta con varias vulnerabilidades documentadas.

Las aplicaciones que se han elegido cumplen con nuestros propósitos. Algunas de estas aplicaciones han sido desarrolladas como trabajos de investigación, otras por empresas que se dedican al análisis de vulnerabilidades. Las aplicaciones pueden ser fácilmente instaladas en distintas plataformas.

4.2. Otras aplicaciones vulnerables

No se han considerado aplicaciones a las que se le ha encontrado alguna vulnerabilidad como Joomla o Wordpress. En estas aplicaciones no se tiene la certeza si han sido detectadas todas las vulnerabilidades que puedan existir en la aplicación. Aplicaciones que no tienen una buena documentación, que cuentan con pocas vulnerabilidades, que no representen aplicaciones con tecnologías actuales o no hayan sido actualizadas en mucho tiempo tampoco han sido consideradas.

Otras conocidas como Hackme Bank, ha sido desarrollada en .NET y su funcionalidad se basa en un portal bancario. Cuenta con varios tipos de vulnerabilidades y con tecnología actual, así como con una buena documentación de las vulnerabilidades que tiene. Como contrapartida, debido a que no se cuenta con su código fuente, no pueden agregarse nuevas vulnerabilidades. Google Gruyere fue una iniciativa de Google, desarrollada en Python, para dar a conocer y enseñar a los desarrolladores las vulnerabilidades que pueden tener las aplicaciones web. Cuenta con una buena documentación de las vulnerabilidades que están en la aplicación, pero no con suficientes vulnerabilidades para nuestro propósito.

4.3. Aplicaciones web con más vulnerabilidades

En este apartado se han seleccionado las aplicaciones que cuentan con el mayor número de tipos de vulnerabilidades distintas de la clasificación presentada en [6], de entre el conjunto representativo de aplicaciones. Para esto, se han analizado de forma manual y con ayuda de distintas herramientas, cada una de las aplicaciones del conjunto seleccionado en la primera fase. El número de vulnerabilidades que contiene la versión analizada de cada una puede verse en la Tabla 1.

Tabla 1. Cantidad de vulnerabilidades en las aplicaciones seleccionadas.

Aplicación	Vulnerabilidades
WebGoat 5.4	40
Mutillidae 2.6.8	38
Damn Vulnerable Web Application 1.0	17
The ButterFly Security Project 1.0	36
WackoPicko 1.0	11

Las aplicaciones con mayor número de vulnerabilidades de la lista y que han sido seleccionadas para el presente trabajo son WebGoat [10] y Mutillidae [11]. Ambas aplicaciones tienen un total de 46 vulnerabilidades diferentes. Butterfly podría ser tenida en cuenta, pero debido a que ya se tiene una aplicación desarrollada con el lenguaje PHP no ha sido considerada al tener menos vulnerabilidades que Mutillidae.

En la Figura 1, se observa el proceso de selección y los filtros por los que tuvieron que pasar las aplicaciones web vulnerables a propósito. A pesar de que existen muchas aplicaciones desarrolladas por instituciones, organizaciones, universidades y algunas en trabajos de investigación, estas no cuentan con lo necesario para pasar los filtros de la primera fase. En la segunda fase, 3 aplicaciones cumplen con los requerimientos, pero solamente 2 han sido consideradas en este trabajo.

