Unidad 3 / Escenario 6

Lectura Fundamental

Otros modelos de seguridad en SDLC

Contenido

Palabras clave:
Seguridad, desarrollo, agiles, scrum, identificación, implementación, verificación, validación, hecho.

1. Introducción

En el escenario anterior y en las unidades anteriores, hemos visto la seguridad aplicada a modelos tradicionales de desarrollo de software, para ser específicos, hemos venido aprendiendo de seguridad aplicada al modelo de desarrollo de software en cascada.

Sin embargo, es importante abrir un poco este enfoque y observar como se implementa seguridad en modelos ágiles de desarrollo de software, los cuales han venido siendo implementados cada vez más por empresas privadas y públicas. Los modelos de seguridad vistos hasta ahora se enfocan en modelos tradicionales de desarrollo, donde se cuenta con bastante tiempo para la etapa de planeación, algo con lo que cuentan los modelos ágiles de desarrollo.

En este escenario aprenderemos como se implementa seguridad en el modelo Scrum, el cuál hace parte de las metodologías ágiles de desarrollo de software y que es cada vez más utilizado por las compañías. En primera instancia, observaremos una descripción de que es Scrum, posteriormente entenderemos que es Scrum Seguro (Pohl, 2015) y cada uno de sus componentes o partes.

Finalmente observaremos la relación que puede tener lo visto en seguridad en el modelo de desarrollo en cascada con Scrum.

2. Scrum

Scrum hace parte de las metodologías de desarrollo de software ágiles. Tiene como característica principal la entrega de resultados parciales del producto final. Las entregas son priorizadas teniendo en cuenta al menos los siguientes aspectos:

- Necesidades del cliente
- Costo
- Urgencia
- Importancia

Scrum básicamente esta modelado como un proceso que se desarrolla en periodos de tiempo denominados iteraciones o “sprints”. Las iteraciones normalmente tardan entre 15 y 30 días de trabajo y sus entregables van aportando parcialmente al producto final. Dichas entregas deben satisfacer la funcionalidad y características solicitadas por el cliente, lo cual es conocido como historias de usuario.

Las historias de usuario se registran en un documento denominado reserva de producto o backlog, existiendo un backlog inicial y uno depurado.

Veamos como se estructura cada sprint:

- Planificación: en la planificación de la iteración inicialmente se seleccionan y priorizan los requisitos del cliente. Posteriormente se identifican todas las tareas requeridas para desarrollar dichos requisitos.
- Ejecución: consiste en la ejecución de las tareas, las cuales son revisadas mediante una reunión diaria o sincronizada. De esta revisión se identifican posibles inconvenientes que dificulten lograr los objetivos, tomando las acciones correctivas a que haya lugar.
- Definición de Hecho: En esta etapa se realiza la definición de hecho (DoD por su sigla en inglés), corresponde a los requisitos o características que son aceptables para considerar que una tarea quedó bien hecha y que se podría cerrar.
- Revisión: el último día del sprint se realiza una demostración al cliente de los requisitos ya culminados, quien realiza la respectiva retroalimentación al equipo. Adicionalmente se identifican oportunidades de mejora que servirán para la planeación, ejecución y revisión del siguiente sprint.

¿Sabía qué...?

La implementación de seguridad en modelos de desarrollo en cascada no es aplicable a modelos ágiles por su corta planeación.

3. Seguridad en Scrum

Introducir seguridad en el modelo de desarrollo ágil Scrum básicamente implica incluir tareas de seguridad conocidas como “Spikes”. Dichos spikes contienen el análisis, diseño y verificación de los componentes de seguridad que debería tener la aplicación.

La seguridad en scrum se divide en 4 componentes o partes, las cuales se han denominado así: Identificación, Implementación, Verificación y Definición de hecho o acabado. Dichos componentes de seguridad están relacionados a uno o más componentes de Scrum.

En las siguientes secciones veremos un detalle de cada uno de los componentes:

3.1. Identificación

El componente identificación tiene como objetivo encontrar posibles brechas o problemas de seguridad durante el desarrollo del software. Este componente hace parte del backlog inicial, del backlog depurado, de la reunión diaria y de la revisión.

El componente de identificación inicia con la descripción, por parte de los interesados, de la pérdida de valor que generaría un ataque informático, cuyo eventual impacto es pérdida, robo o manipulación de datos.

