Introducción

El proyecto de pruebas de la OWASP

El proyecto de pruebas de la OWASP ha estado en desarrollo durante muchos años. El objetivo del proyecto es ayudar a la gente a entender el *qué*, *por qué*, *cuándo*, *dónde* y *cómo* de las pruebas de aplicaciones web. El proyecto ha entregado un marco de pruebas completo, no sólo una simple lista de verificación o prescripción de cuestiones que deben ser abordadas. Los lectores pueden utilizar este marco como plantilla para construir sus propios programas de pruebas o para calificar los procesos de otras personas. La Guía de Pruebas describe en detalle tanto el marco general de pruebas como las técnicas necesarias para poner en práctica el marco en la práctica.

Escribir la guía de pruebas ha demostrado ser una tarea difícil. Fue un desafío obtener un consenso y elaborar un contenido que permitiera a las personas aplicar los conceptos descritos en la guía, y que al mismo tiempo les permitiera trabajar en su propio entorno y cultura. También fue un desafío cambiar el enfoque de las pruebas de aplicaciones web de pruebas de penetración a pruebas integradas en el ciclo de vida del desarrollo de software.

Sin embargo, el grupo está muy satisfecho con los resultados del proyecto. Muchos expertos de la industria y profesionales de la seguridad, algunos de los cuales son responsables de la seguridad del software en algunas de las mayores empresas del mundo, están validando el marco de pruebas. Este marco ayuda a las organizaciones a probar sus aplicaciones web con el fin de crear software fiable y seguro. El marco no se limita a poner de relieve las áreas de debilidad, aunque esta última es ciertamente un subproducto de muchas de las guías y listas de verificación de la OWASP. Como tal, hubo que tomar decisiones difíciles sobre la idoneidad de ciertas técnicas y tecnologías de ensayo. El grupo comprende plenamente que no todos estarán de acuerdo con todas estas decisiones. Sin embargo, OWASP es capaz de tomar la delantera y cambiar la cultura con el tiempo a través de la conciencia y la educación basada en el consenso y la experiencia.

El resto de la presente guía está organizada de la siguiente manera: esta introducción abarca los requisitos previos de las pruebas de las aplicaciones web y el alcance de las pruebas. También cubre los principios de las pruebas exitosas y las técnicas de prueba, las mejores prácticas para la presentación de informes y los casos de uso para las pruebas de seguridad. El capítulo 3 presenta el marco de pruebas de OWASP y explica sus técnicas y tareas en relación con las diversas fases del ciclo de vida del desarrollo de software. El capítulo 4 cubre cómo comprobar la existencia de vulnerabilidades específicas (por ejemplo, SQL Injection) mediante la inspección del código y las pruebas de penetración.

Medición de la seguridad: la economía del software inseguro

Un principio básico de la ingeniería de software se resume en una cita de Controlling Software Projects: Management, Measurement, and Estimates por Tom DeMarco:

> No puedes controlar lo que no puedes medir.

Las pruebas de seguridad no son diferentes. Desafortunadamente, medir la seguridad es un proceso notoriamente difícil.

Un aspecto que debe destacarse es que las medidas de seguridad se refieren tanto a cuestiones técnicas concretas (por ejemplo, la prevalencia de una determinada vulnerabilidad) como a la forma en que esas cuestiones afectan a la economía del software. La mayoría de los técnicos comprenderán al menos los problemas básicos, o tal vez tengan una comprensión más profunda de las vulnerabilidades. Lamentablemente, pocos son capaces de traducir esos conocimientos técnicos en términos monetarios y cuantificar el costo potencial de las vulnerabilidades para el negocio del propietario de la aplicación. Hasta que esto no ocurra, los CIOs no podrán desarrollar un retorno preciso de la inversión en seguridad y, posteriormente, asignar presupuestos adecuados para la seguridad del software.

Si bien la estimación del costo del software inseguro puede parecer una tarea desalentadora, se ha realizado una cantidad significativa de trabajo en este sentido. Por ejemplo, en junio de 2002, el Instituto Nacional de Estándares de los Estados Unidos (NIST) publicó un estudio sobre el costo del software inseguro para la economía de los Estados Unidos debido a la inadecuada prueba del software. Curiosamente, estiman que una mejor infraestructura de pruebas ahorraría más de un tercio de estos costes, o unos 22.000 millones de dólares al año. Más recientemente, los vínculos entre la economía y la seguridad han sido estudiados por investigadores académicos. La página de Ross Andrerson sobre economía y seguridad tiene más información sobre algunos de estos esfuerzos.

El marco descrito en este documento alienta a las personas a medir la seguridad a lo largo de todo el proceso de desarrollo. Así, pueden relacionar el costo de un software inseguro con el impacto que tiene en el negocio y, en consecuencia, desarrollar procesos empresariales adecuados y asignar recursos para gestionar el riesgo. Recuerde que la medición y el ensayo de las aplicaciones web es incluso más crítica que para otros programas informáticos, ya que las aplicaciones web están expuestas a millones de usuarios a través de Internet.

¿Qué son las pruebas?

Muchas cosas necesitan ser probadas durante el ciclo de vida de desarrollo de una aplicación web, pero ¿qué significa realmente probar? El Diccionario de Inglés de Oxford define “prueba” como:

> **prueba** (sustantivo): procedimiento destinado a establecer la calidad, el rendimiento o la fiabilidad de algo, especialmente antes de su uso generalizado.

Para los propósitos de este documento, la prueba es un proceso de comparación del estado de un sistema o aplicación con un conjunto de criterios. En la industria de la seguridad, las personas frecuentemente hacen pruebas contra un conjunto de criterios mentales que no están ni bien definidos ni completos. Como resultado de esto, muchos forasteros consideran las pruebas de seguridad como un arte negro. El objetivo de este documento es cambiar esa percepción y facilitar que las personas sin conocimientos profundos de seguridad marquen la diferencia en las pruebas.

¿Por qué realizar las pruebas?

Este documento está diseñado para ayudar a las organizaciones a comprender lo que comprende un programa de pruebas, y para ayudarlas a identificar los pasos que deben seguirse para construir y operar un programa de pruebas en aplicaciones web. La guía ofrece una amplia visión de los elementos necesarios para realizar un programa de seguridad integral de aplicaciones web. Esta guía puede utilizarse como guía de referencia y como metodología para ayudar a determinar la brecha entre las prácticas existentes y las mejores prácticas de la industria. Esta guía permite a las organizaciones compararse con sus pares de la industria, comprender la magnitud de los recursos necesarios para probar y mantener el software, o prepararse para una auditoría. Este capítulo no entra en los detalles técnicos de cómo probar una aplicación, ya que la intención es proporcionar un marco organizativo de seguridad típico. Los detalles técnicos sobre cómo probar una aplicación, como parte de una prueba de penetración o revisión de código, se tratarán en las partes restantes de este documento.

¿Cuándo probar?

La mayoría de la gente hoy en día no prueba el software hasta que ya ha sido creado y está en la fase de despliegue de su ciclo de vida (es decir, el código ha sido creado e instanciado en una aplicación web que funciona). Esta es generalmente una práctica muy ineficaz y de costo prohibitivo. Uno de los mejores métodos para evitar que aparezcan errores de seguridad en las aplicaciones de producción es mejorar el ciclo de vida del desarrollo de software (SDLC) incluyendo la seguridad en cada una de sus fases. Un SDLC es una estructura impuesta en el desarrollo de artefactos de software. Si un SDLC no está siendo usado actualmente en tu entorno, ¡es hora de elegir uno! La siguiente figura muestra un modelo SDLC genérico, así como el coste creciente (estimado) de arreglar los fallos de seguridad en dicho modelo.

Figura 2-1: Modelo genérico SDLC

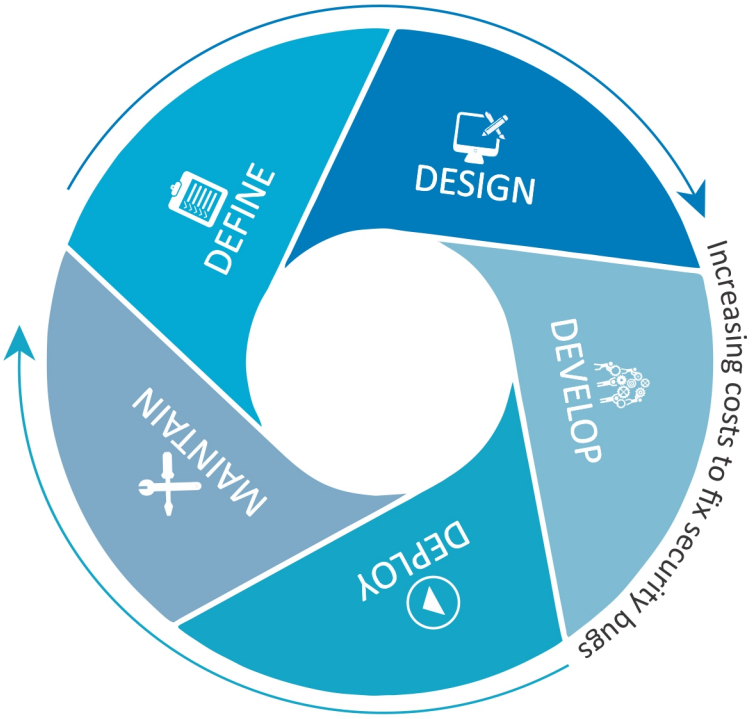

Figure 1: Generic SDLC Model

Las compañías deberían inspeccionar su SDLC general para asegurar que la seguridad es una parte integral del proceso de desarrollo. Los SDLC deberían incluir pruebas de seguridad para asegurar que la seguridad está adecuadamente cubierta y que los controles son efectivos a lo largo del proceso de desarrollo.

¿Qué hay que probar?

Puede ser útil pensar en el desarrollo de software como una combinación de personas, procesos y tecnología. Si estos son los factores que "crean" el software, entonces es lógico que estos sean los factores que deben ser probados. Hoy en día, la mayoría de la gente generalmente prueba la tecnología o el propio software.

Un programa de pruebas eficaz debe tener componentes que comprueben lo siguiente:

Personas - para asegurar que haya una educación y concienciación adecuadas;

Proceso - para asegurar que haya políticas y normas adecuadas y que la gente sepa cómo seguir esas políticas;

Tecnología - para asegurar que el proceso ha sido efectivo en su implementación.

A menos que se adopte un enfoque holístico, el hecho de probar sólo la implementación técnica de una aplicación no pondrá al descubierto las vulnerabilidades de gestión u operativas que podrían estar presentes. Poniendo a prueba a las personas, las políticas y los procesos, una organización puede detectar problemas que más tarde se manifestarían en defectos en la tecnología, erradicando así los errores tempranamente e identificando las causas fundamentales de los defectos. De igual modo, probar sólo algunos de los problemas técnicos que pueden estar presentes en un sistema dará lugar a una evaluación incompleta e inexacta de la postura de seguridad.

Denis Verdon, Jefe de Seguridad de la Información de Fidelity National Financial, presentó una excelente analogía de esta idea errónea en la Conferencia AppSec 2004 de la OWASP en Nueva York:

> Si los coches se construyeran como aplicaciones… las pruebas de seguridad asumirían sólo el impacto frontal. Los coches no serían sometidos a pruebas de rodaje, ni a pruebas de estabilidad en maniobras de emergencia, eficacia de los frenos, impacto lateral y resistencia al robo.

Retroalimentación y comentarios

Como en todos los proyectos de OWASP, agradecemos los comentarios y la retroalimentación. Especialmente nos gusta saber que nuestro trabajo está siendo utilizado y que es efectivo y preciso.

Principios de la prueba

Hay algunos conceptos erróneos comunes cuando se desarrolla una metodología de pruebas para encontrar errores de seguridad en el software. Este capítulo cubre algunos de los principios básicos que los profesionales deben tener en cuenta cuando realizan pruebas de seguridad en el software.

No hay ninguna bala de plata

Aunque es tentador pensar que un escáner de seguridad o un cortafuegos de aplicación proporcionará muchas defensas contra los ataques o identificará una multitud de problemas, en realidad no hay una bala de plata para el problema del software inseguro. El software de evaluación de la seguridad de las aplicaciones, si bien es útil como primera pasada para encontrar frutos de poca monta, suele ser inmaduro e ineficaz en las evaluaciones a fondo o para proporcionar una cobertura de pruebas adecuada. Recuerde que la seguridad es un proceso y no un producto.

