Validación de un Modelo para Incorporar la Seguridad durante el Ciclo de Desarrollo de Software

Lucero, Emilce Beatriz (1) **– Castillo, Julio Javier** (2) **– Rearte, Emilio P.**(3) **– Carrizo, Pablo J.**(4)

(1) Informática II, Departamento de Ingeniería Electrónica - UTN-Facultad Regional La Rioja
(2) Laboratorio de Investigación de Software LIS-DISI - UTN-Facultad Regional Córdoba
(3) Departamento DACyTAPAU – Universidad Nacional de La Rioja
(4) Laboratorio de Electrónica - UTN-Facultad Regional La Rioja
bealucero@yahoo.com.ar

Resumen

El presente artículo pretende dar cuenta de la validación de un modelo metodológico que se ha propuesto para el desarrollo de un producto de software, en el que la seguridad se incorpora, desde etapas tempranas y como una propiedad relevante, al ciclo de su desarrollo. Concretamente, aquí se presentan los pasos establecidos como estrategia para la obtención de un software seguro, empleando como objeto destinatario de este proceso, el aplicativo web de gestión administrativa/académica del Departamento de Ingeniería Electrónica de la Universidad Tecnológica Nacional, Facultad Regional La Rioja. Asimismo, se proporcionan algunos elementos conceptuales, procedimentales y evaluativos que permiten analizar el estado del producto resultante y, consecuentemente, determinar el nivel de seguridad alcanzado y la efectividad del modelo. Finalmente, se ofrece el análisis de los resultados obtenidos y las conclusiones acerca del desarrollo de los lineamientos de gestión técnica del modelo propuesto en el proceso de validación abordado.

Palabras Clave: Software Seguro, Validación de Seguridad

1. INTRODUCCIÓN

Las malas prácticas de programación llevan a tener sistemas con un alto grado de vulnerabilidad. Una vulnerabilidad es un defecto de seguridad en el software que puede ser usado maliciosamente por un atacante para ganar acceso al sistema o a la red (Awang, 2013).

Un software seguro es aquel que fue diseñado, implementado y configurado para seguir funcionando correctamente ante la presencia de la mayoría de los ataques, fallas o debilidades conocidas en el software, tolerando los errores y fracasos que resulten de tales ataques, fallas o debilidades (Correa, 2021).

El Proyecto Abierto de Seguridad de Aplicaciones Web (OWASP, por sus siglas en inglés), en el año 2013, publicó las diez vulnerabilidades de seguridad más comunes en aplicaciones en aplicaciones Web. Estas vulnerabilidades son: inyección de código maligno, administración de la sesión, vulneración de la autenticación, cross-site scripting, referencias directas a objetos de forma insegura, exposición de datos sensibles, control de acceso, cross-site request forgery, utilización de componentes con problemas conocidos, y redireccionamiento y reenvíos no validados (Goertzel, 2006).

Para asegurar la calidad del producto informático en cuanto a seguridad se refiere, se requiere de contar con un informe de medición del estado de seguridad del aplicativo Web para adoptar medidas estratégicas que puedan mitigar las vulnerabilidades detectadas.

Existen diferentes técnicas utilizadas para comprobar el estado de seguridad de un aplicativo Web, pues no se puede garantizar la seguridad de una aplicación Web sin hacer las respectivas pruebas de seguridad en ella.
En este trabajo se presenta un proceso para verificar la seguridad de una aplicación web en la que se implementó una metodología de ciclo de vida que mejoran la seguridad del software (SDLC). La metodología de verificación se basa en los lineamientos propuestos el Estándar para verificación de la seguridad de aplicaciones de OWASP (ASVS, por sus siglas en inglés) (OWASP, 2021), se documenta una serie de pruebas de penetración que se hicieron con las herramientas de análisis de vulnerabilidades OWASP Zed Attack Proxy Project Zap[1] y Burp Suite[2], para finalmente, con los resultados arrojados por estas herramientas, evidenciar el nivel de riesgo o estado de seguridad del aplicativo web y determinar el nivel de seguridad alcanzado en OWASP ASVS.

2. MARCO TEÓRICO

Las pruebas de seguridad se podrían definir como un conjunto de actividades que se llevan a cabo para definir fallas y vulnerabilidades en aplicaciones web, buscando disminuir el impacto de ataques a ellas y perdida de información importante (OWASP, 2020).