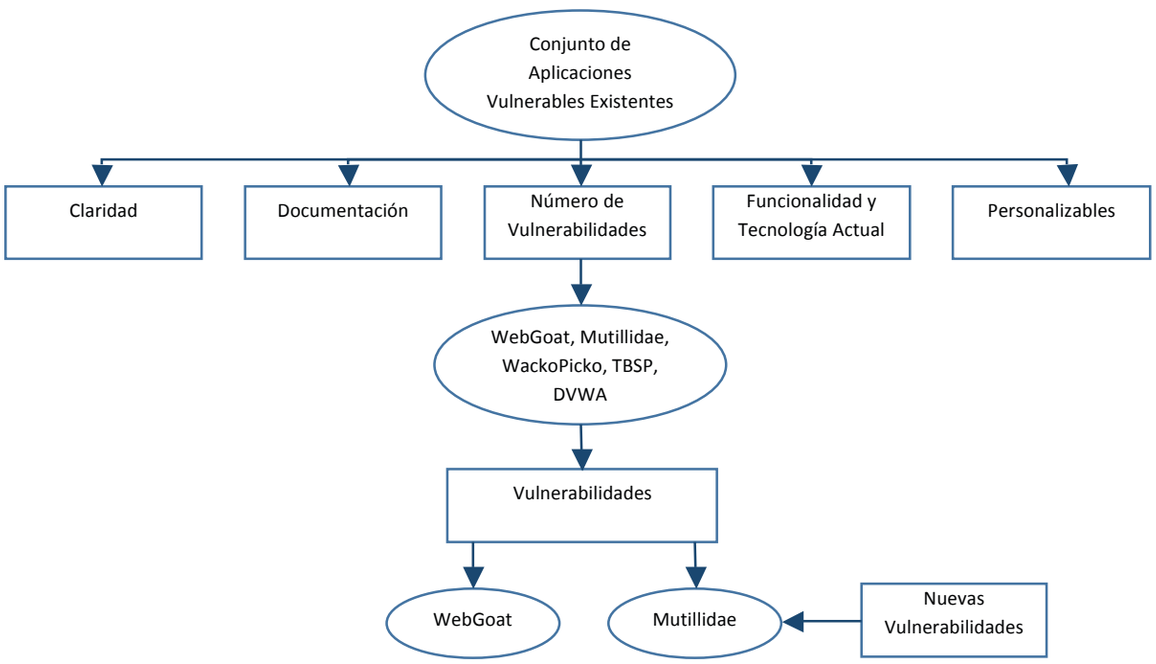

Figura 1: Proceso para la selección de las aplicaciones vulnerables a propósito

4.4. Vulnerabilidades implementadas

Para finalizar el proceso se ha decidido implementar 8 nuevos tipos de vulnerabilidades seleccionadas por su relevancia e información disponible, dejando para una fase posterior la implementación de otras. Se han añadido en Mutillidae debido a que es fácilmente personalizable y a que está desarrollada en el lenguaje más conocido y usado para el desarrollo de aplicaciones web (PHP). Se considera que es un buen modelo de aplicación que puede encontrarse en Internet y con el que se pueden realizar cursos de capacitación y pruebas para las herramientas automatizadas. Las vulnerabilidades añadidas a Mutillidae son las siguientes:

- *Abuso de Funcionalidad, Mecanismo de Recuperación de Contraseñas Inseguro, Proceso de Validación Insuficiente*: Estas vulnerabilidades pueden llevar a grandes pérdidas a las empresas por lo que son importantes detectarlas a tiempo. Se ha implementado como ejemplo de los errores que suelen cometerse en la lógica al desarrollar una aplicación.
- *Null Byte e Inyección SSI*: Aunque ya no son tan comunes, aún siguen habiendo aplicaciones con estas vulnerabilidades, por lo que es necesario implementarlas como ejemplo.
- *Inyección de Código:* Se ha implementado debido a que los programadores validan el manejo de inyecciones SQL pero no toman en cuenta las inyecciones de código de algún lenguaje como javascrip o PHP.
- *Inyección XPath:* Se ha implementado debido a que muchas aplicaciones hacen uso de XPath para navegar a través de elementos y atributos de los documentos XML, pero no se hace una validación correcta por parte de los programadores.
- *Inyección NoSql:* Se ha implementado, debido a que cada día las bases de datos NoSql se implementan más y no se toman las debidas precauciones en las validaciones de los datos de entrada.

En la Figura 2 se muestra una pantalla de ellas en Mutillidae II.

4.5. Vulnerabilidades no implementadas

Uno de los objetivos del presente trabajo es obtener un conjunto de aplicaciones que sean fáciles de utilizar e implementar en un equipo. Por lo que, algunas vulnerabilidades no han sido implementadas o tomadas en cuenta en este trabajo, esto es, debido a que se necesitan de otros componentes, lo que las haría más difícil de configurar.

- *Integer Overflows, Format String*: En lenguajes de alto cuentan con protección lo que hace difícil de realizar ejemplos simples.
- *HTTP Request Smuggling, HTTP Response Smuggling*: No han sido implementadas debido a que se necesitan otros componentes como proxys, por lo que se deja como trabajo a futuro.
- *LDAP, Mail Command Injection*: No han sido implementadas debido a que se necesita la instalación de servidores con estas tecnologías. Se deja como trabajo a futuro.
- *HTTP Response Splitting*: Versiones modernas de Java y Php cuentan con protección ante este ataque.
- *Routing Detour, SOAP Array Abuse, Web services testing*: En el presente trabajo no se hace uso de las tecnologías de servicios web, por lo cual se dejan como trabajo a futuro.
- *XQuery Injection*: No se ha implementado un ejemplo, debido a que en PHP no se cuenta con un parser nativo. Se deja como trabajo a futuro.
- *EL Injection*: Se deja como trabajo a futuro para webgoat.
- *Padding Oracle Attack*: Se deja como trabajo a futuro para Webgoat.

Figura 2: Nuevas vulnerabilidades añadidas a Mutillidae II.