Pensar estratégicamente, no tácticamente

Los profesionales de la seguridad se han dado cuenta de la falacia del modelo de parchear y penetrar que fue omnipresente en la seguridad de la información durante la década de 1990. El modelo de parchear y penetrar implica arreglar un error reportado, pero sin la investigación adecuada de la causa de fondo. Este modelo suele asociarse con la ventana de vulnerabilidad, también denominada ventana de exposición, que se muestra en la figura siguiente. La evolución de las vulnerabilidades en los programas informáticos comunes utilizados en todo el mundo ha demostrado la ineficacia de este modelo. Para más información sobre las ventanas de exposición, véase Schneier sobre Seguridad.

Estudios de vulnerabilidad como el Informe sobre las amenazas a la seguridad en Internet de Symantec han demostrado que con el tiempo de reacción de los atacantes en todo el mundo, la típica ventana de vulnerabilidad no proporciona suficiente tiempo para la instalación de parches, ya que el tiempo que transcurre entre que se descubre una vulnerabilidad y se desarrolla y publica un ataque automatizado contra ella disminuye cada año.

Hay varias suposiciones incorrectas en el modelo de parchear y penetrar. Muchos usuarios creen que los parches interfieren con las operaciones normales o que pueden romper las aplicaciones existentes. También es incorrecto suponer que todos los usuarios están al tanto de los parches recién publicados. Por consiguiente, no todos los usuarios de un producto aplicarán parches, ya sea porque piensan que los parches pueden interferir en el funcionamiento del software o porque desconocen la existencia del parche.

Figura 2-2: Ventana de vulnerabilidad

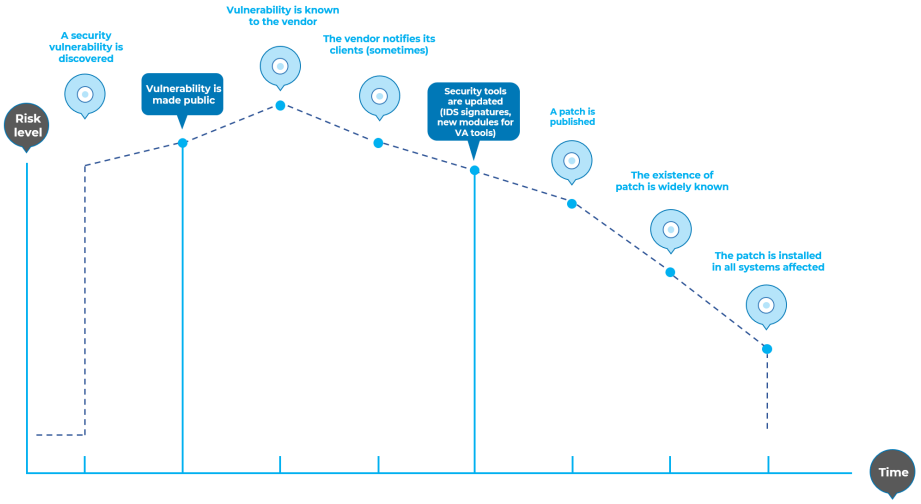

Figure 2: Window of Vulnerability

Es esencial construir la seguridad en el Ciclo de Vida del Desarrollo de Software (SDLC - *Software Development Life Cycle -SDLC*, en inglés) para prevenir problemas de seguridad recurrentes dentro de una aplicación. Los desarrolladores pueden construir la seguridad en el SDLC desarrollando estándares, políticas y directrices que encajen y funcionen dentro de la metodología de desarrollo. El modelado de amenazas y otras técnicas deberían utilizarse para ayudar a asignar los recursos apropiados a aquellas partes de un sistema que están en mayor riesgo.

El SDLC es el rey

El SDLC es un proceso bien conocido por los desarrolladores. Al integrar la seguridad en cada fase del SDLC, permite un enfoque holístico de la seguridad de las aplicaciones que aprovecha los procedimientos ya existentes en la organización. Hay que tener en cuenta que, aunque los nombres de las distintas fases pueden cambiar dependiendo del modelo de SDLC utilizado por una organización, cada fase conceptual del arquetipo SDLC se utilizará para desarrollar la aplicación (es decir, definir, diseñar, desarrollar, desplegar, mantener). Cada fase tiene consideraciones de seguridad que deberían formar parte del proceso existente, para asegurar un programa de seguridad rentable y completo.

Existen varios marcos SDLC seguros que proporcionan asesoramiento tanto descriptivo como prescriptivo. El hecho de que una persona reciba asesoramiento descriptivo o prescriptivo depende de la madurez del proceso de SDLC. Esencialmente, el asesoramiento prescriptivo muestra cómo debería funcionar el SDLC seguro, y el descriptivo muestra cómo se utiliza en el mundo real. Ambos tienen

su lugar. Por ejemplo, si no se sabe por dónde empezar, un marco prescriptivo puede proporcionar un menú de posibles controles de seguridad que se pueden aplicar dentro del SDLC. El asesoramiento descriptivo puede entonces ayudar a impulsar el proceso de decisión presentando lo que ha funcionado bien para otras organizaciones. Los SDLC seguros descriptivos incluyen el BSIMM-V; y los SDLC seguros prescriptivos incluyen el Modelo de Madurez de Garantía de Software Abierto (OpenSAMM) de la OWASP y la ISO/IEC 27034 Partes 1-8, partes de las cuales están todavía en desarrollo.

Prueba temprana y prueba frecuente

Cuando un error se detecta a tiempo en el SDLC puede ser abordado más rápido y a un menor costo. Un fallo de seguridad no es diferente de un fallo funcional o basado en el rendimiento en este sentido. Un paso clave para que esto sea posible es educar a los equipos de desarrollo y de control de calidad sobre los problemas de seguridad comunes y las formas de detectarlos y prevenirlos. Aunque las nuevas bibliotecas, herramientas o lenguajes pueden ayudar a diseñar programas con menos errores de seguridad, constantemente surgen nuevas amenazas y los desarrolladores deben ser conscientes de las amenazas que afectan al software que están desarrollando. La educación en pruebas de seguridad también ayuda a los desarrolladores a adquirir la mentalidad adecuada para probar una aplicación desde la perspectiva de un atacante. Esto permite a cada organización considerar los problemas de seguridad como parte de sus responsabilidades actuales.

Comprender el alcance de la seguridad

Es importante saber cuánta seguridad requerirá un proyecto determinado. Los bienes que se van a proteger deben recibir una clasificación que indique cómo se van a manejar (por ejemplo, confidencial, secreto, alto secreto). Se debe discutir con el consejo legal para asegurar que se cumpla cualquier requisito de seguridad específico. En los Estados Unidos, los requisitos pueden provenir de reglamentos federales, como la Ley Gramm-Leach-Bliley, o de leyes estatales, como la SB-1386 de California. En el caso de las organizaciones con sede en los países de la Unión Europea, pueden aplicarse tanto la reglamentación específica del país como las directivas de la Unión Europea. Por ejemplo, la Directiva 96/46/CE4 obliga a tratar los datos personales en las solicitudes con el debido cuidado, cualquiera que sea la aplicación.

Desarrollar la mentalidad correcta

Probar con éxito una aplicación para las vulnerabilidades de seguridad requiere pensar "fuera de la caja". Los casos de uso normal probarán el comportamiento normal de la aplicación cuando un usuario la está usando de la manera que se

espera. Una buena prueba de seguridad requiere ir más allá de lo que se espera y pensar como un atacante que está tratando de romper la aplicación. El pensamiento creativo puede ayudar a determinar qué datos inesperados pueden hacer que una aplicación falle de manera insegura. También puede ayudar a encontrar cualquier suposición hecha por los desarrolladores web que no siempre es cierta, y cómo esas suposiciones pueden ser subvertidas. Una de las razones por las que las herramientas automatizadas hacen un mal trabajo de comprobación de vulnerabilidades es que las herramientas automatizadas no piensan de forma creativa. El pensamiento creativo debe hacerse caso por caso, ya que la mayoría de las aplicaciones web se están desarrollando de una manera única (incluso cuando se utilizan marcos de trabajo comunes).

Comprender el sujeto

Una de las primeras iniciativas importantes de todo buen programa de seguridad debería ser exigir la documentación exacta de la aplicación. La arquitectura, los diagramas de flujo de datos, los casos de uso, etc., deben registrarse en documentos oficiales y ponerse a disposición para su examen. Los documentos de especificación técnica y de aplicación deben incluir información que enumere no sólo los casos de uso deseados, sino también cualquier caso de uso específicamente desautorizado. Por último, es bueno contar por lo menos con una infraestructura de seguridad básica que permita la vigilancia y el seguimiento de los ataques contra las aplicaciones y la red de una organización (por ejemplo, los sistemas IDS).

Usar las herramientas adecuadas

Si bien ya hemos dicho que no existe una herramienta de bala de plata, las herramientas juegan un papel fundamental en el programa de seguridad general. Hay una gama de herramientas comerciales y de código abierto que pueden automatizar muchas tareas de seguridad rutinarias. Estas herramientas pueden simplificar y acelerar el proceso de seguridad ayudando al personal de seguridad en sus tareas. Sin embargo, es importante entender exactamente lo que estas herramientas pueden y no pueden hacer para que no se vendan en exceso o se utilicen incorrectamente.

El Diablo está en los detalles

Es fundamental no realizar un examen de seguridad superficial de una solicitud y considerarla completa. Esto infundirá una falsa sensación de confianza que puede ser tan peligrosa como no haber hecho una revisión de seguridad en primer lugar. Es vital revisar cuidadosamente los hallazgos y eliminar cualquier falso positivo que pueda quedar en el informe. Informar de un hallazgo de seguridad incorrecto puede a menudo socavar el mensaje válido del resto del informe de

seguridad. Se debe tener cuidado en verificar que cada posible sección de la lógica de la aplicación ha sido probada, y que cada escenario de caso de uso fue explorado para posibles vulnerabilidades.

Usar el código fuente cuando esté disponible

Si bien los resultados de las pruebas de penetración de la caja negra pueden ser impresionantes y útiles para demostrar la forma en que se exponen las vulnerabilidades en un entorno de producción, no son la forma más eficaz o eficiente de asegurar una aplicación. Es difícil para las pruebas dinámicas probar toda la base de código, en particular si existen muchas declaraciones condicionales anidadas. Si el código fuente de la aplicación está disponible, debería entregarse al personal de seguridad para ayudarles en su revisión. Es posible descubrir vulnerabilidades dentro de la fuente de la aplicación que se pasarían por alto durante una intervención de caja negra.

Desarrollar métricas

Una parte importante de un buen programa de seguridad es la capacidad de determinar si las cosas están mejorando. Es importante rastrear los resultados de los compromisos de prueba, y desarrollar métricas que revelen las tendencias de seguridad de la aplicación dentro de la organización.

Las buenas métricas lo demostrarán:

- Si se requiere más educación y capacitación;
- Si hay un mecanismo de seguridad particular que no es claramente comprendido por el equipo de desarrollo;
- Si el número total de problemas relacionados con la seguridad que se encuentran cada mes disminuye.

Las métricas coherentes que pueden generarse de forma automatizada a partir del código fuente disponible también ayudarán a la organización a evaluar la eficacia de los mecanismos introducidos para reducir los errores de seguridad en el desarrollo de programas informáticos. Las métricas no son fáciles de desarrollar, por lo que el uso de métricas estándar como las proporcionadas por el proyecto OWASP Metrics y otras organizaciones es un buen punto de partida.

Documentar los resultados de la prueba

Para concluir el proceso de pruebas, es importante producir un registro formal de las medidas de prueba que se tomaron, por quién, cuándo se realizaron y los detalles de los resultados de las pruebas. Es conveniente acordar un formato aceptable para el informe que sea útil para todas las partes interesadas, entre

las que pueden figurar los promotores, la gestión de proyectos, los propietarios de empresas, el departamento de tecnología de la información, la auditoría y el cumplimiento.

En el informe se debe identificar claramente al propietario del negocio donde existen riesgos materiales, y hacerlo de manera suficiente para obtener su respaldo para las medidas de mitigación subsiguientes. El informe también debe ser claro para el promotor al señalar la función exacta que se ve afectada por la vulnerabilidad y las recomendaciones conexas para resolver los problemas en un lenguaje que el promotor comprenda. El informe también debe permitir que otro probador de seguridad reproduzca los resultados. Escribir el informe no debería ser una carga excesiva para los propios probadores de seguridad. Los probadores de seguridad no son generalmente reconocidos por sus habilidades de escritura creativa, y acordar un informe complejo puede llevar a instancias en las que los resultados de la prueba no estén documentados adecuadamente. El uso de una plantilla de informe de prueba de seguridad puede ahorrar tiempo y asegurar que los resultados se documenten de forma precisa y consistente, y que estén en un formato adecuado para la audiencia.

