Universidad de Buenos Aires

Facultades de Ciencias Económicas,
Cs. Exactas y Naturales e Ingeniería

Maestría en Seguridad Informática

Trabajo Final de Maestría

Tema

Análisis de cumplimiento de la metodología Owasp ante amenazas web en empresas argentinas

Autor:

Carlos Ivan Sulca Galarza

Director de Tesis:

Dr. Juan Pedro Hecht

Año: 2021

Cohorte 2018

DECLARACIÓN JURADA DE ORIGEN DE LOS CONTENIDOS

Por medio de la presente, el autor manifiesta conocer y aceptar el Reglamento de Trabajos Finales vigente y que se hace responsable que la totalidad de los contenidos del presente documento son originales y de su creación exclusiva, o bien pertenecen a terceros u otras fuentes, que han sido adecuadamente referenciados y cuya inclusión no infringe la legislación Nacional e internacional de propiedad Intelectual.

Firmado,

Carlos Ivan Sulca Galarza

DNI: 95722724

Pasaporte**:** 116349564

Resumen

Cuando se habla de Aplicaciones, muchos todavía piensan en las que se accede haciendo doble clic en el ícono del escritorio de Windows (u otro Sistema Operativo). Este tipo de aplicaciones, como así sus interfaces y bases de datos, suelen instalarse y residir en el equipo local y ser ejecutadas por el Sistema Operativo.

Por otro lado, las Aplicaciones Web son también programas informáticos pero que permiten a los visitantes de un sitio web enviar y recibir datos desde y hacia una base de datos a través de Internet utilizando un navegador web.

Características tales como webmail, newsletters, páginas de inicio de sesión, formularios de soporte y solicitud de productos, carritos de compras y sistemas de administración de contenido, son ejemplos de aplicaciones web, que dan forma a sitios web modernos, brindan a las empresas los medios necesarios para comunicarse con sus clientes y les proveen de un modelo de negocio que, a muchas de ellas, les permite incrementar exponencialmente sus beneficios.

A pesar de sus ventajas, algunas aplicaciones web plantean una serie de problemas de seguridad derivados de la **codificación incorrecta y el desbaratamiento de los mecanismos de autenticación**. De esta forma, vulnerabilidades explotables permiten a los atacantes ganar acceso a bases de datos administradas por aplicaciones web que contienen información valiosa y confidencial (por ejemplo, detalles personales y financieros), por lo que son un blanco frecuente de los piratas informáticos debido a los inmensos beneficios que reciben a cambio de esta información.

Para remediar estas brechas de seguridad es recomendable seguir las recomendaciones del Proyecto de Seguridad para Aplicaciones en Red Abierta (OWASP), nos brinda una **metodología** para minimizar los softwares inseguros, cumple un papel importante en desempeñar la solución de este grave problema. Es de vital importancia que nuestro enfoque para pruebas de software

por temas de seguridad se base en los principios de ingeniería y ciencia. Necesitamos un enfoque consistente, repetible y definido para las pruebas de aplicaciones web. Un mundo sin normas mínimas en materia de ingeniería y tecnología es un mundo caótico.

Owasp nos brinda una guía de pruebas con muchas maneras diferentes para detectar fallos de seguridad y esta guía captura el consenso de los principales expertos sobre cómo realizar esta prueba con rapidez, exactitud y eficiencia. OWASP da a personas de seguridad con conocimientos afines la capacidad de trabajar conjuntamente y formar un enfoque de práctica de liderazgo para un problema de seguridad.

Tabla de contenido

1. Introducción

El increíble mundo de la web hoy en día se ha vuelto un complemento de nuestra vida cotidiana y también en entornos corporativos por esta razón se crea la necesidad de implementar un sistema que permita mantener más seguro este ámbito web, esto no quiere decir que la implementación de una aplicación especializada en filtrar ataques web sea un sustitutivo de otras medidas de protección que debe llevar a cabo los desarrolladores. Las aplicaciones web sin protección son el punto más fácil de entrada para los hackers y son vulnerables a muchos tipos de ataques.

El Proyecto de Seguridad para Aplicaciones en Red Abierta (OWASP), es una organización internacional sin fines de lucro dedicada a la seguridad de las aplicaciones web . Uno de los principios fundamentales de OWASP es que todos sus materiales estén disponibles de forma gratuita y de fácil acceso en su sitio web, lo que permite que cualquiera pueda mejorar la seguridad de su propia aplicación web. Los materiales que ofrecen incluyen documentación, herramientas, videos y foros. Quizás su proyecto más conocido sea el OWASP Top 10.