Una vez identificados estos escenarios (ataques, principios de seguridad, problemas, tareas), los miembros del equipo evalúan casos de abuso o desuso y los priorizan por el nivel de riesgo. Para este ejercicio se recomienda incluir o revisar ejercicios similares que se hayan ejecutado en el pasado y que puedan reutilizarse o aportar.

Para implementar este componente, a cada historia de usuario se le relaciona una o varias "etiquetas de seguridad", es decir los escenarios identificados. Al volverse parte del backlog, hará parte de cada uno de los sprints, garantizando así su seguimiento constante. Cada historia de usuario se relaciona con la etiqueta de seguridad mediante una "marca de seguridad" y una conexión.

Las etiquetas de seguridad incluyen una descripción completa y detallada del escenario de seguridad identificado, incluye además recomendaciones sobre como trabajar en dichos escenarios y adicionalmente debe hacer parte de la base de datos de conocimientos, información que puede reutilizarse en proyectos posteriores para hacer más eficiente y eficaz un futuro desarrollo.

3.2. 1.2. Implementación

Alineado con el modelo Scrum, la etapa de implementación busca aplicar a llevar a cabo de manera ágil los requerimientos de seguridad identificados hasta el momento. En este sentido, es importante recordar que en el backlog del producto se deben incluir las etiquetas de seguridad, de manera que los escenarios de seguridad permanezcan siempre visibles y que hagan parte de los sprints.

Las marcas de seguridad, así como las historias de usuario, se pueden dividir o fraccionar en tareas. Sin embargo, cuando esto se hace, es necesario mantener en mente el escenario inicial que se busca cubrir, por lo que la división en tareas debe incluir las marcas de seguridad y conectores relacionados con el

escenario a trabajar. Esto asegura que los desarrolladores estén siempre conscientes del escenario original de seguridad que se necesita trabajar.

3.3. Verificación

La verificación, como su nombre lo indica, consiste en realizar la verificación de que el requerimiento de seguridad ha sido correctamente implementado. Dependiendo del conocimiento en seguridad que tenga un desarrollador, puede requerirse que la verificación de una etiqueta de seguridad requiera de:

- Capacitación
- Apoyo de un desarrollador con más conocimientos en seguridad
- Contratar un recurso externo

Cuando los resultados de la verificación no son satisfactorios, es necesario comentarlo en la reunión diaria para tomar las acciones que sean necesarias y así lograr el cumplimiento requerido.

3.4. Definición de hecho o acabado

La definición de lo hecho está directamente relacionada con la verificación, en consecuencia, la verificación de una marca de seguridad debe hacer parte o es una condición para considerar que se a acabado la tarea.

Teniendo en cuenta las implicaciones que puede tener una falla de seguridad en un software y las tendencias de ataques cada vez más crecientes, es muy importante confirmar que la definición de lo hecho está muy bien realizada y completada al 100%, cualquier equivocación en esta parte pude tener resultados desastrosos.

¿Sabía qué...?

Comprobar la implementación de seguridad en Scrum es mucho más rápido que en el modelo de desarrollo en cascada.

4. Consideraciones finales

Como pudimos ver, Scrum es un modelo ágil de desarrollo y la implementación de seguridad en dicho modelo también lo es. Es importante aclarar que todas las actividades de seguridad que vimos para el modelo de desarrollo en cascada se deben implementar en Scrum pero con unas consideraciones diferentes, por lo que a continuación se muestra una orientación al respecto:

- ***Requerimientos:*** Se debe considerar en el sprint los requerimientos del cliente y si existen requerimientos legales de seguridad que se deban cumplir. Durante la definición de los requerimientos se puede tener en cuenta los casos de desuso y de abuso que permitan identificar desde el punto de vista de seguridad qué requiere la aplicación.
- ***Análisis de riesgos:*** El análisis de riesgos se debe realizar de manera simplificada, pero considerando los riesgos más peligrosos e importantes, pues teniendo en cuenta que es un modelo ágil esta tarea podría tardar mucho y afectarla entrega parcial. Sin embargo se deben contemplar todos los riesgos que garanticen un nivel de tranquilidad suficiente, evitando así tener una sensación de desconfianza por tener que hacer o realizar las cosas "de afán".
- ***Codificación segura:*** Teniendo en cuenta que Scrum está orientado a codificar, esta es tal ves la actividad que más visibilidad tiene dentro del modelo Scrum. Para tal fin, es muy recomendable capacitar a los desarrolladores para que ellos se apropien de la responsabilidad y contribuyan a los resultados ágiles que demanda el modelo. Como alternativa, se recomienda obtener el apoyo de un recurso externo con la experiencia y conocimientos en codificación segura, de manera que pueda acompañar al desarrollador mientras realiza el aprendizaje requerido. Acá nuevamente se recuerda que un factor motivante para el desarrollador sería certificarlo en temas de codificación segura.
- ***Revisión de código:*** Hace parte de la verificación del sprint, al ser entregas parciales del producto es una tarea que no consumirá mucho tiempo. Sin embargo, es importante contar con las herramientas y recursos requeridos para realizar la actividad, si internamente la compañía no cuenta con ellas, se recomienda realizar la contratación de una compañía o recurso externo que pueda apoyar la actividad.
- ***Escaneo de vulnerabilidades:*** El escaneo de vulnerabilidades es una de las verificaciones de seguridad más rápidas que se puede realizar, pues básicamente las herramientas o escáneres son configurados con las URL y/o direcciones IP, se inicia el escaneo y después de un periodo de tiempo se obtiene el informe. Teniendo en cuenta que Scrum es una metodología ágil, se recomienda