2.1. OWASP ASVS (Application Security Verification Standard)

Estándar de Verificación de Seguridad en Aplicaciones (ASVS) de OWASP, documenta y organiza una lista de verificación de requerimientos de seguridad que se utiliza para diseñar, desarrollar, probar y definir que tan segura es una aplicación web. El ASVS de OWASP define 3 niveles de seguridad que una aplicación puede lograr (OWASP, 2020).

Nivel 1: Las aplicaciones con este nivel se defienden contra vulnerabilidades que son fáciles de descubrir, e incluido el Top 10 de OWASP (OWASP, 2017) y otras listas de comprobación similares. Los controles de nivel 1 se pueden comprobar automáticamente mediante herramientas o simplemente manualmente sin acceso al código fuente. Las amenazas a la aplicación probablemente serán de atacantes que utilizan técnicas simples y de bajo esfuerzo para identificar vulnerabilidades fáciles de encontrar y fáciles de explotar.
Nivel 2: Una aplicación alcanza ASVS Nivel 2 (o Estándar) si se defiende adecuadamente contra la mayoría de los riesgos asociados con el software hoy en día. El nivel 2 garantiza que los controles de seguridad estén en su lugar, sean eficaces y se utilicen dentro de la aplicación. Las amenazas a las aplicaciones de nivel 2 suelen ser atacantes calificados y motivados que se centran en objetivos específicos utilizando herramientas y técnicas, que son altamente practicadas y eficaces para descubrir y explotar las debilidades dentro de las aplicaciones. Nivel recomendado para la mayoría de aplicaciones. Tiene defensas contra la mayoría de riesgos actuales.
Nivel 3: Es el nivel más alto de verificación dentro del ASVS. Este nivel se reserva normalmente para aplicaciones que requieren niveles significativos de verificación de seguridad, como los que se pueden encontrar dentro de áreas de militar, salud y

[1] Es un proyecto desarrollado por la comunidad de OWASP. Su página oficial es https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project.
[2] Herramienta para auditar la seguridad de páginas Web. Su página oficial es https://portswigger.net/burp.

seguridad, infraestructura crítica, etc. Recomendado para aplicaciones de misión crítica, como aplicaciones bancarias, médicas, militares, entre otras.

Cada nivel ASVS contiene una lista de requisitos de seguridad, que permite evaluar objetivamente los niveles. Cada uno de estos requisitos también se puede asignar a características y capacidades específicas de seguridad que los desarrolladores deben integrar en el software.

2.2. Guía de pruebas OWASP

La Guía de Pruebas OWASP es un marco referencial que incluye un conjunto de puntos de control, basados en la colección de buenas prácticas de seguridad utilizadas en aplicaciones web (OWASP, 2014). El enfoque de OWASP, además de ser orientada a las pruebas de seguridad en todo el ciclo de desarrollo, también se centra en las pruebas de intrusión o caja negra; que es la manera como la mayoría de atacantes puede tener acceso a la aplicación. El conjunto de pruebas se divide en categorías:

1. Recopilación de información (OTG-INFO)
2. Pruebas de gestión de configuración y despliegue (OTG-CONFIG)
3. Pruebas de gestión de identidad (OTG-IDENT)
4. Pruebas de autenticación (OTG-AUTHN)
5. Pruebas de autorización (OTG-AUTHZ)
6. Pruebas de gestión de sesiones (OTG-SESS)
7. Pruebas de validación de entradas (OTG-INPVAL)
8. Pruebas de manejo de errores (OTG-ERR)
9. Pruebas de criptografía débil (OTG-CRYPST)
10. Pruebas de lógica de negocio (OTG-BUSLOGIC)
11. Pruebas del lado del cliente (OTG-CLIENT)

Esta metodología establece las partes de una aplicación web que se deben probar sucesivamente utilizando diferentes tipos de herramientas.

2.3. Técnicas de pruebas de seguridad

La guía para pruebas de OWASP (OWASP, 2014) define cuatro principales técnicas para la realización de pruebas de seguridad en las aplicaciones web:

Inspecciones manuales: Consiste principalmente en hacer entrevistas, revisar manuales, políticas, documentos de requerimientos y diseño, entre otros.

Modelamiento de amenazas: Permite encontrar posibles amenazas e identificar vulnerabilidades; enfocándose en la gestión de riesgos y en la descomposición de requerimientos de la aplicación (conocer sus funcionalidades, activos, lógica de negocio, objetivos, etc.) para crear estrategias de mitigación.

