UN MODELO PARTICULARIZADO PARA DESARROLLAR SOFTWARE SEGURO

Sancho Núñez, José Carlos

Los informes de los últimos años recogen un incremento en el éxito de los ciberataques (National Cryptologic Center Computer Emergency Team Response (CCN-CERT) 2018, 2019). Así, la falta de seguridad en el software ha tomado un papel relevante para las empresas que lo desarrollan. Por este motivo, en el año 2015 se firma la Cátedra Viewnext-UEx, sobre "Seguridad y Auditoría de Sistemas Software" entre la Universidad de Extremadura (UEx) y la empresa de Servicios de Tecnología de la Información del grupo IBM España, Viewnext SA. Bajo este acuerdo que garantiza la transferencia directa de conocimiento se presenta esta investigación.

Tradicionalmente, los controles de seguridad realizados para detectar las vulnerabilidades en el software llegan en la fase de pruebas, una de las últimas de un proyecto de software. Identificar fallos de seguridad en las fases finales no permite conocer su origen y conlleva una alta probabilidad de tener que realizar rediseños en los proyectos. Además, la posibilidad de que vulnerabilidades existentes en el software provoquen ciberataques con éxito ha llevado a las empresas especializadas a plantearse un cambio en el paradigma del desarrollo. Las razones para pensar en un cambio que mejore la seguridad en el software de forma preventiva tienen que ver con las consecuencias devastadoras que implican la pérdida de clientes, reputación o confianza empresarial, los retrasos temporales en la entrega de los proyectos o posibles denuncias por incumplir el Reglamento General de Protección de Datos (RGPD).

Esta investigación buscar mejorar la seguridad en el desarrollo del software y, por ello, persigue los siguientes objetivos: i) identificar otros modelos especializados que apliquen la seguridad de forma preventiva; ii) proponer un nuevo modelo de desarrollo seguro particularizado a partir de los anteriores; iii) definir los indicadores para diseñar un experimento que permita medir las diferencias al desarrollar software utilizando la metodología tradicional (teniendo en cuenta la seguridad únicamente en la fase de pruebas) y el modelo propuesto.

1. Modelos de Desarrollo de Software Seguro

El proceso seguido para el desarrollo del software se denomina ciclo de vida del desarrollo de software. De forma tradicional este proceso se ha segmentado en fases como la toma de requisitos, el análisis, el diseño, la implementación, las pruebas, el despliegue y el mantenimiento. Para establecer este proceso existen varios modelos cuyos enfoques son diferentes en función de las actividades y el momento en el que tienen lugar durante el proceso. De manera general, todos los modelos buscan mantener un control planificado y programado del proceso y, con ello, ajustar los tiempos y reducir el riesgo de exceso de gastos. Destacan varios procesos de desarrollo que se ajustan a distintos paradigmas como son: i) tradicionales, pensados para dividir el proyecto en fases de manera estructurada; ii) orientado a objetos, cuyo enfoque radica principalmente en la reutilización de software; y iii) desarrollos ágiles, donde el objetivo es entregar el software lo más pronto posible.

Sin embargo, los paradigmas anteriores están muy centrados en el proceso de planificación o la entrega del software al cliente y no dedican la suficiente atención a la seguridad de las aplicaciones. Un ciclo de vida de desarrollo de software es considerado seguro (*Secure Software Development Life Cycle*) cuando introduce actividades y controles específicos de seguridad. Esta investigación toma como origen una serie de marcos de trabajos especializados en el desarrollo de software seguro para analizarlos en profundidad. Se estudian los modelos más populares y que son de uso industrial: *Microsoft Security Development Lifecycle* (Microsoft SDL) (Lipner y Howard, 2004), *Agile Security Development Lifecycle en su versión ágil* (Microsoft Corporation, 2010), *Oracle Software Security Assurance* (OSSA) (Redwood Shores, 2011), *Comprehensive Lightweight Application Security Process* (CLASP) (OWASP Project, n.d.) de la organización *Open Web Application Security Project* (OWASP), *Team Software Process Secure* (TSP-Secure) (Davis, Miller, Nichols y Seacord, 2009) *Software Assurance Maturity Model* (OpenSAMM) (OWASP Project, 2009), *Building Security In Maturity Model Framework* (BSIMM) (McGraw, Chess y Miques, 2011).

2. Modelo de Desarrollo de Software Seguro Viewnext-UEx

El análisis en detalle de los modelos anteriores permite tomar de ellos las actividades más importantes que, por su recurrencia, los autores consideran deben existir en cualquier modelo seguro. Se detectan en ellos algunas carencias que son complementadas con nuevas actividades. Con todo ello, se presenta el Modelo

de Desarrollo de Software Seguro Viewnext-UEx, un modelo preventivo que incorpora novedosas y sistemáticas prácticas de seguridad planificadas en el ciclo de vida de desarrollo y auditoría del software.