Explicación de las técnicas de prueba

Esta sección presenta una visión general de alto nivel de las diversas técnicas de prueba que pueden emplearse al construir un programa de pruebas. No presenta metodologías específicas para estas técnicas, ya que esta información se trata en el capítulo 3. Esta sección se incluye para proporcionar un contexto para el marco presentado en el siguiente capítulo y destacar las ventajas y desventajas de algunas de las técnicas que deben considerarse. En particular, nos ocuparemos de ello:

- Inspecciones y revisiones manuales
- Modelado de amenazas
- Revisión del código
- Prueba de penetración

Inspecciones y revisiones manuales

Visión general

Las inspecciones manuales son revisiones humanas que normalmente prueban las implicaciones de seguridad de las personas, las políticas y los procesos. Las inspecciones manuales también pueden incluir la inspección de decisiones tecnológicas como los diseños arquitectónicos. Normalmente se llevan a cabo analizando la documentación o realizando entrevistas con los diseñadores o los propietarios de los sistemas.

Si bien el concepto de inspecciones manuales y revisiones humanas es simple, pueden estar entre las técnicas más poderosas y eficaces disponibles. Preguntando a alguien cómo funciona algo y por qué se implementó de una manera específica, el examinador puede determinar rápidamente si es probable que haya alguna preocupación por la seguridad. Las inspecciones y revisiones manuales son una de las pocas formas de probar el proceso del ciclo de vida del desarrollo de software en sí mismo y de asegurar que existe una política o un conjunto de habilidades adecuadas.

Como ocurre con muchas cosas en la vida, cuando se realizan inspecciones y revisiones manuales se recomienda adoptar un modelo de confianza pero de verificación. No todo lo que se muestra o se dice al probador será exacto. Las revisiones manuales son particularmente buenas para comprobar si las personas entienden el proceso de seguridad, si han sido informadas de la política y si tienen las habilidades apropiadas para diseñar o implementar una aplicación segura.

Otras actividades, incluyendo la revisión manual de la documentación, las políticas de codificación segura, los requisitos de seguridad y los diseños arquitectónicos, deberían realizarse mediante inspecciones manuales.

Ventajas

- No requiere ninguna tecnología de apoyo
- Puede aplicarse a una variedad de situaciones
- Flexible
- Promueve el trabajo en equipo
- Al principio del SDLC

Desventajas

- Puede llevar mucho tiempo
- El material de apoyo no siempre está disponible
- Requiere un pensamiento y una habilidad humana significativa para ser efectivo

Modelización de amenazas

Visión general

El modelado de amenazas se ha convertido en una técnica popular para ayudar a los diseñadores de sistemas a pensar en las amenazas de seguridad que sus sistemas y aplicaciones podrían enfrentar. Por lo tanto, el modelado de amenazas puede considerarse como una evaluación de los riesgos para las aplicaciones.

Permite al diseñador desarrollar estrategias de mitigación de las posibles vulnerabilidades y le ayuda a centrar sus recursos y su atención, inevitablemente limitados, en las partes del sistema que más lo necesitan. Se recomienda que todas las aplicaciones tengan un modelo de amenaza desarrollado y documentado. Los modelos de amenazas deben crearse lo antes posible en el SDLC, y deben revisarse a medida que la aplicación evoluciona y el desarrollo avanza.

Para desarrollar un modelo de amenaza, se recomienda adoptar un enfoque simple que siga el estándar NIST 800-30 para la evaluación de riesgos. Este enfoque implica:

- Descomponer la aplicación - utilizar un proceso de inspección manual para comprender cómo funciona la aplicación, sus activos, funcionalidad y conectividad.
- Definir y clasificar los activos - clasificar los activos en tangibles e intangibles y clasificarlos de acuerdo a la importancia del negocio.
- Explorando las vulnerabilidades potenciales - ya sean técnicas, operacionales o de gestión.
- Exploración de las amenazas potenciales - desarrollar una visión realista de los posibles vectores de ataque desde la perspectiva de un atacante mediante el uso de escenarios de amenaza o árboles de ataque.
- Crear estrategias de mitigación - desarrollar controles de mitigación para cada una de las amenazas que se consideren realistas.

El resultado de un modelo de amenaza en sí mismo puede variar, pero normalmente es una colección de listas y diagramas. La Guía de revisión de código OWASP esboza una metodología de modelado de amenazas de aplicación que puede utilizarse como referencia para probar las aplicaciones en busca de posibles fallos de seguridad en el diseño de la aplicación. No existe una forma correcta o incorrecta de desarrollar modelos de amenazas y realizar evaluaciones de riesgos de información en las aplicaciones.

Ventajas

- La visión práctica del atacante del sistema
- Flexible
- Al principio del SDLC

Desventajas

- Una técnica relativamente nueva
- Los buenos modelos de amenaza no significan automáticamente un buen software

Revisión del código fuente

Panorama general

La revisión del código fuente es el proceso de comprobar manualmente el código fuente de una aplicación web para detectar problemas de seguridad. Muchas vulnerabilidades graves de seguridad no pueden detectarse con ninguna otra forma de análisis o pruebas. Como dice el dicho popular "si quieres saber lo que realmente está pasando, ve directamente a la fuente". Casi todos los expertos en seguridad están de acuerdo en que no hay sustituto para revisar realmente el código. Toda la información para identificar los problemas de seguridad está en el código, en algún lugar. A diferencia de las pruebas de software cerrado de terceros, como los sistemas operativos, cuando se prueban aplicaciones web (especialmente si han sido desarrolladas internamente) el código fuente debería estar disponible para fines de prueba.

Muchos problemas de seguridad no intencionados pero significativos son también extremadamente difíciles de descubrir con otras formas de análisis o pruebas, como las pruebas de penetración. Esto hace que el análisis del código fuente sea la técnica preferida para las pruebas técnicas. Con el código fuente, un probador puede determinar con precisión lo que está sucediendo (o se supone que está sucediendo) y eliminar las suposiciones de las pruebas de caja negra.

Ejemplos de cuestiones que son particularmente propicias para ser encontradas a través de las revisiones de código fuente incluyen problemas de concurrencia, lógica comercial defectuosa, problemas de control de acceso y debilidades criptográficas, así como puertas traseras, troyanos, huevos de pascua, bombas de tiempo, bombas lógicas y otras formas de código malicioso. Estos problemas se manifiestan a menudo como las vulnerabilidades más perjudiciales de los sitios web. El análisis del código fuente también puede ser sumamente eficiente para encontrar problemas de aplicación, como lugares en los que no se ha realizado la validación de los datos o en los que pueden existir procedimientos de control abiertos a fallos. También es necesario revisar los procedimientos operacionales, ya que el código fuente que se está desplegando podría no ser el mismo que el que se está analizando aquí. El discurso del Premio Turing de Ken Thompson describe una posible manifestación de este problema.

Ventajas

- Completitud y eficacia
- Precisión
- Rápido (para revisores competentes)

Desventajas

- Requiere desarrolladores de seguridad altamente calificados

- Pueden faltar problemas en las bibliotecas compiladas
- No pueden detectar fácilmente los errores de tiempo de ejecución
- El código fuente realmente desplegado podría diferir del que se está analizando

Para más información sobre la revisión de códigos, vea el proyecto de revisión de códigos de OWASP.

Prueba de penetración

Visión general

La prueba de penetración ha sido una técnica común utilizada para probar la seguridad de la red durante muchos años. También se conoce comúnmente como pruebas de caja negra o hacking ético. La prueba de penetración es esencialmente el "arte" de probar una aplicación en ejecución de forma remota para encontrar vulnerabilidades de seguridad, sin conocer el funcionamiento interno de la propia aplicación. Típicamente, el equipo de prueba de penetración es capaz de acceder a una aplicación como si fueran usuarios. El probador actúa como un atacante e intenta encontrar y explotar las vulnerabilidades. En muchos casos, el probador recibirá una cuenta válida en el sistema.

Si bien las pruebas de penetración han demostrado ser eficaces en la seguridad de las redes, la técnica no se traduce naturalmente en las aplicaciones. Cuando se realizan pruebas de penetración en redes y sistemas operativos, la mayor parte del trabajo consiste en encontrar, y luego explotar, las vulnerabilidades conocidas en tecnologías específicas. Como las aplicaciones web están casi exclusivamente hechas a medida, las pruebas de penetración en el ámbito de las aplicaciones web se asemejan más a la investigación pura. Se han desarrollado algunas herramientas automatizadas de prueba de penetración, pero teniendo en cuenta la naturaleza a medida de las aplicaciones web, su eficacia por sí sola suele ser escasa.

Muchas personas utilizan las pruebas de penetración de aplicaciones web como su principal técnica de prueba de seguridad. Si bien es cierto que tiene su lugar en un programa de pruebas, no creemos que deba considerarse como la principal o única técnica de prueba. Como escribió Gary McGraw en Software Penetration Testing, "En la práctica, una prueba de penetración sólo puede identificar una pequeña muestra representativa de todos los posibles riesgos de seguridad de un sistema" Sin embargo, las pruebas de penetración enfocadas (es decir, las pruebas que intentan explotar las vulnerabilidades conocidas detectadas en revisiones anteriores) pueden ser útiles para detectar si algunas vulnerabilidades específicas están realmente arregladas en el código fuente desplegado en el sitio web.

Ventajas

- Puede ser rápido (y por lo tanto barato)
- Requiere un conjunto de habilidades relativamente más bajo que la revisión del código fuente
- Prueba el código que está siendo expuesto

Desventajas

- Demasiado tarde en el SDLC
- Sólo pruebas de impacto frontal

La necesidad de un enfoque equilibrado

Con tantas técnicas y enfoques para probar la seguridad de las aplicaciones web, puede ser difícil comprender qué técnicas utilizar o cuándo utilizarlas. La experiencia demuestra que no hay una respuesta correcta o incorrecta a la pregunta de qué técnicas exactamente se deben utilizar para construir un marco de pruebas. De hecho, todas las técnicas deberían utilizarse para probar todas las áreas que necesitan ser probadas.

Aunque está claro que no existe una única técnica que pueda realizarse para cubrir eficazmente todas las pruebas de seguridad y garantizar que se han abordado todos los problemas, muchas empresas adoptan un solo enfoque. El único enfoque utilizado ha sido históricamente la prueba de penetración. La prueba de penetración, aunque es útil, no puede abordar eficazmente muchas de las cuestiones que deben probarse. Simplemente es "demasiado poco y demasiado tarde" en el SDLC.

El enfoque correcto es un enfoque equilibrado que incluye varias técnicas, desde revisiones manuales hasta pruebas técnicas. Un enfoque equilibrado debería cubrir las pruebas en todas las fases del SDLC. Este enfoque aprovecha las técnicas más apropiadas disponibles, dependiendo de la fase actual del SDLC.

Por supuesto, hay momentos y circunstancias en que sólo es posible una técnica. Por ejemplo, considérese la prueba de una aplicación web ya creada, pero en la que la parte que realiza la prueba no tiene acceso al código fuente. En este caso, la prueba de penetración es claramente mejor que no hacer ninguna prueba. Sin embargo, se debe alentar a las partes que realizan las pruebas a que cuestionen supuestos, como el de no tener acceso al código fuente, y a que exploren la posibilidad de realizar pruebas más completas.

Un enfoque equilibrado varía en función de muchos factores, como la madurez del proceso de prueba y la cultura empresarial. Se recomienda que un marco de pruebas equilibrado se parezca a las representaciones que se muestran en las figuras 3 y 4. La siguiente figura muestra una típica representación proporcional

superpuesta en el SLDC. De acuerdo con la investigación y la experiencia, es esencial que las empresas pongan un mayor énfasis en las primeras etapas de desarrollo.