Revisión de código (de caja blanca): Pruebas de caja blanca también, brindando una ventaja no solamente en la seguridad de las aplicaciones sino también en la calidad del software y en la implementación de buenas prácticas de desarrollo. Aspectos como debilidades en algoritmos criptográficos, problemas de concurrencia, lógica de negocio, problemas de control de acceso, tipos de datos, etc.

Pruebas de penetración: También conocidas como hacking ético, las pruebas de penetración (o **intrusión**) consisten en encontrar vulnerabilidades en la aplicación sin conocer su funcionamiento interno. Se requiere que el encargado de las pruebas tenga conocimientos sobre seguridad informática y habilidades para actuar o pensar como un atacante para encontrar las vulnerabilidades.

2.4. Estimación del Riesgo

Para determinar el riesgo global que una aplicación Web insegura puede causar en una organización, es necesario valorar la probabilidad que se asocia a cada agente de amenaza, el vector de ataque y las debilidades en la seguridad, combinándolas con una estimación del impacto a nivel técnico y de negocios y con esto evaluar finalmente si el daño causado puede no tener consecuencias, o si por el contrario, puede colocar a la organización por fuera del negocio, es decir, determinar si la aplicación Web es vulnerable, junto con las medidas de seguridad que debería tener en cuenta para prevenir un posible ataque.

3. METODOLOGÍA

La metodología tiene como referencia el estándar OWASP ASVS. Se busca verificar si la aplicación se encuentra dentro del nivel 2 o Estándar, que es el apropiado para aplicaciones que usan transacciones Business-to-Business, con riesgo de seguridad moderado y cuyas amenazas son del tipo oportunista y del tipo atacante decidido.
Como primera medida, se revisa que los controles de seguridad establecidos durante el ciclo de desarrollo del software, existan, se aplican y sean eficaces, utilizando para ello listas de verificación. Como estrategia para evidenciar las vulnerabilidades en el sistema de información Web, se optó por el enfoque metodológico basado en pruebas de seguridad, siguiendo algunos de los lineamientos propuestos en la guía de pruebas de seguridad de OWASP. Para llevar a cabo estas pruebas fue necesario establecer una posición de usuario y agente malicioso, donde se asume un desconocimiento de la estructura interna del sistema, así como del código de la aplicación web. Usando pruebas de Black Box y Grey Box, usando las herramientas de evaluación y penetración de vulnerabilidades: ZAP para un análisis de escáner activo de la aplicación y Burp Suite para trabajar pruebas pasivas de auditoría. Se documentan las pruebas realizadas a la aplicación web para crear una evaluación de los riesgos en base a las vulnerabilidades encontradas, basándose en la metodología de evaluación de riesgos OWASP Top 10 – 2017 (OWASP, 2017) y OWASP Risk Rating Methodology (OWASP, 2020). Finalmente, se determina si la aplicación cumple con los requisitos del estándar de OWASP ASVS.

4. DESARROLLO E IMPLEMENTACIÓN

4.1. Descripción del Aplicativo Web

El objetivo es validar la seguridad del aplicativo web del departamento de Ingeniería Electrónica, que permite gestionar y visualizar el almacenamiento de información que genera el departamento (disposiciones, planificaciones de docentes, entre otros), por medio de una interfaz que variará según el tipo de privilegio asociado al usuario. Este aplicativo se constituye como un activo de información valiosa para la institución y es al que se le ejecutaran las herramientas para el escaneo automático de vulnerabilidades.
En su construcción se utilizó una metodología de desarrollo seguro de software, cuyas actividades de seguridad se incorporan en las distintas fases del ciclo de vida del desarrollo de software, tal como se presenta en la figura 1.

La aplicación web está basada en una típica arquitectura de tres capas. Las diferentes tecnologías y versiones utilizadas para la creación del aplicativo son las siguientes: Sistema Operativo: Linux, Software servidor web: Apache, Lenguaje de

programación: PHP, Framework de desarrollo Laravel, Sistema de Administración de Base de Datos: MySQL.

Figura 1. Fases y Actividades

Durante la fase de Diseño del aplicativo web, se realizaron listas de verificación de codificación seguras, tutoría y capacitación, codificación y pruebas, creación, implementación, configuración y operaciones, y terminando con pruebas independientes de seguimiento para asegurar que todos los controles de seguridad están presentes y funcionales.

Para la validación del software, se preparó un entorno de prueba utilizando el administrador de servicios para XAMPP que contiene un paquete de servicios y aplicaciones que se utiliza para ejecutar aplicaciones web PHP, luego, se instaló la aplicación seleccionada en un entorno local como parte de una evaluación objetiva.