La Guía de pruebas o Testing Guide proporcionada por OWASP ofrece una metodología organizada y práctica para saber cómo organizar una auditoria del software de aplicaciones web en cada momento de desarrollo de la aplicación así como información con todas las posibles áreas que puedan suponer un vector de ataque para las aplicaciones.

2. Objetivos

El objetivo de este trabajo es describir los beneficios que nos ofrece la metodología OWASP apoyados con un checklist para verificar el cumplimiento de la guía de pruebas en organizaciones argentinas. Puede ser visto como un framework referencial que comprende técnicas y tareas que son apropiadas en diferentes fases del ciclo de vida de desarrollo del software (SDLC). Las empresas y equipos de proyecto pueden utilizar este modelo para desarrollar su propio framework de pruebas y para mirar los servicios de pruebas de los proveedores.

La Guía de pruebas no puede considerarse como prescriptivo, sino como un enfoque flexible que puede ser extendido y moldeado para adaptarse a los procesos de desarrollo y cultura de la organización. Este framework de pruebas consiste de las siguientes actividades que deben ocurrir:

- Antes del inicio del desarrollo
- Durante la definición y diseño
- Durante el desarrollo
- Durante la implementación
- Mantenimiento y operaciones

3. Marco Teórico

Las amenazas a la seguridad de las redes e infraestructuras de las organizaciones han evolucionado enormemente y suponen una gran amenaza para las aplicaciones y servicios web, que constituyen el punto de acceso a la información confidencial y crítica guardada en las bases de datos de backend. En respuesta a todas estas amenazas se constituyeron los estándares regulatorios como PCI. Sin embargo, asegurar que las aplicaciones web están completamente libres de vulnerabilidades es complicado debido a la constante aparición de nuevas vulnerabilidades, necesidades de parcheos, revisiones de código, a las presiones del mercado sobre los plazos temporales de las aplicaciones, la dificultad de la identificación de vulnerabilidades e incluso las dificultades de acceso al código de las aplicaciones.

El Proyecto de Seguridad para Aplicaciones en Red Abierta o OWASP, es una organización internacional sin fines de lucro dedicada a la seguridad de las aplicaciones web . Uno de los principios fundamentales de OWASP es que todos sus materiales estén disponibles de forma gratuita y de fácil acceso en su sitio web, lo que permite que cualquiera pueda mejorar la seguridad de su propia aplicación web. Los materiales que ofrecen incluyen documentación, herramientas, videos y foros. Quizás su proyecto más conocido sea el OWASP Top 10.

La Guía de pruebas o Testing Guide proporcionada por OWASP ofrece una metodología organizada y práctica para saber cómo organizar una auditoria del software de aplicaciones web en cada momento de desarrollo de la aplicación así como información con todas las posibles áreas que puedan suponer un vector de ataque para las aplicaciones.

Este modelo se centra en ayudar a las organizaciones a probar sus aplicaciones Web para construir software confiable y seguro, más que simplemente poner de relieve las áreas de debilidad, aunque esto último es ciertamente utilizado en muchas guías y lista de verificación de OWASP.

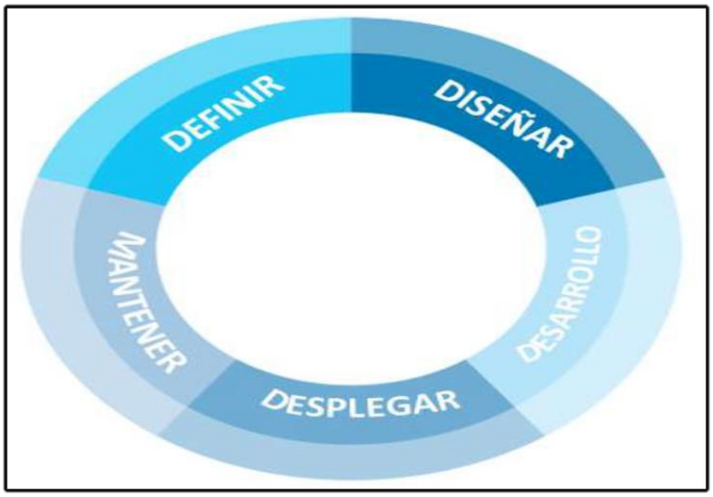
4. Ciclo de vida de desarrollo de software

El ciclo de vida del desarrollo del software (también conocido como SDLC o *Systems Development Life Cycle*) contempla las fases necesarias para validar el desarrollo del software y así garantizar que este cumpla los requisitos para la aplicación y verificación de los procedimientos de desarrollo, asegurándose de que los métodos usados son apropiados.