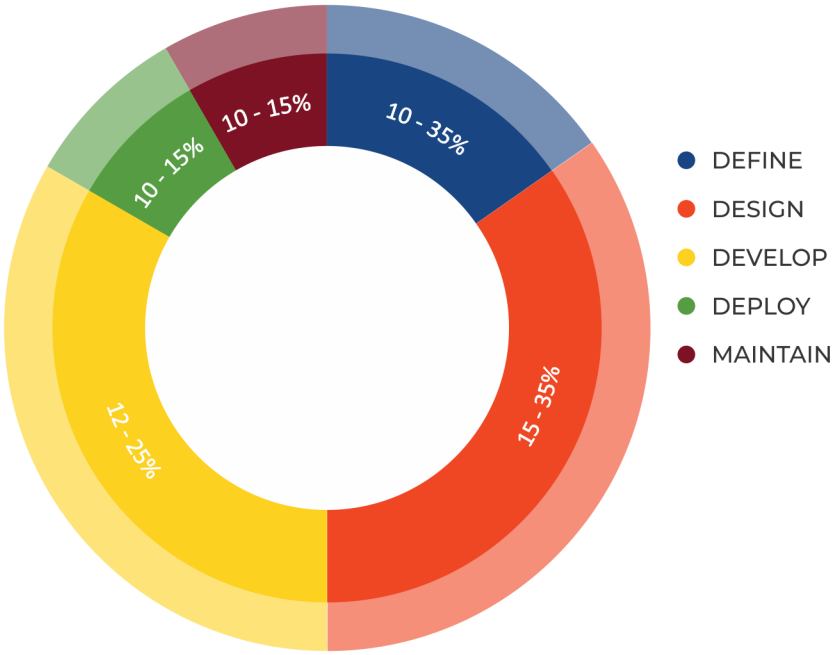

Figure 3: Proportion of Test Effort in SDLC

Figura 2-3: Proporción del esfuerzo de prueba en el SDLC

La siguiente figura muestra una típica representación proporcional superpuesta a las técnicas de prueba.

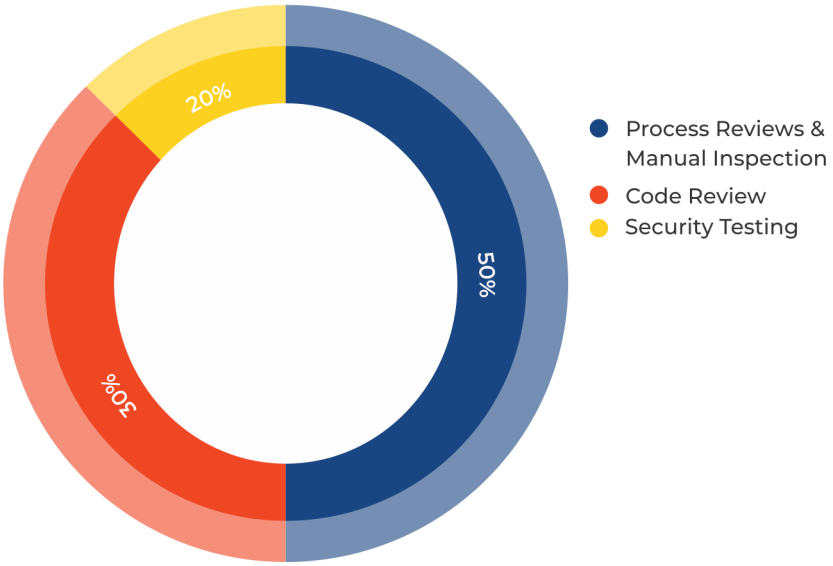

Figura 2-4: Proporción del esfuerzo de prueba según la técnica de prueba

Una nota sobre los escáneres de aplicaciones web

Muchas organizaciones han comenzado a utilizar escáneres automáticos de aplicaciones web. Aunque sin duda tienen un lugar en un programa de pruebas, hay que destacar algunas cuestiones fundamentales sobre por qué se cree que la automatización de las pruebas de caja negra no es (ni será nunca) completamente eficaz. Sin embargo, el hecho de destacar estas cuestiones no debe desalentar el uso de los escáneres de aplicaciones web. El objetivo es más bien asegurar que se comprendan las limitaciones y que los marcos de pruebas se planifiquen adecuadamente.

Es útil entender la eficacia y las limitaciones de las herramientas automatizadas de detección de vulnerabilidades. Con este fin, el Proyecto de Referencia OWASP es un conjunto de pruebas diseñado para evaluar la velocidad, cobertura y precisión de las herramientas y servicios automatizados de detección de vulnerabilidades de software. La evaluación comparativa puede ayudar a probar las capacidades de estas herramientas automatizadas, y ayudar a hacer explícita su utilidad.

Los siguientes ejemplos muestran por qué las pruebas de caja negra automatizadas pueden no ser eficaces.

Ejemplo 1: Parámetros mágicos

Imagina una simple aplicación web que acepta un par de nombre-valor de "magia" y luego el valor. Por simplicidad, la solicitud GET puede ser: `http://www.host/application?magic=value`

Para simplificar aún más el ejemplo, los valores en este caso sólo pueden ser caracteres ASCII a - z (en mayúsculas o minúsculas) y números enteros de 0 - 9.

Los diseñadores de esta aplicación crearon una puerta trasera administrativa durante las pruebas, pero la ofuscaron para evitar que el observador casual la descubriera. Al enviar el valor *sf8g7sfjdsurtsdieerwqredsgnfg8d* (30 caracteres), el usuario se conectará y se le presentará una pantalla administrativa con control total de la aplicación. La solicitud HTTP es ahora: `http://www.host/application?magic=sf8g7sfjdsurtsdieerwqredsgnfg8d`

Dado que todos los demás parámetros eran simples campos de dos y tres caracteres, no es posible comenzar a adivinar combinaciones de aproximadamente 28 caracteres. Un escáner de aplicaciones web necesitará forzar (o adivinar) todo el espacio clave de 30 caracteres. Eso es hasta 30\N28 permutaciones, o trillones de solicitudes HTTP. Eso es un electrón en un pajar digital.

El código de este ejemplo de comprobación de parámetros mágicos puede tener el siguiente aspecto:

```
public void doPost( HttpServletRequest request, HttpServletResponse response) {
  String magic = "sf8g7sfjdsurtsdieerwqredsgnfg8d";
  boolean admin = magic.equals(request.getParameter("magic"));
  if(admin) doAdmin(request, response);
  else …//normal processing
}
```

Al mirar el código, la vulnerabilidad prácticamente salta de la página como un problema potencial.

Ejemplo 2: Mala criptografía

La criptografía se utiliza ampliamente en las aplicaciones web. Imaginemos que un desarrollador decidiera escribir un simple algoritmo de criptografía para que un usuario inicie sesión en el sitio A en el sitio B automáticamente. En su sabiduría, el desarrollador decide que si un usuario se registra en el sitio A, entonces generará una clave usando una función hash MD5 que comprende: `Hash { username : date }`

Cuando un usuario es pasado al sitio B, enviará la clave de la cadena de consulta al sitio B en una redirección HTTP. El sitio B calcula independientemente el

hash, y lo compara con el hash pasado en la solicitud. Si coinciden, el sitio B registra al usuario como el usuario que dice ser.

Como se explica el esquema, las insuficiencias pueden ser resueltas. Cualquiera que descubra el esquema (o se le diga cómo funciona, o descargue la información de Bugtraq) puede entrar como cualquier usuario. Una inspección manual, como una revisión o inspección del código, habría descubierto rápidamente este problema de seguridad. Un escáner de aplicaciones web de caja negra no habría descubierto la vulnerabilidad. Habría visto un hash de 128 bits que cambiaba con cada usuario, y por la naturaleza de las funciones de hash, no cambiaba de ninguna manera predecible.

Una nota sobre las herramientas de revisión de código fuente estático

Muchas organizaciones han comenzado a utilizar escáneres estáticos de código fuente. Si bien es cierto que tienen un lugar en un programa de pruebas completo, es necesario destacar algunas cuestiones fundamentales acerca de por qué este enfoque no es eficaz cuando se utiliza solo. El análisis del código fuente estático por sí solo no puede identificar problemas debido a defectos en el diseño, ya que no puede comprender el contexto en el que se construye el código. Los instrumentos de análisis del código fuente son útiles para determinar los problemas de seguridad debidos a errores de codificación, aunque se requiere un importante esfuerzo manual para validar las conclusiones.

Derivación de los requisitos de las pruebas de seguridad

Para tener un programa de pruebas exitoso, uno debe saber cuáles son los objetivos de las pruebas. Estos objetivos están especificados por los requisitos de seguridad. En esta sección se discute en detalle cómo documentar los requisitos para las pruebas de seguridad derivándolos de las normas y reglamentos aplicables, de los requisitos de aplicación positiva (especificando lo que la aplicación debe hacer) y de los requisitos de aplicación negativa (especificando lo que la aplicación no debe hacer). También se discute cómo los requisitos de seguridad impulsan efectivamente las pruebas de seguridad durante el SDLC y cómo los datos de las pruebas de seguridad pueden ser utilizados para gestionar eficazmente los riesgos de seguridad del software.

Objetivos de las pruebas

Uno de los objetivos de las pruebas de seguridad es validar que los controles de seguridad funcionen como se espera. Esto se documenta mediante `security requirements` que describen la funcionalidad del control de seguridad. A un alto nivel, esto significa probar la confidencialidad, integridad y disponibilidad de los datos así como del servicio. El otro objetivo es validar que los controles

de seguridad se implementan con pocas o ninguna vulnerabilidad. Se trata de vulnerabilidades comunes, como el Top Ten de la OWASP, así como vulnerabilidades que han sido previamente identificadas con evaluaciones de seguridad durante el SDLC, como el modelado de amenazas, el análisis de código fuente, y la prueba de penetración.

Documentación de los requisitos de seguridad

El primer paso en la documentación de los requisitos de seguridad es comprender el `business requirements`. Un documento de requisitos comerciales puede proporcionar información inicial de alto nivel sobre la funcionalidad prevista de la aplicación. Por ejemplo, la finalidad principal de una aplicación puede ser prestar servicios financieros a los clientes o permitir la compra de bienes de un catálogo en línea. Una sección de seguridad de los requisitos comerciales debe destacar la necesidad de proteger los datos del cliente, así como de cumplir con la documentación de seguridad aplicable, como los reglamentos, normas y políticas.

Una lista de verificación general de los reglamentos, normas y políticas aplicables es un buen análisis preliminar del cumplimiento de la seguridad de las aplicaciones web. Por ejemplo, las reglamentaciones de cumplimiento pueden identificarse comprobando la información sobre el sector empresarial y el país o estado en que funcionará la aplicación. Algunas de estas directrices y reglamentos de cumplimiento podrían traducirse en requisitos técnicos específicos para los controles de seguridad. Por ejemplo, en el caso de las aplicaciones financieras, el cumplimiento de las directrices del Consejo Federal de Examen de las Instituciones Financieras para la autenticación requiere que las instituciones financieras apliquen aplicaciones que mitiguen los riesgos de una autenticación deficiente con un control de seguridad de múltiples capas y una autenticación de múltiples factores.

Las normas industriales aplicables en materia de seguridad también deben quedar recogidas en la lista general de verificación de requisitos de seguridad. Por ejemplo, en el caso de las aplicaciones que manejan datos de tarjetas de crédito de clientes, el cumplimiento de la Norma de Seguridad de Datos del Consejo de Normas de Seguridad de la Industria de las Tarjetas de Pago (PCI) prohíbe el almacenamiento de números de identificación personal (PIN) y datos CVV2 y exige que el comerciante proteja los datos de las bandas magnéticas en el almacenamiento y la transmisión con cifrado y en la exhibición mediante enmascaramiento. Estos requisitos de seguridad del DSS del PCI podrían ser validados mediante el análisis del código fuente.

En otra sección de la lista de verificación es necesario hacer cumplir los requisitos generales para el cumplimiento de las normas y políticas de seguridad de la información de la organización. Desde la perspectiva de los requisitos funcionales, los requisitos para el control de la seguridad deben corresponder a una sección específica de las normas de seguridad de la información. Un ejemplo

de ese requisito puede ser: "una complejidad de contraseña de diez caracteres alfanuméricos debe ser impuesta por los controles de autenticación utilizados por la aplicación" Cuando los requisitos de seguridad se asignan a las normas de cumplimiento, una prueba de seguridad puede validar la exposición a los riesgos de cumplimiento. Si se encuentra una violación de las normas y políticas de seguridad de la información, esto dará lugar a un riesgo que puede ser documentado y que la empresa tiene que gestionar. Dado que estos requisitos de cumplimiento de la seguridad son exigibles, deben estar bien documentados y validados con pruebas de seguridad.