El modelo Viewnext-UEx se divide en cuatro áreas de desarrollo y catorce actividades como se puede observar en la Imagen 1 y Tabla 1.

Imagen 1. Modelo de Desarrollo de Software Seguro Viewnext-UEx Fuente: Elaboración propia.

Es importante destacar que el modelo propuesto es flexible dado que, con pequeños cambios, se puede adaptar tanto a los modelos tradicionales como a los orientados a objetos, o a los ágiles. También se considera retroalimentable debido a que todos los fallos de seguridad y la resolución encontrados durante el avance del ciclo de vida son llevados a las fases iniciales del ciclo de vida para evitar su repetición.

Tabla 1. Actividades de seguridad del Modelo Viewnext-UEx según la fase de ejecución y el área de desarrollo en el que se engloban.

Fase de ejecución	Área de desarrollo	Actividad de Seguridad
Todo el ciclo de vida	Políticas	Estrategia y orientación
		Formación
		Definición de riesgos
Toma de requisitos	Metodología SDL	Validación de requisitos
Análisis/Diseño		Modelado de amenazas
Diseño		Revisión de Diseño
Codificación		Revisión de Desarrollo
Pruebas		Testing de Seguridad
Pre-realese		Validación de salidas
Post-realese	Observatorio	Plan de respuesta a incidentes
Todo el ciclo de vida		Observatorio de seguridad
		Repositorio de vulnerabilidades
	Supervisión	Estado del Proyecto
		Evaluación y métricas

Fuente: Elaboración propia.

3. Indicadores de rendimiento y seguridad para validar el Modelo

En este apartado se definen los indicadores considerados necesarios para medir a futuro los resultados de desarrollar software mediante la metodología tradicional (con seguridad únicamente en la fase de pruebas) y el modelo propuesto. Los autores consideran medir indicadores relativos a la productividad del desarrollo e indicadores específicos de seguridad. Como indicadores de rendimiento se propone medir el número global de horas de desarrollo, las horas segmentadas por fases, el tiempo dedicado a la seguridad en cada fase y el tiempo dedicado a resolver las vulnerabilidades. Con respecto a los medidores de seguridad se recomienda el número total de vulnerabilidades, el tipo según sean de arquitectura o de desarrollo y la criticidad en función del estándar Common Vulnerability Score System (FIRST, 2015).

REFERENCIAS

Davis, N., Miller, P. L., Nichols, W. R. y Seacord, R. C. (2009). *TSP-Secure. Proceedings of the Fourth Annual TSP Symposium*.

FIRST. (2015). Common Vulnerability Scoring System v3.0: Specification Document. *Forum of Incident Response and Security Teams (FIRST)*, 1–21. Retrieved from https://www.first.org/cvss/specification-document

Lipner, S. y Howard, M. (2004). The Trustworthy Computing Security Development Lifecycle.

McGraw, G., Chess, B. y Miques, S. (2011). Building security In maturity model. *2012 Faulkner Information Services*. https://doi.org/10.1126/science.1130-a

Microsoft Corporation (2010). Agile Development Using Microsoft Security Development Lifecycle. Retrieved June 19, 2017, from https://www.microsoft.com/en-us/SDL/Discover/sdlagile.aspx

National Cryptologic Center Computer Emergency Team Response (CCN-CERT) (2018). *Cyberthreats and tendencies. Executive Summary 2018*.

National Cryptologic Center Computer Emergency Team Response (CCN-CERT) (2019). *Cyber Threats and Cyber Security 2019*.

OWASP Project (n.d.). Comprehensive, Lightweight Application Security Process. Retrieved from https://www.owasp.org/index.php/CLASP_Concepts

OWASP Project (2009). Software Assurance Maturity Model. Retrieved from http://www.opensamm.org/downloads/SAMM-1.0.pdf

Redwood Shores, C. O. C. (2011). Oracle Software Security Assurance. Retrieved from https://www.oracle.com/support/assurance/index.html

APUNTES BIOGRÁFICOS

José Carlos Sancho Núñez (Cáceres, 6 de febrero de 1987) es Ingeniero Técnico en Informática de Gestión (2010) e Ingeniero Informático (2014) por la Universidad de Extremadura. Estudiante del Programa de Doctorado Tecnologías Informáticas y miembro del Grupo de Ingeniera de Medios (GIM), ambos de la Universidad de Extremadura.

Contacto: **jcsanchon@unex.es**