4.2. Verificación de requisitos de seguridad

Se verificó que la lista de chequeo de requisitos de seguridad, se haya aplicado correctamente durante el desarrollo: accesos, proceso de inicio de sesión y autenticación. Manejo de Excepciones. Usuarios, perfil de usuario, verificación de privilegio mínimo. Protección de Datos Personales. Autenticación segura y secreta, que todos los controles de autenticación se realicen del lado del servidor. Bloqueo de usuarios, después de varios intentos erróneos para evitar ataques de fuerza bruta. Desarrollo Seguro, se comprueba que se aplicó patrones y recomendaciones de programación para incrementar la seguridad de datos. Sesiones Seguras, se verificar que toda autenticación exitosa y re-autenticaciones generen un nuevo identificador de sesión, cierre de sesión después de un periodo de inactividad. Validación de datos de entrada en formularios, que los controles estén del lado del servidor. Transacciones seguras, parametrización de consultas, y que no sean susceptibles a la inyección de SQL, desinfección de parámetros del lado del servidor. Asegurar HTML, para que la aplicación no sea susceptible a ataques Cross-Site Scripting (XSS). No Cache, se verificar que todos los formularios que contengan información sensible se les haya desactivado el almacenamiento de caché en el cliente; entre otros.

4.2.1. Ejecución de las Pruebas

Siguiendo la guía de Pruebas de OWASP y con herramientas automatizadas se realizan las distintas pruebas. Se inician las pruebas haciendo una navegación manual de la aplicación web, de forma pasiva con Burp Suite y activa con ZAP, este tipo de exploración ayuda a tener un panorama de cómo está respondiendo el sistema, muestra las peticiones que trabaja, los tipos de entrada que recibe y las tecnologías que usa. Burp Suite muestra una visión detallada en forma de árbol con las carpetas y ficheros a los que se han accediendo con el proxy, construyendo así el site map de la web. Con ZAP se realiza una exploración activa con el escáner, se identifican distintos métodos GET que corresponden a vistas, así como llamadas de archivos que corresponden al Framework. Se observan también, dos métodos POST que corresponden al Login y al Registro, junto con las variables que se envían en cada uno de sus respectivos formularios.

4.2.2. Resultado de las Herramientas de Pruebas

Tras finalizar estas pruebas pasivas de Burp y activas de ZAP, tenemos un panorama general de los riesgos encontrados en la aplicación web. Es importante tener presente que el nivel de riesgo (Alto[3], Medio[4], Bajo[5]) en el que se clasifica cada vulnerabilidad, determina el grado de amenaza o exposición ante fallos de seguridad que puede tener el aplicativo ante inminentes ataques y por consiguiente define su estado de seguridad.

Tabla 1. Vulnerabilidades detectadas por la Herramienta OWASP ZAP

Nivel de Riesgo	Vulnerabilidad Detectada	No. de Instancias
Medio	X-Frame-Options Header Not Set	7
	Private IP Disclosure	5
Bajo	Cookie No HttpOnly Flag	13
	Password Autocomplete in Browser	
	Web Browser XSS Protection Not Enabled	2
	X-Content-Type-Options Header Missing	19

Las alertas lanzadas por la exploración activa de ZAP, muestran una carencia en la configuración del servidor y de las herramientas del navegador. Estas pueden ser resueltas con actualizaciones y ajustes en la configuración por defecto del servidor. Pero en específico la ausencia de tres cabeceras de seguridad HTTP (X-Content-Type-Options, X-Frame-Options y X-XSS-Protection), estas cabeceras pueden ser añadidas dentro de las funciones de Laravel Framework para solucionarlas.Durante las pruebas pasivas con Burp Suite encontramos varias fallas de diseño en la aplicación web listadas en la tabla 2.

[3] Alto: Puede poner en peligro la disponibilidad, confidencialidad o integridad de los datos de los usuarios, o la disponibilidad de los recursos de procesamiento

[4] Medio: Su impacto puede reducirse en gran medida ya sea mediante configuraciones predeterminadas o por la dificultad propia en su explotabilidad.

[5] Baja: Es una vulnerabilidad difícil de explotar y con impacto mínimo.