Validación de los requisitos de seguridad

Desde la perspectiva de la funcionalidad, la validación de los requisitos de seguridad es el principal objetivo de las pruebas de seguridad. Desde la perspectiva de la gestión de riesgos, la validación de los requisitos de seguridad es el objetivo de las evaluaciones de la seguridad de la información. A un alto nivel, el principal objetivo de las evaluaciones de la seguridad de la información es la identificación de las lagunas en los controles de seguridad, como la falta de controles básicos de autenticación, autorización o cifrado. Si se examina más detenidamente, el objetivo de la evaluación de la seguridad es el análisis de los riesgos, como la determinación de posibles deficiencias en los controles de seguridad que garantizan la confidencialidad, la integridad y la disponibilidad de los datos. Por ejemplo, cuando la aplicación se ocupa de información de identificación personal y datos sensibles, el requisito de seguridad que debe validarse es el cumplimiento de la política de seguridad de la información de la empresa que exige la codificación de esos datos en tránsito y en almacenamiento. Suponiendo que la encriptación se utilice para proteger los datos, los algoritmos de encriptación y las longitudes de las claves deben cumplir las normas de encriptación de la organización. Éstas podrían exigir que sólo se utilizaran determinados algoritmos y longitudes de clave. Por ejemplo, un requisito de seguridad que puede ser objeto de pruebas de seguridad es verificar que sólo se utilicen los cifrados permitidos (por ejemplo, SHA-256, RSA, AES) con longitudes de clave mínimas permitidas (por ejemplo, más de 128 bits para el cifrado simétrico y más de 1024 para el asimétrico).

Desde la perspectiva de la evaluación de la seguridad, los requisitos de seguridad pueden ser validados en diferentes fases del SDLC utilizando diferentes artefactos y metodologías de prueba. Por ejemplo, el modelado de amenazas se centra en la identificación de los fallos de seguridad durante el diseño; el análisis y las revisiones de código seguro se centran en la identificación de los problemas de seguridad en el código fuente durante el desarrollo; y las pruebas de penetración se centran en la identificación de las vulnerabilidades de la aplicación durante las pruebas o la validación.

Los problemas de seguridad que se identifican al principio del SDLC se pueden documentar en un plan de pruebas para que se puedan validar más tarde con pruebas de seguridad. Combinando los resultados de diferentes técnicas de

prueba, es posible derivar mejores casos de prueba de seguridad y aumentar el nivel de garantía de los requisitos de seguridad. Por ejemplo, es posible distinguir las verdaderas vulnerabilidades de las no explotables cuando se combinan los resultados de las pruebas de penetración y el análisis del código fuente. Si se considera la prueba de seguridad de una vulnerabilidad de inyección SQL, por ejemplo, una prueba de caja negra podría implicar en primer lugar una exploración de la aplicación para tomar la huella digital de la vulnerabilidad. La primera prueba de una posible vulnerabilidad de inyección SQL que puede validarse es la generación de una excepción SQL. Otra validación de la vulnerabilidad SQL podría consistir en inyectar manualmente vectores de ataque para modificar la gramática de la consulta SQL con el fin de obtener una explotación de la divulgación de información. Esto podría implicar un gran análisis de ensayo y error antes de que se ejecute la consulta maliciosa. Suponiendo que el probador tiene el código fuente, podría aprender directamente del análisis del código fuente cómo construir el vector de ataque SQL que explotará con éxito la vulnerabilidad (por ejemplo, ejecutar una consulta maliciosa devolviendo datos confidenciales a un usuario no autorizado). Esto puede acelerar la validación de la vulnerabilidad de SQL.

Taxonomías de amenazas y contramedidas

A `threat and countermeasure classification` que tiene en cuenta las causas fundamentales de las vulnerabilidades, es el factor crítico para verificar que los controles de seguridad están diseñados, codificados y construidos para mitigar el impacto de la exposición de dichas vulnerabilidades. En el caso de las aplicaciones web, la exposición de los controles de seguridad a las vulnerabilidades comunes, como el Top Ten de OWASP, puede ser un buen punto de partida para derivar los requisitos generales de seguridad. La lista de verificación de la guía de pruebas de OWASP es un recurso útil para guiar a los probadores a través de vulnerabilidades específicas y pruebas de validación.

El objetivo de la categorización de las amenazas y las contramedidas es definir los requisitos de seguridad en función de las amenazas y la causa fundamental de la vulnerabilidad. Una amenaza puede ser categorizada usando STRIDE, un acrónimo de Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, y Elevation of privilege. La causa raíz puede ser categorizada como un defecto de seguridad en el diseño, un error de seguridad en la codificación, o un problema debido a una configuración insegura. Por ejemplo, la causa fundamental de la vulnerabilidad de la autenticación débil podría ser la falta de autenticación mutua cuando los datos cruzan un límite de confianza entre los niveles de cliente y servidor de la aplicación. Un requisito de seguridad que capte la amenaza de no repudio durante un examen del diseño de la arquitectura permite documentar el requisito de la contramedida (por ejemplo, la autenticación mutua) que puede ser validado más adelante con pruebas de seguridad.

También puede utilizarse una categorización de amenazas y contramedidas para

las vulnerabilidades a fin de documentar los requisitos de seguridad para la codificación segura, como las normas de codificación segura. Un ejemplo de un error de codificación común en los controles de autenticación consiste en aplicar una función hash para cifrar una contraseña, sin aplicar una semilla al valor. Desde la perspectiva de la codificación segura, se trata de una vulnerabilidad que afecta a la codificación utilizada para la autenticación con una vulnerabilidad que es la causa fundamental de un error de codificación. Dado que la causa fundamental es una codificación insegura, el requisito de seguridad puede documentarse en normas de codificación segura y validarse mediante exámenes de códigos seguros durante la fase de desarrollo del SDLC.

Pruebas de seguridad y análisis de riesgos

Los requerimientos de seguridad necesitan tomar en consideración la severidad de las vulnerabilidades para apoyar un `risk mitigation strategy`. Suponiendo que la organización mantiene un repositorio de las vulnerabilidades encontradas en las aplicaciones (es decir, una base de conocimientos sobre vulnerabilidades), los problemas de seguridad se pueden notificar por tipo, problema, mitigación, causa principal, y se pueden asignar a las aplicaciones donde se encuentran. Esa base de conocimientos sobre vulnerabilidades también puede utilizarse para establecer una métrica que permita analizar la eficacia de las pruebas de seguridad en todo el SDLC.

Por ejemplo, considere un problema de validación de entrada, como una inyección SQL, que se haya identificado mediante un análisis del código fuente y se haya notificado con una causa raíz de error de codificación y un tipo de vulnerabilidad de validación de entrada. La exposición de esa vulnerabilidad puede evaluarse mediante una prueba de penetración, probando los campos de entrada con varios vectores de ataque de inyección SQL. Esta prueba podría validar que los caracteres especiales se filtran antes de llegar a la base de datos y mitigar la vulnerabilidad. Combinando los resultados del análisis del código fuente y la prueba de penetración, es posible determinar la probabilidad y la exposición de la vulnerabilidad y calcular la clasificación del riesgo de la vulnerabilidad. Al informar sobre las clasificaciones de riesgo de la vulnerabilidad en los resultados (por ejemplo, el informe de la prueba) es posible decidir la estrategia de mitigación. Por ejemplo, se puede dar prioridad a las vulnerabilidades de riesgo alto y medio para su corrección, mientras que las vulnerabilidades de bajo riesgo pueden fijarse en posteriores publicaciones.

Al considerar los escenarios de amenaza de la explotación de las vulnerabilidades comunes, es posible identificar los posibles riesgos para los que el control de seguridad de la aplicación debe ser sometido a pruebas de seguridad. Por ejemplo, las diez principales vulnerabilidades de OWASP pueden ser mapeadas a ataques como el phishing, violaciones de la privacidad, robo de identidad, compromiso del sistema, alteración o destrucción de datos, pérdida financiera y pérdida de reputación. Estos problemas deben ser documentados como parte

de los escenarios de amenaza. Pensando en términos de amenazas y vulnerabilidades, es posible concebir una batería de pruebas que simulen tales escenarios de ataque. Lo ideal sería que la base de conocimientos sobre vulnerabilidades de la organización se utilizara para derivar casos de prueba basados en el riesgo de seguridad para validar los escenarios de ataque más probables. Por ejemplo, si el robo de identidad se considera de alto riesgo, los escenarios de prueba negativos deberían validar la mitigación de los impactos derivados del aprovechamiento de las vulnerabilidades en la autenticación, los controles criptográficos, la validación de entradas y los controles de autorización.

Derivación de los requisitos de las pruebas funcionales y no funcionales

Requisitos de seguridad funcional Desde la perspectiva de los requisitos de seguridad funcional, las normas, políticas y reglamentos aplicables impulsan tanto la necesidad de un tipo de control de seguridad como la funcionalidad del control. Estos requisitos también se denominan "requisitos positivos", ya que establecen la funcionalidad esperada que puede ser validada a través de pruebas de seguridad. Ejemplos de requisitos positivos son: "la aplicación bloqueará al usuario después de seis intentos fallidos de inicio de sesión" o "las contraseñas deben tener un mínimo de diez caracteres alfanuméricos". La validación de los requisitos positivos consiste en afirmar la funcionalidad esperada y puede probarse volviendo a crear las condiciones de prueba y ejecutando la prueba de acuerdo con entradas predefinidas. Los resultados se muestran entonces como una condición de fallo o de aprobación.

Para validar los requisitos de seguridad con pruebas de seguridad, es necesario que los requisitos de seguridad estén basados en la funcionalidad. Necesitan resaltar la funcionalidad esperada (el qué) e implicar la implementación (el cómo). Ejemplos de requisitos de diseño de seguridad de alto nivel para la autenticación pueden ser:

- Proteger las credenciales de los usuarios y los secretos compartidos en tránsito y en almacenamiento.
- Enmascarar cualquier dato confidencial que se muestre (por ejemplo, contraseñas, cuentas).
- Bloquear la cuenta del usuario después de un cierto número de intentos fallidos de inicio de sesión.
- No mostrar al usuario errores de validación específicos como resultado de un fallo en el inicio de sesión.
- Sólo permita contraseñas que sean alfanuméricas, que incluyan caracteres especiales y que tengan un mínimo de diez caracteres de longitud, para limitar la superficie de ataque.
- Permitir la funcionalidad de cambio de contraseña sólo a los usuarios autentificados mediante la validación de la contraseña antigua, la nueva con-

traseña y la respuesta del usuario a la pregunta de desafío, para evitar el forzamiento bruto de una contraseña mediante el cambio de la misma.
- El formulario de restablecimiento de la contraseña debería validar el nombre de usuario y el correo electrónico registrado del usuario antes de enviar la contraseña temporal al usuario por correo electrónico. La contraseña temporal emitida debería ser una contraseña de una sola vez. Se enviará al usuario un enlace a la página web de restablecimiento de la contraseña. La página web de restablecimiento de la contraseña debería validar la contraseña temporal del usuario, la nueva contraseña, así como la respuesta del usuario a la pregunta de desafío.

Requisitos de seguridad basados en el riesgo Las pruebas de seguridad también deben ser impulsadas por el riesgo. Necesitan validar la aplicación para un comportamiento inesperado, o requisitos negativos.

Ejemplos de requisitos negativos son:

- La aplicación no debe permitir que los datos sean alterados o destruidos.
- La aplicación no debería comprometerse ni utilizarse indebidamente para transacciones financieras no autorizadas por parte de un usuario malintencionado.

Los requisitos negativos son más difíciles de probar, porque no hay un comportamiento esperado que buscar. La búsqueda de un comportamiento esperado que se ajuste a los requisitos anteriores puede requerir que un analista de amenazas proponga de forma poco realista condiciones, causas y efectos de entrada imprevisibles. Por lo tanto, las pruebas de seguridad deben ser impulsadas por el análisis de riesgos y el modelado de amenazas. La clave es documentar los escenarios de amenaza, y la funcionalidad de la contramedida como un factor para mitigar una amenaza.

Por ejemplo, en el caso de los controles de autenticación, los siguientes requisitos de seguridad pueden documentarse desde la perspectiva de las amenazas y las contramedidas:

- Cifrar los datos de autenticación en el almacenamiento y tránsito para mitigar el riesgo de revelación de información y ataques al protocolo de autenticación.
- Cifrar las contraseñas utilizando un cifrado no reversible, por ejemplo, utilizando un resumen (por ejemplo, HASH) y una semilla para evitar los ataques de diccionario.
- Bloquear las cuentas después de alcanzar un umbral de fallo de inicio de sesión y hacer cumplir la complejidad de las contraseñas para mitigar el riesgo de ataques con contraseñas de fuerza bruta.
- Mostrar mensajes de error genéricos al validar las credenciales para mitigar el riesgo de cosecha o enumeración de cuentas.