que durante la planeación del sprint se configure y programe el escaneo de vulnerabilidades, de manera que se ahorre tiempo. De igual manera, teniendo en cuenta la facilidad en su ejecución, es recomendable que sea el mismo desarrollador quien sea el responsable por la ejecución del mismo.

- ***Pentesting:*** El pentesting debe definirse y planearse muy bien, pues es un control que puede tomar mucho tiempo. Algunas compañías que tienen implementado Scrum, han decidido no incluir el pentesting en todos los sprints, pero si lo hacen en el sprint del medio (50% del producto final) y en el sprint final, pues confían en que cerrando las vulnerabilidades encontradas durante los escaneos (esta actividad si la incluyen en todos los sprints) el pentesting no debería salir tan mal. Sin embargo, lo ideal desde el punto de vista de seguridad es ejecutar pentesting en cada sprint, claro está con los recursos y la capacidad suficiente para obtener resultados ágiles y de esta manera no atrasar las entregas parciales prometidas.
- ***Verificación de parcheo:*** Es importante realizar la verificación de parcheo en el sentido en que cierra las brechas de seguridad encontradas durante el análisis de vulnerabilidades y durante el pentesting. Hace parte de la definición de lo hecho, por lo que es estrictamente necesario confirmar el cierre de las brechas para dar por cerrada la marca de seguridad, de no ser así se entiende que la entrega parcial no cuenta con un nivel mínimo de seguridad.
- ***Hardening:*** Generalmente no es una actividad que realicen directamente los desarrolladores (para quienes fue creado Scrum) ya que está más orientado a un tema de infraestructura. Sin embargo, desde el punto de vista de seguridad es una actividad muy importante pues disminuye la superficie de ataque significativamente, motivo por el cual es necesario incluirlo en el sprint, de manera que se le haga constante seguimiento hasta que se acepte como hecho.
- ***Gestión de cambios:*** Se ejecuta de manera similar como se hace en el modelo de desarrollo de software en cascada. Para cada entregable del sprint, se debe ejecutar el procedimiento que se tenga definido en la compañía, incluyendo el desarrollo del requerimiento de cambio o RFC (Request for change) y la respectiva aprobación del comité de cambios.

En conclusión, todo lo que vimos en el modelo en cascada se debe aplicar a Scrum, básicamente la diferencia está en el alcance de cada una de las actividades, pues en Scrum están acotadas a lo definido para el sprint.

¿Sabía qué...?

Las actividades de seguridad en el modelo en cascada se pueden aplicar a Scrum.

En síntesis...

Se puede y se debe aplicar a todos los modelos de desarrollo de software, tanto los tradicionales como los más modernos.

Referencias bibliográficas

Pohl, C., & Jof, H. (2015). *Secure Scrum: Development of Secure Software with Scrum. SECURWARE 2015: The Ninth International Conference on Emerging Security Information, Systems and Technologies.* Recuperado de https://arxiv.org/pdf/1507.02992.pdf

INFORMACIÓN TÉCNICA

FACULTAD DE
INGENIERÍA, DISEÑO
E INNOVACIÓN

Módulo: Seguridad en el Ciclo de Desarrollo

Unidad 3: Integración de la seguridad con modelos de desarrollo de software existentes

Escenario 6: Otros modelos de seguridad en SDLC

Autor: Miguel Ángel Zambrano Puentes

Asesor Pedagógico: Edwin Mojica Quintero

Diseñador Gráfico: Brandon Steven Ramírez Carrero

Asistente: Ginna Quiroga