Su origen radica en que es muy costoso rectificar los posibles errores que se detectan tarde en la fase de implementación. Utilizando metodologías apropiadas, se podría detectar a tiempo para que los programadores puedan centrarse en la calidad del software, cumpliendo los plazos y los costes asociados.

Aunque existen diferentes ciclos de desarrollo de software, la normativa ISO/IEC/IEEE 12207:2017 establece:

"Un marco común para los procesos del ciclo de vida de los programas informáticos, con una terminología bien definida, a la que pueda remitirse la industria del software. Contiene procesos, actividades y tareas aplicables durante la adquisición, el suministro, el desarrollo, el funcionamiento, el mantenimiento o la eliminación de sistemas, productos y servicios informáticos. Estos procesos del ciclo de vida se llevan a cabo mediante la participación de los interesados, con el objetivo final de lograr la satisfacción del cliente".



Fases del ciclo de vida de desarrollo

La metodología para el desarrollo de software es un modo sistemático de realizar, gestionar y administrar un proyecto para llevarlo a cabo con grandes posibilidades de éxito. Esta sistematización indica cómo se divide un proyecto en módulos más pequeños para normalizar cómo se administra el mismo.

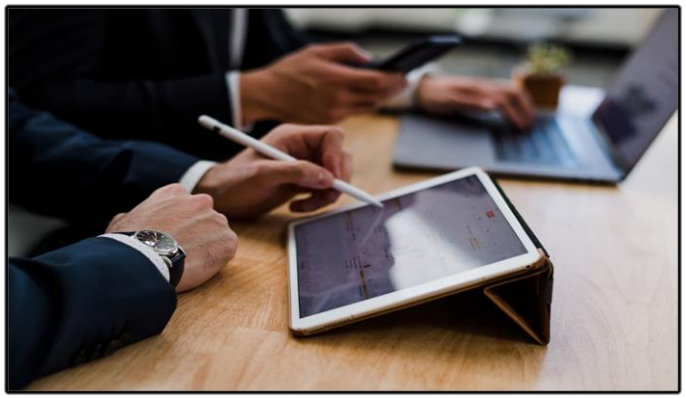
Así, una metodología para el desarrollo de software son los procesos a seguir sistemáticamente para idear, implementar y mantener un producto de software desde que surge la necesidad del producto hasta que se cumple el objetivo por el cual fue creado.

4.1 Fase de planificación

Antes de empezar un proyecto de desarrollo de un sistema de información, es necesario hacer ciertas tareas que influirán decisivamente en el éxito del mismo. Dichas tareas son conocidas como el **fuzzy front-end** del proyecto, puesto que no están sujetas a plazos.

Algunas de las tareas de esta fase incluyen actividades como la determinación del ámbito del proyecto, la realización de un estudio de viabilidad, el análisis de los riesgos asociados, la estimación del coste del proyecto, su planificación temporal y la asignación de recursos a las diferentes etapas del proyecto.

Por supuesto, hay que averiguar qué es exactamente lo que tiene que hacer el software. Por eso, la etapa de planificación en el ciclo de vida del software corresponde al proceso a través del cual se intenta descubrir qué es lo que realmente se necesita y se llega a una comprensión adecuada de los requerimientos del sistema (las características que el sistema debe poseer).



Planificación del proyecto

4.2 Fase de diseño

En esta fase se estudian posibles opciones de implementación para el software que hay que construir, así como decidir la estructura general del mismo. El diseño es una etapa compleja y su proceso debe realizarse de manera iterativa.

Es posible que la solución inicial no sea la más adecuada, por lo que en tal caso hay que refinarla. No obstante, hay catálogos de patrones de diseño muy útiles que recogen errores que otros han cometido para no caer en la misma trampa.