- Autentificar mutuamente el cliente y el servidor para prevenir el no repudio y los ataques de Man In the Middle (MiTM).

Las herramientas de modelización de amenazas, como los árboles de amenazas y las bibliotecas de ataques, pueden ser útiles para derivar los escenarios de prueba negativos. Un árbol de amenazas supondrá un ataque de raíz (por ejemplo, el atacante podría ser capaz de leer los mensajes de otros usuarios) e identificará diferentes exploits de los controles de seguridad (por ejemplo, la validación de datos falla debido a una vulnerabilidad de inyección SQL) y las contramedidas necesarias (por ejemplo, aplicar la validación de datos y las consultas parametrizadas) que podrían ser validadas para ser eficaces en la mitigación de esos ataques.

Derivación de los requisitos de las pruebas de seguridad mediante el uso y los casos de uso indebido

Un prerrequisito para describir la funcionalidad de la aplicación es entender qué se supone que debe hacer la aplicación y cómo. Esto puede hacerse describiendo casos de uso. Los casos de uso, en la forma gráfica como se usa comúnmente en la ingeniería de software, muestran las interacciones de los actores y sus relaciones. Ayudan a identificar los actores de la aplicación, sus relaciones, la secuencia de acciones prevista para cada escenario, las acciones alternativas, los requisitos especiales, las condiciones previas y las condiciones posteriores.

De manera similar a los casos de uso, los casos de uso indebido y abuso describen los escenarios de uso no previsto y malicioso de la aplicación. Estos casos de uso indebido proporcionan una forma de describir escenarios de cómo un atacante podría hacer un uso indebido y abusivo de la aplicación. Al repasar los pasos individuales de un escenario de uso y pensar en cómo puede ser explotado maliciosamente, se pueden descubrir posibles fallos o aspectos de la aplicación que no están bien definidos. La clave es describir todos los escenarios de uso y abuso posibles o, al menos, los más críticos.

Los escenarios de uso indebido permiten analizar la aplicación desde el punto de vista del atacante y contribuyen a identificar las posibles vulnerabilidades y las contramedidas que deben aplicarse para mitigar el impacto causado por la posible exposición a dichas vulnerabilidades. Dados todos los casos de uso y abuso, es importante analizarlos para determinar cuáles son los más críticos y deben ser documentados en los requisitos de seguridad. La identificación de los casos más críticos de uso indebido y abuso impulsa la documentación de los requisitos de seguridad y los controles necesarios en los que deben mitigarse los riesgos de seguridad.

Para derivar los requisitos de seguridad tanto de los casos de uso como de los de abuso, es importante definir los escenarios funcionales y los escenarios negativos y ponerlos en forma gráfica. El siguiente ejemplo es una metodología paso a paso para el caso de derivar requisitos de seguridad para la autenticación.

Paso 1: Describir el escenario funcional El usuario se autentica suministrando un nombre de usuario y una contraseña. La aplicación concede el acceso a los usuarios basándose en la autenticación de las credenciales del usuario por parte de la aplicación y proporciona errores específicos al usuario cuando la validación falla.

Paso 2: Describir el escenario negativo El atacante rompe la autenticación mediante un ataque de fuerza bruta o de diccionario de contraseñas y vulnerabilidades de recolección de cuentas en la aplicación. Los errores de validación proporcionan al atacante información específica que se utiliza para adivinar qué cuentas son cuentas registradas válidas (nombres de usuario). A continuación, el atacante intenta forzar la contraseña de una cuenta válida. Un ataque de fuerza bruta sobre contraseñas con una longitud mínima de cuatro dígitos puede tener éxito con un número limitado de intentos (es decir, 10^4).

Paso 3: Describir los escenarios funcionales y negativos con el caso de uso y mal uso El ejemplo gráfico que figura a continuación muestra la derivación de los requisitos de seguridad mediante casos de uso y abuso. El escenario funcional consiste en las acciones del usuario (introduciendo un nombre de usuario y una contraseña) y las acciones de la aplicación (autenticando al usuario y proporcionando un mensaje de error si la validación falla). El caso de uso indebido consiste en las acciones del atacante, es decir, tratar de romper la autenticación forzando brutalmente la contraseña mediante un ataque de diccionario y adivinando los nombres de usuario válidos a partir de los mensajes de error. Al representar gráficamente las amenazas a las acciones del usuario (usos indebidos), es posible derivar las contramedidas como las acciones de la aplicación que mitigan esas amenazas.

Figura 2-5: Caso de uso y mal uso

Paso 4: Obtener los requisitos de seguridad En este caso, se derivan los siguientes requisitos de seguridad para la autenticación:

1. Los requisitos de las contraseñas deben estar alineados con los estándares actuales para una suficiente complejidad.
2. Las cuentas deben ser bloqueadas después de cinco intentos fallidos de ingreso.
3. Los mensajes de error de inicio de sesión deben ser genéricos.

Estos requisitos de seguridad deben ser documentados y probados.

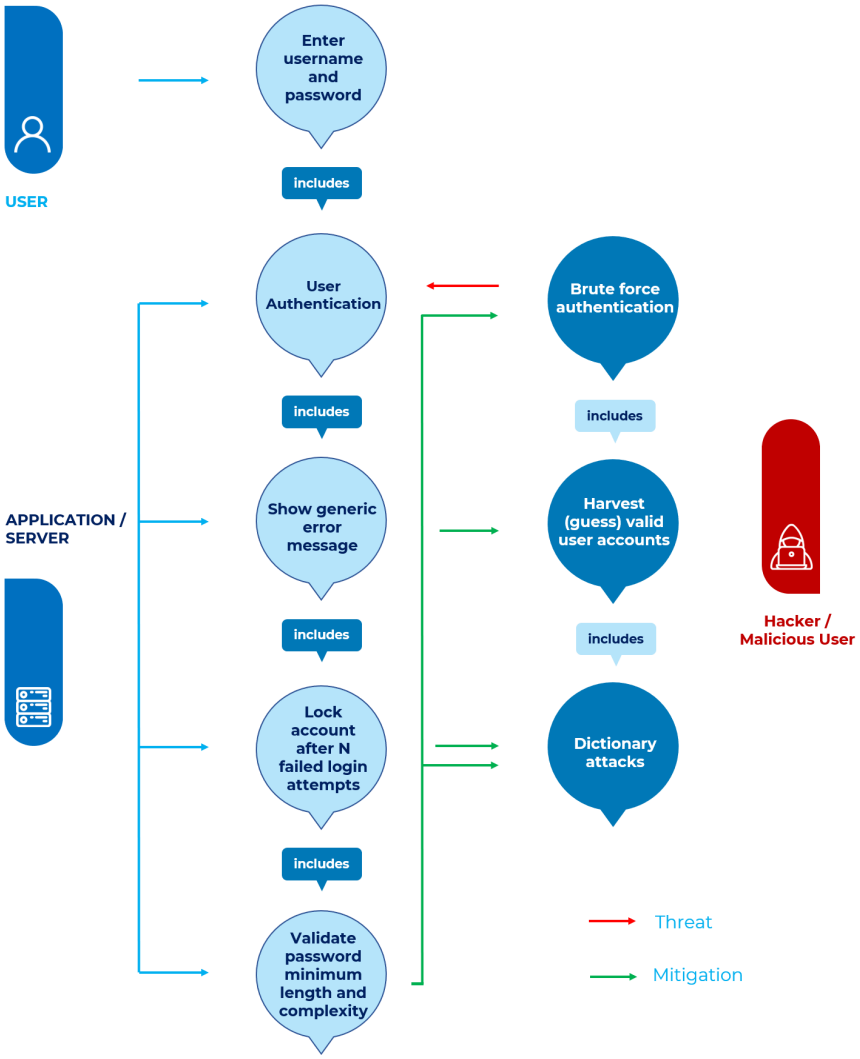

Figure 4: Use and Misuse case

Pruebas de seguridad integradas en los flujos de trabajo de desarrollo y pruebas

Pruebas de seguridad en el flujo de trabajo de desarrollo

Las pruebas de seguridad durante la fase de desarrollo del SDLC representan la primera oportunidad para que los desarrolladores se aseguren de que los componentes individuales de software que han desarrollado se someten a pruebas de seguridad antes de que se integren con otros componentes y se incorporen a la aplicación. Los componentes de software pueden consistir en artefactos de software como funciones, métodos y clases, así como interfaces de programación de aplicaciones, bibliotecas y archivos ejecutables. Para las pruebas de seguridad, los desarrolladores pueden confiar en los resultados del análisis del código fuente para verificar estáticamente que el código fuente desarrollado no incluya posibles vulnerabilidades y cumpla con las normas de codificación segura. Las pruebas de la unidad de seguridad pueden verificar además de forma dinámica (es decir, en tiempo de ejecución) que los componentes funcionan como se espera. Antes de integrar los cambios de código nuevos y existentes en la construcción de la aplicación, se deben revisar y validar los resultados del análisis estático y dinámico.

La validación del código fuente antes de la integración en las construcciones de la aplicación suele ser responsabilidad del desarrollador principal. Los desarrolladores principales son también los expertos en seguridad de los programas informáticos y su función es dirigir el examen del código seguro. Deben tomar decisiones sobre si aceptar el código que se va a publicar en la construcción de la aplicación, o exigir más cambios y pruebas. Este flujo de trabajo de revisión de código seguro puede hacerse cumplir a través de la aceptación formal, así como de una verificación en una herramienta de gestión de flujo de trabajo. Por ejemplo, suponiendo el típico flujo de trabajo de gestión de defectos utilizado para los fallos funcionales, los fallos de seguridad que han sido corregidos por un desarrollador pueden ser notificados en un sistema de gestión de defectos o de cambios. El maestro de construcción puede entonces examinar los resultados de las pruebas comunicados por los desarrolladores en la herramienta, y conceder aprobaciones para la verificación de los cambios de código en la construcción de la aplicación.

Prueba de seguridad en el flujo de trabajo de la prueba

Después de que los desarrolladores prueben los componentes y los cambios de código y los verifiquen en la construcción de la aplicación, el siguiente paso más probable en el flujo de trabajo del proceso de desarrollo de software es realizar pruebas en la aplicación como una entidad completa. Este nivel de pruebas suele denominarse prueba integrada y prueba a nivel de sistema. Cuando las pruebas de seguridad forman parte de estas actividades de prueba, pueden utilizarse para

validar tanto la funcionalidad de seguridad de la aplicación en su conjunto, como la exposición a las vulnerabilidades a nivel de la aplicación. Estas pruebas de seguridad de la aplicación incluyen tanto pruebas de caja blanca, como el análisis del código fuente, como pruebas de caja negra, como las pruebas de penetración. Las pruebas también pueden incluir pruebas de caja gris, en las que se asume que el probador tiene algún conocimiento parcial sobre la aplicación. Por ejemplo, con algún conocimiento sobre la gestión de sesiones de la aplicación, el probador puede entender mejor si las funciones de cierre de sesión y de tiempo de espera están debidamente aseguradas.

El objetivo de las pruebas de seguridad es el sistema completo que es vulnerable a los ataques. Durante esta fase, es posible que los probadores de seguridad determinen si las vulnerabilidades pueden ser explotadas. Entre ellas se incluyen las vulnerabilidades comunes de las aplicaciones web, así como los problemas de seguridad que se han identificado anteriormente en el SDLC con otras actividades como el modelado de amenazas, el análisis del código fuente y las revisiones de código seguro.

Normalmente, los ingenieros de pruebas, en lugar de los desarrolladores de software, realizan pruebas de seguridad cuando la aplicación está en el ámbito de las pruebas del sistema de integración. Los ingenieros de pruebas tienen conocimientos de seguridad sobre las vulnerabilidades de las aplicaciones web, las técnicas de pruebas de caja negra y caja blanca, y son los dueños de la validación de los requisitos de seguridad en esta fase. Para poder realizar pruebas de seguridad, es un requisito previo que los casos de pruebas de seguridad se documenten en las directrices y procedimientos de pruebas de seguridad.

Un ingeniero de pruebas que valide la seguridad de la aplicación en el entorno del sistema integrado podría liberar la aplicación para su prueba en el entorno operacional (por ejemplo, pruebas de aceptación del usuario). En esta etapa del SDLC (es decir, la validación), las pruebas funcionales de la aplicación suelen ser responsabilidad de los encargados de las pruebas de control de calidad, mientras que los hackers de sombrero blanco o los consultores de seguridad suelen ser responsables de las pruebas de seguridad. Algunas organizaciones confían en su propio equipo especializado de hacking ético para llevar a cabo dichas pruebas cuando no se requiere una evaluación de un tercero (por ejemplo, para fines de auditoría).