Tabla 2. Vulnerabilidades detectadas por la Herramienta Burp Suite

Vulnerabilidad Detectada
Ausencia de certificados HTTPS
Ausencia de las cabeceras de seguridad
Ausencia de una configuración adecuada del servidor

4.2.3. Análisis y Resultado de las Pruebas

Una vez concluidas las pruebas realizadas a la aplicación web del Departamento de Electrónica mediante la metodología de pruebas de OWASP, se observa que las herramientas detectaron algunos agujeros en la seguridad del aplicativo creado para el proyecto, identificando cada uno de ellos, no obstante no se detectaron vulnerabilidades para ataques de SQL inyección a razón que el framework utilizado tiene funciones pre-establecidas evitando esta técnica, pero se encontraron otras vulnerabilidades como son HTTP (X-Content-Type-Options, X-Frame-Options y X-XSS-Protection).

Luego de este análisis **se asigna un valor de riesgo mínimo a la aplicación web**, la mayor parte de las vulnerabilidades encontradas corresponden al diseño y la falta de algunas herramientas de seguridad dentro de la aplicación web, como por ejemplo validaciones tipo captcha para distinguir usuarios físicos de bots, certificados en capas de transporte, limites en tiempos de sesión, implementación de recomendaciones para la creación de contraseñas seguras pero pueden ser cubiertas implementado las recomendaciones propuestas para la siguiente auditoria de pruebas.

5. CONCLUSIONES

En este trabajo se llevó a cabo la ejecución de una metodología de evaluación de la seguridad en una aplicación en la que incorporó la seguridad desde etapas tempranas del ciclo de desarrollo del software.

Los estándares de pruebas y las prácticas recomendadas permitieron garantizar que las pruebas se centren en detectar posibles vulnerabilidades de seguridad, en vez de centrarse únicamente en el funcionamiento correcto de las características y las funciones del software. Las revisiones del código complementan las herramientas automatizadas y las pruebas, constituyen un paso fundamental para eliminar las vulnerabilidades de seguridad del software durante el proceso de desarrollo.

Con la ejecución de herramientas automatizadas, se pudo determinar que existen ciertas vulnerabilidades presentes en la aplicación web, pero representan un nivel de riesgo mínimo. Como la aplicación ha seguido durante su desarrollo una metodología de software seguro, podemos afirmar que los controles de seguridad requeridos para alcanzar el nivel 2 del OWASP ASVS están presentes, pero no todos han sido probados y validados, por lo que se hace necesario incorporar a este análisis más pruebas e implementar otras herramientas de seguridad automatizadas.

Se puede concluir que implementar una metodología de desarrollo de software seguro, adoptar herramientas de desarrollo, aplicar estándares de codificación y de pruebas no solo ayuda a implementar la seguridad en el software, sino que al considerar las posibles amenazas que pueden afectar a este tipo de aplicaciones y

realizar los pasos necesarios para eliminar los errores de seguridad desde el principio reducen considerablemente la probabilidad de que las vulnerabilidades de seguridad lleguen a la versión final del software.

Mediante la integración de las actividades de seguridad de software en el SDLC, se puede producir software y aplicaciones con un menor número de vulnerabilidades con riesgos más bajos y sin costos extras para los clientes.

6. REFERENCIAS

Awang, N., and Manaf, A. Detecting vulnerabilities in web applications using automated black box and manual penetration testing. In Advances in Security of Information and Communication Networks, Springer Berlin Heidelberg (2013), 230–239.

Correa Roddy A., Bermejo Higuera J. “Hybrid SecurityAssessmentMethodology forWebApplications” CMES, 2021, vol.126, no.1 https://www.academia.edu/59422750/Hybrid_Security_AssessmentMethodology_for Web_Applications

Goertzel K., Winograd T., McKinley H., P. H. and Hamilton B. (2006) “Security in the software lifecycle. Making Software Development Processes -and Software Produced by Them-More Secure”. Draft V1.2. Department of Homeland Security. USA. https://resources.sei.cmu.edu/asset_files/WhitePaper/2006_019_001_52113.pdf

OWASP Fundation, OWASP Top 10 2017, https://owasp.org/www-project-top-ten/

OWASP Risk Rating Methodology. Recuperado 3 de diciembre de 2020, de https://owasp.org/wwwcommunity/OWASP_Risk_Rating_Methodology

OWASP testing guide v4, 2014 https://owasp.org/www-project-web-security-testing-guide/assets/archive/OWASP_Testing_Guide_v4.pdf

OWASP, Application Security Verification Standard 4.0.3 OCtubre 2021, Url: http://www.owasp.org/index.php/Category:OWASP_Application_Security_Verifi cation_Standard_Project