Arquitecturas seguras a partir de requerimientos patrones de seguridad y vulnerabilidades

Miguel Solinas[1] and Leandro Antonelli[2]

[1] Laryc - FCEFyN, Universidad Nacional de Córdoba, Córdoba, AR
miguel.solinas@unc.edu.ar
[2] Lifia - Fac. de Informática, Universidad Nacional de La Plata, La Plata, AR
lanto@lifia.info.unlp.edu.ar

Abstract. La construcción de software seguro continúa siendo un desafío que trata los requerimientos de seguridad como no funcionales y los resuelve en etapas tardías del proceso de desarrollo de software. En un contexto de Continuous Software Engineering (CSE), considerada como la práctica corriente para la construcción de software, raramente existe una etapa de análisis de una arquitectura que garantice minimizar riesgos de vulnerabilidades conocidas. Esto conduce a la construcción de software inseguro que demanda acciones de detección posterior de vulnerabilidades. En el mejor de los casos serán eliminadas con un gran costo de recursos. O directamente se libera software con vulnerabilidades conocidas que tendrán que ser eliminadas en una etapa de actualización en versiones posteriores. Este trabajo describe las ideas seminales para enfrentar el desafío de construir un método y un sistema de recomendación de arquitecturas seguras de software.

Keywords. Software Engineering, Vulnerabilities, Security Patterns.

1 Introducción

Bajo las prácticas del Continuous Software Engineering (CSE)[1], las principales actividades orientadas a mejorar la seguridad se enfocan en revisiones de código y el uso de herramientas automatizadas de análisis, estático o dinámico, de vulnerabilidades sobre el artefacto ya construido. No es habitual utilizar una visión holística de la seguridad, aplicando por ejemplo, acciones que contemplen la especificación de requerimientos funcionales. Aún reconociendo que los costos de resolver problemas de seguridad en etapas tardías del proceso de desarrollo de software es varios órdenes de magnitud más costoso que solucionar los problemas de seguridad en etapas tempranas del mismo proceso.

2 Estado del arte

Hace al menos 10 años que la problemática de construcción de software seguro está vigente [2]. Lo que no se tuvo en cuenta en aquellos trabajos, fue el registro de vulnerabilidades. En ese momento sumaban 53.263 entradas y en mayo de este año han superado las 176.000. En todo este tiempo se han hecho esfuerzos para tomar fallas observadas en el mundo real que existen en el código, abstraerlas y agruparlas en clases comunes que representan vulnerabilidades potenciales más generales. Esto redujo ese número a una estructura relativamente adecuada para ser accesible y útil a un conjunto diverso de audiencias y propósitos.

El concepto original de lo que luego se convertiría en la lista de Common Vulnerabilities and Exposures (CVE) [4] fue presentado en enero de 1999 [3]. Han pasado más de 20 años de registro y existe un “conocimiento” contenido en la terna CVE, Common Weakness Enumeration (CWE) [5] y Common Attack Pattern Enumerations and Classifications (CAPEC) [6] que aun no se ha consolidado como una "tabla periódica" de exposiciones pero esos registros empiezan a estudiarse para mejorar de la seguridad del software.

El problema que se presenta en los escenarios de desarrollo propuestos por las metologías ágiles y las CSE es la falta de conocimiento, herramientas e interés por incorporar la seguridad como un derivado de la experiencia de vulnerabilidades conocidas [1].

Por otro lado, en el área de conocimiento de la seguridad del software los patrones de seguridad son una ayuda para hacer cumplir principios de seguridad en el diseño ya que encapsulan conocimiento experto en una formato reutilizable que es la base para la construcción de arquitecturas mas robustas, seguras y validadas. En su documentación, pueden hasta incluir código a fin de ayudar a automatizar etapas de implementación. Los patrones de seguridad siguen siendo un condimento para proporcionar seguridad a la construcción de software.

3 Planteo del problema y contribuciones

Las preguntas que surgen es ¿por qué no se generaliza la utilización de aquellos patrones? ¿dan cobertura a todos los problemas de seguridad? ¿contemplan alguna relación con las exposiciones que generan las vulnerabilidades conocidas? La respuesta a esta última pregunta conduce a una solución eficáz y eficiente para mejorar la seguridad del software.

4 Metodología y enfoque de investigación

Se utilizará una metodología analítica experimental con un enfoque cuantitativo a fin

de generalizar resultados a partir de un caso de estudio básico y su evolución hacia escenarios mas complejos. La novedad en el abordaje del problema de construcción de software seguro, está en incorporar los registros de vulnerabilidades, desde sus clases, conjuntamente con patrones propios del dominio para mejorar la seguridad de una arquitectura en una etapa temprana del proceso de desarrollo.

5 Plan de evaluación

Se hará un caso de estudio para evaluar la aplicabilidad de la propuesta y cuestionarios a los involucrados para evaluar la usabilidad.

6 Resultados preliminares o intermedios

No hay resultados aun por ser una etapa muy temprana.

7 Conclusiones y lecciones aprendidas

Si los registros de vulnerabilidades y las clases comunes que representan están evolucionando en la dirección de una "tabla periódica" de exposiciones, será posible descubrir un método y construir un sistema para que esas clases puedan ser vinculadas con requerimientos y patrones de seguridad a través de una ontología. De no ser posible su mapeo directo hoy, dado que el número de patrones de seguridad es reducido, al menos será posible generar recomendaciones esenciales para el diseño de arquitecturas mas seguras, desde etapas tempranas del proceso de desarrollo de software en un contexto de CSE.

8 Ph.D. Stage

Early Stage. El resumen que aquí se presenta se está completando para presentarse como propuesta en el Doctorado en Informática en la UNLP.

Agredecimientos a mi director de Maestría, el Dr. Leandro Antonelli quien después de diez años tuvo la generosidad de escuchar mis ideas, aconsejarme y alentarme para enfrentar el desafío de un Doctorado.

References

1. Roshan N. Rajapakse, Mansooreh Zahedi, M. Ali Babar, Haifeng Shen,

Challenges and solutions when adopting DevSecOps: A system- atic review, Information and Software Technology, Volume 141, 2022, https://doi.org/10.1016/j.infsof.2021.106700.
2. Solinas, M., Elicitación y trazabilidad de requerimientos utilizando patrones de se- guridad, Tesis Maestría, 2012, https://doi.org/10.35537/10915/4211.
3. Mann, David E. and Christey Steven M., Towards a Common Enumeration of Vul- nerabilities, 2nd Workshop on Research with Security Vulnerability Databases on January 21-22, 1999 at Purdue University in West Lafayette, Indiana, USA
4. CWE https://cwe.mitre.org/index.html. Last accessed 13 May 2022.
5. CVE https://www.cve.org/. Last accessed 13 May 2022.
6. CAPEC https://capec.mitre.org/. Last accessed 13 May 2022.