Dado que estas pruebas pueden ser a veces la última línea de defensa para arreglar las vulnerabilidades antes de que la aplicación sea lanzada a producción, es importante que los problemas se aborden según las recomendaciones del equipo de pruebas. Las recomendaciones pueden incluir cambios en el código, el diseño o la configuración. En este nivel, los auditores de seguridad y los oficiales de seguridad de la información discuten los problemas de seguridad reportados y analizan los riesgos potenciales de acuerdo con los procedimientos de gestión de riesgos de la información. Tales procedimientos pueden requerir que el equipo de desarrollo arregle todas las vulnerabilidades de alto riesgo antes de que la aplicación pueda ser desplegada, a menos que tales riesgos sean reconocidos y

aceptados.

Pruebas de seguridad del desarrollador

Pruebas de seguridad en la fase de codificación: Pruebas unitarias Desde la perspectiva del desarrollador, el principal objetivo de las pruebas de seguridad es validar que el código se está desarrollando de acuerdo con los requisitos de las normas de codificación segura. Los propios artefactos de codificación de los desarrolladores (como funciones, métodos, clases, API y bibliotecas) deben ser validados funcionalmente antes de ser integrados en la construcción de la aplicación.

Los requisitos de seguridad que deben cumplir los desarrolladores deben documentarse en normas de codificación segura y validarse con análisis estáticos y dinámicos. Si la actividad de prueba unitaria sigue una revisión de código seguro, las pruebas unitarias pueden validar que los cambios de código requeridos por las revisiones de código seguro se implementen correctamente. Tanto las revisiones de código seguro como el análisis del código fuente mediante instrumentos de análisis del código fuente pueden ayudar a los desarrolladores a identificar los problemas de seguridad del código fuente a medida que se desarrolla. Mediante las pruebas de unidad y el análisis dinámico (por ejemplo, la depuración) los desarrolladores pueden validar la funcionalidad de seguridad de los componentes, así como verificar que las contramedidas que se están desarrollando mitigan cualquier riesgo de seguridad identificado previamente mediante el modelado de amenazas y el análisis del código fuente.

Una buena práctica para los desarrolladores consiste en elaborar casos de prueba de seguridad como un conjunto genérico de pruebas de seguridad que forme parte del marco de pruebas unitarias existente. Un conjunto de pruebas de seguridad genéricas podría derivarse de casos de uso y abuso previamente definidos a funciones, métodos y clases de pruebas de seguridad. Un conjunto de pruebas de seguridad genérico podría incluir casos de pruebas de seguridad para validar los requisitos tanto positivos como negativos de los controles de seguridad, como por ejemplo

- Identidad, autenticación y control de acceso
- Validación y codificación de la entrada
- Cifrado
- Gestión de usuarios y sesiones
- Manejo de errores y excepciones
- Auditoría y registro

Los desarrolladores dotados de una herramienta de análisis de código fuente integrada en su IDE, de estándares de codificación segura y de un marco de pruebas de unidad de seguridad pueden evaluar y verificar la seguridad de los componentes de software que se están desarrollando. Pueden ejecutarse casos

de prueba de seguridad para identificar posibles problemas de seguridad que tengan causas fundamentales en el código fuente: además de la validación de entrada y salida de los parámetros que entran y salen de los componentes, estos problemas incluyen las comprobaciones de autenticación y autorización realizadas por el componente, la protección de los datos dentro del componente, el manejo seguro de excepciones y errores, y la auditoría y el registro seguros. Los marcos de pruebas de la unidad, como JUnit, NUnit y CUnit, pueden adaptarse para verificar los requisitos de las pruebas de seguridad. En el caso de las pruebas funcionales de seguridad, las pruebas de nivel unitario pueden probar la funcionalidad de los controles de seguridad a nivel de componente de software, como funciones, métodos o clases. Por ejemplo, un caso de prueba podría validar la validación de entrada y salida (por ejemplo, el saneamiento de variables) y las comprobaciones de límites de las variables afirmando la funcionalidad esperada del componente.

Los escenarios de amenaza identificados con los casos de uso y abuso pueden utilizarse para documentar los procedimientos de prueba de los componentes de software. En el caso de los componentes de autenticación, por ejemplo, las pruebas de la unidad de seguridad pueden afirmar la funcionalidad de establecer un bloqueo de cuenta, así como el hecho de que no se puede abusar de los parámetros de entrada del usuario para eludir el bloqueo de la cuenta (por ejemplo, estableciendo el contador de bloqueo de cuenta en un número negativo).

A nivel de los componentes, las pruebas de la unidad de seguridad pueden validar tanto las afirmaciones positivas como las negativas, como los errores y el manejo de excepciones. Las excepciones deben detectarse sin dejar el sistema en un estado de inseguridad, como la posible denegación de servicio causada por la no reasignación de recursos (por ejemplo, las manijas de conexión no cerradas dentro de un bloque de declaración final), así como la posible elevación de los privilegios (por ejemplo, mayores privilegios adquiridos antes de que se lance la excepción y no volver al nivel anterior antes de salir de la función). El tratamiento seguro de los errores puede validar la posible revelación de información mediante mensajes de error informativos y rastros de pila.

Los casos de prueba de seguridad a nivel de unidad pueden ser desarrollados por un ingeniero de seguridad que es el experto en la materia de la seguridad de los programas informáticos y también es responsable de validar que los problemas de seguridad en el código fuente han sido corregidos y pueden ser comprobados en la construcción del sistema integrado. Por lo general, el administrador de las construcciones de la aplicación también se asegura de que las bibliotecas y los archivos ejecutables de terceros sean evaluados desde el punto de vista de la seguridad para detectar posibles vulnerabilidades antes de ser integrados en la construcción de la aplicación.

Los escenarios de amenazas para vulnerabilidades comunes que tienen causas de raíz en la codificación insegura también pueden documentarse en la guía de pruebas de seguridad del desarrollador. Cuando se implementa una corrección de un defecto de codificación identificado con el análisis del código fuente, por

ejemplo, los casos de prueba de seguridad pueden verificar que la implementación del cambio de código sigue los requisitos de codificación segura documentados en las normas de codificación segura.

El análisis del código fuente y las pruebas de unidad pueden validar que el cambio de código mitiga la vulnerabilidad expuesta por el defecto de codificación previamente identificado. Los resultados del análisis automatizado del código seguro también pueden utilizarse como puertas de registro automático para el control de versiones; por ejemplo, los artefactos de software no se pueden registrar en la construcción con problemas de codificación de gravedad alta o media.

Pruebas de seguridad de los probadores funcionales

Pruebas de seguridad durante la fase de integración y validación: Pruebas del sistema integrado y pruebas de funcionamiento El objetivo principal de las pruebas de sistemas integrados es validar el concepto de "defensa en profundidad", es decir, que la aplicación de controles de seguridad proporciona seguridad en diferentes capas. Por ejemplo, la falta de validación de entrada cuando se llama a un componente integrado con la aplicación es a menudo un factor que puede ser probado con las pruebas de integración.

El entorno de pruebas del sistema de integración es también el primer entorno en el que los encargados de las pruebas pueden simular escenarios de ataque reales, como los que puede ejecutar potencialmente un usuario externo o interno malintencionado de la aplicación. Las pruebas de seguridad a este nivel pueden validar si las vulnerabilidades son reales y pueden ser explotadas por los atacantes. Por ejemplo, una posible vulnerabilidad encontrada en el código fuente puede calificarse de alto riesgo debido a la exposición a posibles usuarios malintencionados, así como por su posible impacto (por ejemplo, el acceso a información confidencial).

Los escenarios de ataques reales pueden ser probados tanto con técnicas de pruebas manuales como con herramientas de pruebas de penetración. Las pruebas de seguridad de este tipo también se conocen como pruebas de hacking ético. Desde el punto de vista de las pruebas de seguridad, son pruebas basadas en el riesgo y tienen el objetivo de probar la aplicación en el entorno operativo. El objetivo es la construcción de la aplicación que es representativa de la versión de la aplicación que se está desplegando en la producción.

Incluir las pruebas de seguridad en la fase de integración y validación es fundamental para identificar las vulnerabilidades debidas a la integración de los componentes, así como para validar la exposición de dichas vulnerabilidades. Las pruebas de seguridad de las aplicaciones requieren un conjunto especializado de habilidades, que incluyen tanto el conocimiento del software como de la seguridad, que no son típicas de los ingenieros de seguridad. En consecuencia, a menudo se exige a las organizaciones que capaciten a sus desarrolladores de

software en técnicas de piratería ética y en procedimientos y herramientas de evaluación de la seguridad. Un escenario realista es desarrollar esos recursos internamente y documentarlos en guías y procedimientos de pruebas de seguridad que tengan en cuenta los conocimientos de pruebas de seguridad del desarrollador. Por ejemplo, una denominada "lista de tramposos o lista de verificación de casos de pruebas de seguridad" puede proporcionar casos de prueba sencillos y vectores de ataque que pueden ser utilizados por los encargados de las pruebas para validar la exposición a vulnerabilidades comunes como la suplantación de identidad, la divulgación de información, el desbordamiento de búferes, las cadenas de formato, la inyección SQL y la inyección XSS, XML, SOAP, los problemas de canonización, la denegación de servicio y el código gestionado y los controles ActiveX (por ejemplo, .NET). Una primera batería de estas pruebas puede realizarse manualmente con un conocimiento muy básico de la seguridad del software.

El primer objetivo de las pruebas de seguridad podría ser la validación de un conjunto de requisitos mínimos de seguridad. Estos casos de pruebas de seguridad podrían consistir en forzar manualmente la aplicación a estados de error y excepcionales y reunir conocimientos del comportamiento de la aplicación. Por ejemplo, las vulnerabilidades de inyección SQL pueden probarse manualmente inyectando vectores de ataque a través de la entrada del usuario, y comprobando si las excepciones de SQL se devuelven al usuario. La prueba de un error de excepción SQL puede ser una manifestación de una vulnerabilidad que puede ser explotada.

Una prueba de seguridad más profunda podría requerir el conocimiento del probador de técnicas y herramientas de prueba especializadas. Además del análisis del código fuente y las pruebas de penetración, estas técnicas incluyen, por ejemplo: inyección de código fuente y fallos binarios, análisis de propagación de fallos y cobertura de código, pruebas de detección de pelusas e ingeniería inversa. La guía de pruebas de seguridad debería proporcionar procedimientos y recomendar herramientas que puedan ser utilizadas por los encargados de las pruebas de seguridad para llevar a cabo esas evaluaciones de seguridad en profundidad.

El siguiente nivel de pruebas de seguridad después de las pruebas del sistema de integración es realizar pruebas de seguridad en el entorno de aceptación del usuario. Existen ventajas únicas al realizar pruebas de seguridad en el entorno operativo. El entorno de pruebas de aceptación del usuario (UAT) es el más representativo de la configuración de la versión, con la excepción de los datos (por ejemplo, los datos de las pruebas se utilizan en lugar de los datos reales). Una característica de las pruebas de seguridad en la UAT es la comprobación de los problemas de configuración de seguridad. En algunos casos estas vulnerabilidades pueden representar riesgos elevados. Por ejemplo, el servidor que aloja la aplicación web podría no estar configurado con los privilegios mínimos, un certificado SSL válido y una configuración segura, los servicios esenciales desactivados y el directorio raíz de la web limpio de páginas web de prueba y

de administración.

Análisis de datos de pruebas de seguridad y presentación de informes

Objetivos de las métricas y mediciones de las pruebas de seguridad

Definir los objetivos de las métricas y mediciones de las pruebas de seguridad es un requisito previo para utilizar los datos de las pruebas de seguridad en los procesos de análisis y gestión de riesgos. Por ejemplo, una medición, como el número total de vulnerabilidades encontradas con las pruebas de seguridad, podría cuantificar la postura de seguridad de la aplicación. Estas mediciones también ayudan a identificar los objetivos de seguridad para las pruebas de seguridad del software, por ejemplo, reduciendo el número de vulnerabilidades a un número mínimo aceptable antes de que la aplicación se despliegue en la producción.

Otro objetivo manejable podría ser comparar la postura de seguridad de la aplicación con una línea de base para evaluar las mejoras en los procesos de seguridad de la aplicación. Por ejemplo, la línea de base de las métricas de seguridad podría consistir en una aplicación que se probara sólo con pruebas de penetración. Los datos de seguridad obtenidos de una aplicación que también fue probada en cuanto a seguridad durante la codificación deberían mostrar una mejora (por ejemplo, un menor número de vulnerabilidades) en comparación con la línea de base.