5 Experimentos y resultados

Se ha realizado un análisis de nuestras aplicaciones con Vega 1.0 [12] y Zaproxy 2.3 [13], las cuales son 2 herramientas automatizadas de código abierto para la búsqueda de vulnerabilidades web muy conocidas. Éstas se encuentran preinstaladas en la distribución de Kali Linux 1.0, utilizada en este trabajo. Las aplicaciones vulnerables se han instalado en XAMPP 1.7.3, que se ha atacado con las herramientas para su análisis. Este análisis se hace con el fin de examinar las 2 aplicaciones Web con herramientas para probar sus capacidades de detección de vulnerabilidades.

5.1. Procedimiento

Para poder realizar el análisis con las herramientas se realizó lo siguiente:

1. Se configuraron y activaron los distintos plugins de las herramientas para detectar vulnerabilidades.
2. Se les proporcionó a las herramientas la dirección de la aplicación para realizar un análisis automático.
3. Se activó el proxy de la herramienta y se realizó una navegación manual de la aplicación web.
4. Se verificó que las vulnerabilidades detectadas no sean falsos positivos.

Esto se realizó tanto para Mutillidae como para WebGoat. El paso 2, se tuvo que realizar de dos formas en ambas herramientas, una navegación automática y debido a la dificultad de las herramientas para realizar la navegación automática se procedió a realizar una navegación semiautomática, en la cual se hizo una navegación manual con el proxy de la herramienta capturando las secciones visitadas. Al finalizar la navegación se tomó como base todo lo capturado por el proxy y se realizó de nuevo el análisis automático.

5.2. Resultados y análisis

En la Tabla II, se observa la capacidad de detección de las herramientas Vega y Zaproxy. Vega ha sido la que más vulnerabilidades ha detectado, con 17 vulnerabilidades de las 49 que se encuentran en las dos aplicaciones. En Mutillidae detecto 17 vulnerabilidades y en WebGoat 13. También puede observarse que Vega solo fue capaz de detectar una de las vulnerabilidades que fueron agregadas a Mutillidae. Esta vulnerabilidade es inyección de código. Ambas herramientas detectaron 14 vulnerabilidades del mismo tipo. Mientras que Zaproxy fue capaz de detectar la vulnerabilidad conocida como Falsificación de Petición en Sitios Cruzados (CSRF) que Vega no detectó, siendo esta vulnerabilidad una de las más comunes actualmente. Por otro lado Vega fue capaz de detectar Inyección de Código y Enumeración de Usuarios, al contrario que Zaproxy.

Las vulnerabilidades detectadas por ambas herramientas se basan en la lista conocida como OWASP Top Ten. Esto puede darse a que los desarrolladores de ambas, basan sus scripts o plugins en dicha lista, por ser la más conocida y difundida entre los profesionales de seguridad web. En la Tabla II también se observa que el porcentaje de detección es bajo, esto es debido a la dificultad de las herramientas para realizar el rastreo y a los pocos plugins implementados para la detección de distintas vulnerabilidades, sobre todo en tecnologías como NoSql, XML y Json. Durante las dos fases de análisis realizado con las herramientas, se observó que la principal limitación de estas es en la navegación dentro de la aplicación. Al realizar la navegación semiautomática se pudieron detectar más vulnerabilidades.

Tabla 2: Capacidades de Detección Vega y Zaproxy.

	Vega	Zaproxy
49 Vulnerabilidades (WebGoat y Mutillidae).	17	15
Porcentaje de Detección	35.69%	30.61%
Mutillidae	17	15
WebGoat	13	13
Vulnerabilidades Nuevas (Mutillidae)	1	0

Esta sección deberá contener en primer lugar todas las condiciones en las que se realizaron los experimentos además de una descripción detallada de cómo fueron realizados y en qué consisten, además de una descripción de la base de datos utilizada si es que se utilizó alguna. Para poder comprender mejor éstos resultados es una buena idea utilizar tablas y figuras. Sin embargo si éstas se utilizan es muy importante describirlas detalladamente. Cada experimento sólo debe ser descrito, evitar discutir los experimentos ya que las discusiones deben colocarse en la sección de Discusión de los Experimentos.

6 Conclusiones y trabajo futuro

En el presente trabajo se ha realizado la selección de dos aplicaciones (WebGoat y Mutilliddae), ambas desarrolladas bajo la supervisión de OWASP. Estas aplicaciones tienen como objetivo, comprobar las capacidades de las herramientas de análisis de vulnerabilidades web y como apoyo a una mejor formación para los desarrolladores. Para la elección de las aplicaciones se ha realizado un proceso de selección basándose en las aplicaciones vulnerables existentes y más conocidas en el ámbito de seguridad web.