Es por ello que**,** a estas alturas del ciclo de vida de un sistema de información conviene identificar soluciones potenciales, evaluarlas y elegir la más conveniente. Ésta será o la más efectiva, o la más eficiente en costes o la menos compleja. Una vez completadas esas tareas, habrá que continuar haciendo la selección tecnológica de software y hardware, desarrollando las especificaciones para las distintas aplicaciones y obteniendo aprobación de la gerencia para poder proceder a la implementación del nuevo sistema.

Fase de diseño

4.3 Fase de desarrollo

En esta fase hay que elegir las herramientas adecuadas, un entorno de desarrollo que facilite el trabajo y un lenguaje de programación apropiado para el tipo de software a construir. Esta elección dependerá tanto de las decisiones de diseño tomadas como del entorno en el que el software deba funcionar.

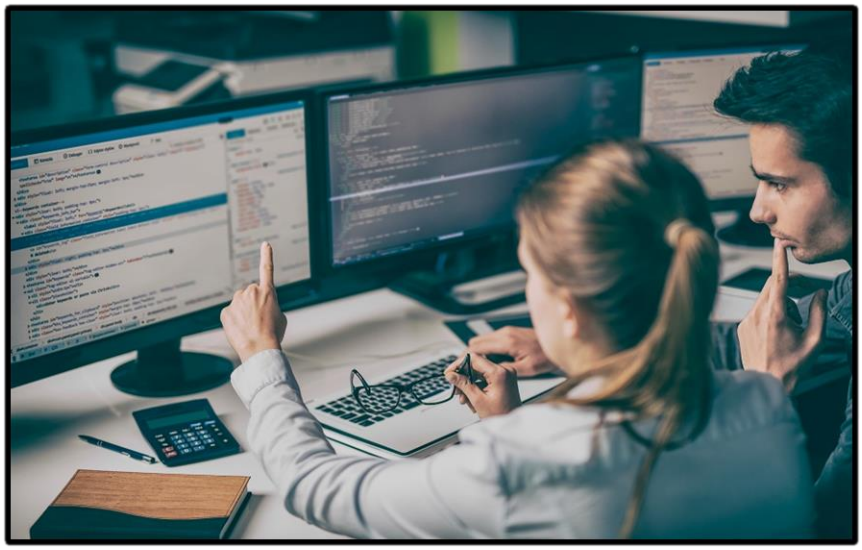
Al programar, hay que intentar que el código no sea indescifrable siguiendo distintas pautas como las siguientes:

- ✓ Evitar bloques de control no estructurados.
- ✓ Identificar correctamente las variables y su alcance.
- ✓ Elegir algoritmos y estructuras de datos adecuadas para el problema.
- ✓ Mantener la lógica de la aplicación lo más sencilla posible.
- ✓ Documentar y comentar adecuadamente el código de los programas.
- ✓ Facilitar la interpretación visual del código utilizando reglas de formato de código previamente consensuadas en el equipo de desarrollo.

También hay que tener en cuenta la adquisición de recursos necesarios para que el software funcione, además de desarrollar casos de prueba para comprobar el funcionamiento del mismo según se vaya programando.

El desarrollo software marca un antes y un después en la vida del sistema y significa, además, el inicio de la producción. El cambio es una constante durante esta etapa, en la que suele ser recomendable poner el foco en la formación y capacitación de los usuarios y el equipo técnico.

Puede ser necesario repetir pruebas tantas veces como haga falta para evitar errores y de hecho, conviene que el usuario final dé su conformidad con el resultado. Por último, este estadio concluye con la **verificación y validación,** que ayudan a asegurar el resultado del software.



Fase de desarrollo

4.4 Fase de implementación

En esta etapa del ciclo de vida de un sistema de información hay que proceder a la instalación del hardware y software elegidos, crear las aplicaciones correspondientes, someterlas a pruebas, crear la documentación pertinente y capacitar a los usuarios. La conversión de datos es importante en este estadio, en el que ya se empieza a trabajar en el nuevo sistema.

Se debe planificar el entorno teniendo en cuenta las dependencias existentes entre los diferentes componentes del mismo. Es posible que haya componentes que funcionen correctamente por separado, pero que al combinarlos provoquen problemas. Por ello, hay que usar combinaciones conocidas que no causen problemas de compatibilidad.