En las pruebas de software tradicionales, el número de defectos del software, como los errores encontrados en una aplicación, podría proporcionar una medida de la calidad del software. Del mismo modo, las pruebas de seguridad pueden proporcionar una medida de la seguridad del software. Desde la perspectiva de la gestión de los defectos y la presentación de informes, la calidad del software y las pruebas de seguridad pueden utilizar categorizaciones similares para las causas fundamentales y los esfuerzos de corrección de defectos. Desde la perspectiva de las causas profundas, un defecto de seguridad puede deberse a un error de diseño (por ejemplo, defectos de seguridad) o a un error de codificación (por ejemplo, un error de seguridad). Desde la perspectiva del esfuerzo necesario para corregir un defecto, tanto los defectos de seguridad como los de calidad pueden medirse en términos de horas de desarrollo para llevar a cabo la corrección, los instrumentos y recursos necesarios y el costo de llevar a cabo la corrección.

Una característica de los datos de las pruebas de seguridad, en comparación con los datos de calidad, es la categorización en términos de la amenaza, la exposición de la vulnerabilidad y el posible impacto que ésta plantea para determinar el riesgo. Las pruebas de seguridad de las aplicaciones consisten en gestionar los riesgos técnicos para asegurarse de que las contramedidas de la aplicación

alcanzan niveles aceptables. Por esta razón, los datos de las pruebas de seguridad deben apoyar la estrategia de riesgos de seguridad en los puntos de control críticos durante el SDLC. Por ejemplo, las vulnerabilidades encontradas en el código fuente con el análisis del código fuente representan una medida inicial de riesgo. Se puede calcular una medida del riesgo (por ejemplo, alto, medio, bajo) de la vulnerabilidad determinando los factores de exposición y probabilidad, y validando la vulnerabilidad con pruebas de penetración. Las medidas de riesgo asociadas a las vulnerabilidades encontradas con las pruebas de seguridad permiten a la dirección de la empresa tomar decisiones de gestión de riesgos, como decidir si los riesgos pueden aceptarse, mitigarse o transferirse a diferentes niveles dentro de la organización (por ejemplo, tanto los riesgos comerciales como los técnicos).

Al evaluar la postura de seguridad de una aplicación, es importante tener en cuenta ciertos factores, como el tamaño de la aplicación que se está desarrollando. Se ha demostrado estadísticamente que el tamaño de la aplicación está relacionado con el número de problemas encontrados en la aplicación durante las pruebas. Dado que las pruebas reducen los problemas, es lógico que las aplicaciones de mayor tamaño se prueben más a menudo que las de menor tamaño.

Cuando se realizan pruebas de seguridad en varias fases del SDLC, los datos de las pruebas pueden demostrar la capacidad de las pruebas de seguridad para detectar vulnerabilidades tan pronto como se introducen. Los datos de las pruebas también pueden probar la eficacia de la eliminación de las vulnerabilidades mediante la aplicación de contramedidas en diferentes puntos de control del SDLC. Una medida de este tipo también se define como "métrica de contención" y proporciona una medida de la capacidad de una evaluación de seguridad realizada en cada fase del proceso de desarrollo para mantener la seguridad dentro de cada fase. Estas métricas de contención también son un factor crítico para reducir el costo de la reparación de las vulnerabilidades. Es menos costoso tratar las vulnerabilidades en la misma fase del SDLC en que se encuentran, en lugar de arreglarlas más tarde en otra fase.

Las métricas de pruebas de seguridad pueden apoyar el análisis de riesgos de seguridad, de costos y de gestión de defectos cuando se asocian con objetivos tangibles y temporales como:

- Reduciendo el número total de vulnerabilidades en un 30%.
- Arreglar los problemas de seguridad en un plazo determinado (por ejemplo, antes del lanzamiento de la versión beta).

Los datos de las pruebas de seguridad pueden ser absolutos, como el número de vulnerabilidades detectadas durante la revisión manual del código, así como comparativos, como el número de vulnerabilidades detectadas en las revisiones de código comparadas con las pruebas de penetración. Para responder a las preguntas sobre la calidad del proceso de seguridad, es importante determinar una línea de base de lo que podría considerarse aceptable y bueno.

Los datos de las pruebas de seguridad también pueden servir de apoyo a objetivos específicos del análisis de seguridad. Esos objetivos podrían ser el cumplimiento de los reglamentos de seguridad y las normas de seguridad de la información, la gestión de los procesos de seguridad, la identificación de las causas fundamentales de la seguridad y las mejoras de los procesos, y el análisis de la relación costo-beneficio de la seguridad.

Cuando se comunican los datos de las pruebas de seguridad, éstos tienen que proporcionar métricas para apoyar el análisis. El alcance del análisis es la interpretación de los datos de las pruebas para encontrar pistas sobre la seguridad del software que se está produciendo, así como la eficacia del proceso.

Algunos ejemplos de indicios respaldados por los datos de las pruebas de seguridad pueden ser:

- ¿Se reducen las vulnerabilidades a un nivel aceptable para la liberación?
- ¿Cómo se compara la calidad de seguridad de este producto con la de otros productos de software similares?
- ¿Se cumplen todos los requisitos de las pruebas de seguridad?
- ¿Cuáles son las principales causas de los problemas de seguridad?
- ¿Cuán numerosas son las fallas de seguridad comparadas con los errores de seguridad?
- ¿Qué actividad de seguridad es más eficaz para encontrar vulnerabilidades?
- ¿Qué equipo es más productivo para arreglar los defectos de seguridad y las vulnerabilidades?
- ¿Qué porcentaje de las vulnerabilidades generales son de alto riesgo?
- ¿Qué herramientas son más eficaces para detectar las vulnerabilidades de seguridad?
- ¿Qué tipo de pruebas de seguridad son más efectivas para encontrar vulnerabilidades (por ejemplo, pruebas de caja blanca vs. caja negra)?
- ¿Cuántos problemas de seguridad se encuentran durante las revisiones de código seguro?
- ¿Cuántos problemas de seguridad se encuentran durante los exámenes de diseño de seguridad?

Para poder emitir un juicio sensato utilizando los datos de las pruebas, es importante tener una buena comprensión del proceso de pruebas, así como de las herramientas de prueba. Se debe adoptar una taxonomía de herramientas para decidir qué herramientas de seguridad se deben utilizar. Las herramientas de seguridad pueden calificarse como buenas para encontrar vulnerabilidades comunes y conocidas, cuando se trata de artefactos diferentes.

Es importante señalar que no se prueban los problemas de seguridad desconocidos. El hecho de que una prueba de seguridad esté libre de problemas no significa que el software o la aplicación sean buenos.

Incluso las herramientas de automatización más sofisticadas no son adecuadas para un probador de seguridad experimentado. El simple hecho de confiar en los resultados exitosos de las pruebas de las herramientas automatizadas dará a los profesionales de la seguridad una falsa sensación de seguridad. Típicamente, cuanto más experimentados sean los probadores de seguridad con la metodología y las herramientas de prueba de seguridad, mejores serán los resultados de la prueba y el análisis de seguridad. Es importante que los gestores que inviertan en herramientas de pruebas de seguridad también consideren la posibilidad de invertir en la contratación de recursos humanos cualificados, así como en la formación en pruebas de seguridad.

Requisitos de presentación de informes

La postura de seguridad de una aplicación puede caracterizarse desde la perspectiva del efecto, como el número de vulnerabilidades y la clasificación de riesgos de las mismas, así como desde la perspectiva de la causa o el origen, como los errores de codificación, los fallos de arquitectura y los problemas de configuración.

Las vulnerabilidades pueden clasificarse según diferentes criterios. La medida de la severidad de la vulnerabilidad más comúnmente utilizada es el Sistema Común de Puntuación de Vulnerabilidades (CVSS), un estándar mantenido por el Foro de Equipos de Respuesta a Incidentes y Seguridad (FIRST).

Al reportar los datos de las pruebas de seguridad, la mejor práctica es incluir la siguiente información:

- una categorización de cada vulnerabilidad por tipo;
- la amenaza de seguridad a la que cada problema está expuesto;
- la causa principal de cada problema de seguridad, como el fallo o la falla;
- cada técnica de prueba utilizada para encontrar los problemas;
- la reparación, o contramedida, para cada vulnerabilidad; y
- la clasificación de la gravedad de cada vulnerabilidad (por ejemplo, puntuación alta, media, baja o CVSS).

Al describir cuál es la amenaza a la seguridad, será posible comprender si el control de mitigación es ineficaz para mitigar la amenaza y por qué.

Informar sobre la causa de fondo del problema puede ayudar a determinar qué es lo que hay que arreglar. En el caso de las pruebas de caja blanca, por ejemplo, la causa fundamental de la seguridad del software de la vulnerabilidad será el código fuente ofensivo.

Una vez notificados los problemas, también es importante dar orientación al desarrollador del software sobre cómo volver a probar y encontrar la vulnerabilidad. Esto podría implicar el uso de una técnica de prueba de caja blanca (por ejemplo, la revisión del código de seguridad con un analizador de código

estático) para encontrar si el código es vulnerable. Si se puede encontrar una vulnerabilidad mediante una prueba de penetración de caja negra, el informe de la prueba también debe proporcionar información sobre cómo validar la exposición de la vulnerabilidad al front end (por ejemplo, el cliente).

La información sobre cómo corregir la vulnerabilidad debe ser lo suficientemente detallada como para que un desarrollador pueda llevar a cabo una corrección. Debe proporcionar ejemplos de codificación segura, cambios de configuración y proporcionar referencias adecuadas.

Por último, la clasificación de la gravedad contribuye al cálculo de la clasificación de los riesgos y ayuda a priorizar el esfuerzo de reparación. Típicamente, la asignación de una calificación de riesgo a la vulnerabilidad implica un análisis de riesgo externo basado en factores como el impacto y la exposición.

Casos comerciales

Para que las métricas de las pruebas de seguridad sean útiles, deben aportar valor a los interesados en los datos de las pruebas de seguridad de la organización. Entre los interesados pueden figurar directores de proyectos, desarrolladores, oficinas de seguridad de la información, auditores y jefes de información. El valor puede ser en términos del caso comercial que tiene cada interesado en el proyecto, en términos de función y responsabilidad.

Los desarrolladores de software examinan los datos de las pruebas de seguridad para demostrar que el software está codificado de forma segura y eficiente. Esto les permite argumentar a favor de la utilización de herramientas de análisis de código fuente, el seguimiento de estándares de codificación segura y la asistencia a cursos de formación en seguridad de software.

Los directores de proyectos buscan datos que les permitan gestionar y utilizar con éxito las actividades y recursos de las pruebas de seguridad de acuerdo con el plan del proyecto. Para los directores de proyectos, los datos de las pruebas de seguridad pueden mostrar que los proyectos están en el plazo previsto y se mueven en las fechas de entrega previstas, y que mejoran durante las pruebas.

Los datos de las pruebas de seguridad también ayudan al caso empresarial de las pruebas de seguridad si la iniciativa proviene de los oficiales de seguridad de la información (ISO). Por ejemplo, pueden proporcionar pruebas de que las pruebas de seguridad durante el SDLC no afectan a la entrega del proyecto, sino que reducen la carga de trabajo total necesaria para abordar las vulnerabilidades más adelante en la producción.

Para los auditores de cumplimiento, las métricas de las pruebas de seguridad proporcionan un nivel de garantía de seguridad del software y la confianza de que el cumplimiento de las normas de seguridad se aborda a través de los procesos de revisión de la seguridad dentro de la organización.

Por último, los jefes de información (CIO) y los jefes de seguridad de la información (CISO), que son responsables del presupuesto que debe asignarse en recursos de seguridad, buscan la derivación de un análisis de costo-beneficio a partir de los datos de las pruebas de seguridad. Esto les permite tomar decisiones informadas sobre las actividades y herramientas de seguridad en las que invertir. Una de las métricas que respalda ese análisis es el rendimiento de la inversión en seguridad. Para obtener esa métrica a partir de los datos de las pruebas de seguridad, es importante cuantificar el diferencial entre el riesgo, debido a la exposición de las vulnerabilidades, y la eficacia de las pruebas de seguridad para mitigar el riesgo de seguridad, y luego tener en cuenta esta diferencia con el costo de la actividad de las pruebas de seguridad o las herramientas de prueba adoptadas.