Las aplicaciones consideradas tienen claramente definidas las vulnerabilidades, son fáciles de personalizar y agregar nuevos tipos de vulnerabilidades, cuentan con tecnología y funcionalidad de aplicaciones web actuales, al igual que cuentan con una buena documentación. Después de haber agregado 8 nuevas vulnerabilidades a Mutillidae, entre ambas aplicaciones cuentan se cuenta con 49 vulnerabilidades. Para probar las aplicaciones y realizar una valoración de dos herramientas automatizadas que vienen en la distribución Linux Kali, conocidas como Vega y Zaproxy, se realizó un análisis a ambas aplicaciones con dichas herramientas, esto con el objetivo de comprobar sus capacidades y limitaciones.

Como puede observarse en los resultados, las herramientas sólo fueron capaces de detectar un porcentaje muy bajo del total. Vega fue el que más vulnerabilidades detectó con 17 y Zaproxy detectó 15. Solamente Vega fue capaz de detectar 1 de las 8 vulnerabilidades agregadas a Mutillidae. Las vulnerabilidades que detectaron ambas son las consideradas en el top ten de OWASP, esto puede ser debido a que los desarrolladores de las herramientas basan sus scripts o plugins en esta clasificación. Otra de las limitaciones de las herramientas es el rastreador, debido a que se les complicó el poder analizar o encontrar todos los enlaces. Esta limitación se pudo ver aún más en la aplicación WebGoat.

Como puede apreciarse, las aplicaciones web vulnerables pueden ayudar a determinar las capacidades reales de detección de las herramientas de análisis de vulnerabilidades web y también ayudar a impartir una formación más completa a desarrolladores en detección de vulnerabilidades web.

Nuestros planes para trabajos futuros son desarrollar las vulnerabilidades que faltan, analizar las capacidades de más herramientas existentes en el mercado e impartir cursos de formación haciendo uso de éstas dos

aplicaciones. También se pretende realizar una aplicación web con una temática como por ejemplo, una aplicación escolar, que cuente con todas las vulnerabilidades de la lista y cumpla con los 5 criterios mencionados, así poder realizar una mejor valoración de las herramientas y contribuir con una herramienta más completa para la enseñanza o capacitación.

Agradecimientos

Los autores agradecen la financiación que les brinda el Programa Marco de Investigación e Innovación Horizonte 2020 de la Comisión Europea a través del Proyecto H2020-FCT-2015/700326-RAMSES (Internet Forensic Platform for Tracking the Money Flow of Financially-Motivated Malware).

Referencias bibliográficas

[1] Yuliana Martirosyan: “Security Evaluation of Web Application Vulnerability Scanners Strengths and Limitations Using Custom Web Application”, PhD Thesis, California State University - East Bay, Octubre de 2012.

[2] OWASP Open Web Application Security Project, “OWASP Testing Guide”, 2013, https://www.owasp.org/index.php/OWASP_Testing\ _project

[3] WASC Web Application Security Consortium, “The WASC Threat Classification”, 2010, http://projects.webappsec.org/w/page/13246978/ThreatClassification

[4] National Institute of Standards and Technology (NIST), “Engineering Principles for Information Technology Security (A Baseline for Achieving Security)”, NIST SP 800-27, Revision A, June 2004, http://csrc.nist.gov/publications/nistpubs/

[5] Adam Doupé, Marco Cova, and Giovanni Vigna, “Why Johnny Can’t Pentest: An Analysis of Black-box Web Vulnerability Scanners”, Proceedings of the 7th International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA'10), pp. 111-131, 2010.

[6] Fernando Román Muñoz, Iván Israel Sabido Cortes, Luis Javier García Villalba, “Capacidades de detección de las herramientas de análisis de vulnerabilidades en aplicaciones Web”, Actas del XIII Reunión Española sobre Criptología y Seguridad de la Información (RECSI 2014), Septiembre 2014.

[7] J. Bau, E. Bursztein, D.Gupta, J. Mitchell, “State of the Art: Automated Black-Box Web Application Vulnerability Testing”, Proceedings of the 2010 IEEE Symposium on Security and Privacy, Berkeley, USA, Mayo 16-19, pp. 332-345, 2010.

[8] Alexandre Miguel Ferreira y Harald Kleppe, “Effectiveness of Automated Application Penetration Testing Tools”, 2011.

[9] Fong, E.; Gaucher, R.; Okun, V.; Black, P.E., “Building a Test Suite for Web Application Scanners”, Proceedings of the Hawaii International Conference on System Sciences, Proceedings of the 41st Annual, pp.478-478, 2008.

[10] WebGoat, https://www.owasp.org/index.php/Proyecto _WebGoat _OWASP

[11] Mutillidae, https://www.owasp.org/index.php/OWA SP _Mutillidae_2_ Project

[12] Vega Vulnerability Scanner, https://subgraph.com/vega/

[13] Zaproxy, https://www.owasp.org/index.php/OWASP _Zed_Attack_ Proxy_Project