Fase de implementación

4.5 Fase de mantenimiento

Esta es una de las fases más importantes del ciclo de vida de desarrollo del software. Puesto que el software ni se rompe ni se desgasta con el uso, su mantenimiento incluye tres puntos diferenciados:

- ✓ Eliminar los defectos detectados durante su vida útil (mantenimiento correctivo).
- ✓ Adaptarlo a nuevas necesidades (mantenimiento adaptativo).
- ✓ Añadirle nuevas funcionalidades (mantenimiento perfectivo).

Aunque suene contradictorio, cuanto mejor es el software más tiempo hay que invertir en su mantenimiento. La principal razón es que se usará más (incluso de formas que no se habían previsto) y, por ende, habrá más propuestas de mejoras.

Fase de mantenimiento

5. Owasp

OWASP, es una organización sin fines de lucro a nivel mundial. Su objetivo es promover la codificación y el endurecimiento de aplicaciones seguras.

OWASP ha estado creciendo de forma constante desde 2004, y ha contribuido a proyectos como la aplicación de educación de seguridad

WebGoat. Ahora puede encontrar presentadores de OWASP en todas partes, desde las conferencias oficiales AppSec de OWASP en California.

5.1 OWASP Top 10-2017

El top 10 de OWASP es una lista de vulnerabilidades que los expertos en seguridad consideran las más serias amenazas de seguridad web. OWASP los actualiza periódicamente en función de los datos de ataque disponibles.

Muchas organizaciones grandes, incluida PCI, recomiendan que analice estas vulnerabilidades y las corrija o defienda en su contra, a continuación, el top 10:

- ✓ Inyección SQL.
- ✓ Pérdida de autenticación y gestión de sesiones.
- ✓ Exposición de datos sensibles.
- ✓ Entidad externa de XML (XXE).
- ✓ Pérdida de control de acceso.
- ✓ Configuraciones de seguridad incorrecta.
- ✓ Secuencia de comandos en sitios cruzados (XSS).
- ✓ Deserialización insegura.
- ✓ Uso de componentes con vulnerabilidades conocidas.
- ✓ Registro y monitoreo insuficientes.

6. Guía de pruebas para SDLC

La guía de pruebas tiene como objetivo alentar a las personas a medir la seguridad a través del proceso completo de desarrollo. Pueden, entonces, relacionar el costo del software inseguro con el impacto que tiene en el negocio y, en consecuencia, desarrollar procesos de negocios adecuados y

asignar recursos para manejar el riesgo. Recuerde que la medición y pruebas de aplicaciones web son incluso más críticas que otros programas, ya que las aplicaciones web están expuestas a millones de usuarios a través de Internet.

¿Qué es probar?

Probar es un proceso de comparar el estado de un sistema o una aplicación contra un conjunto de criterios. En la industria de seguridad, las personas, con frecuencia, prueban contra un conjunto de criterios mentales que no son bien definidos ni completos. Como resultado de esto, muchas personas ajenas consideran las pruebas como un arte oscuro. El objetivo de esta guía de pruebas es cambiar esa percepción y hacerlo más fácil para que personas sin conocimientos detallados de seguridad puedan hacer una diferencia en las pruebas.

¿Por qué realizar pruebas?

Con el presente trabajo queremos ayudar a las organizaciones a entender lo que comprende un programa de pruebas y para ayudarles a identificar los pasos que deben realizarse para construir y operar un programa de pruebas en aplicaciones web. La guía ofrece una amplia visión de los elementos necesarios para hacer un programa comprensible de seguridad para aplicaciones web. Esta guía puede utilizarse como una guía de referencia y 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 colegas del sector, para comprender la magnitud de los recursos necesarios para probar y mantener el software.

¿Cuándo probar?

Hoy en día, la mayoría de la gente 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 y utilizado en una aplicación web activa). Esto suele ser una práctica muy ineficaz y con un costo prohibitivo. Uno de los mejores métodos para impedir que haya fallos de seguridad que aparecen en

aplicaciones en producción es mejorar el Ciclo de Vida de Desarrollo de Software (SDLC), incluyendo seguridades en cada una de sus fases.

¿Qué se prueba?

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

Un programa efectivo de pruebas debe tener componentes que prueban:

- ✓ Personas – para asegurar que existe una educación adecuada y consciente.
- ✓ Proceso – para asegurar que hay criterios y políticas adecuadas y que las personas sepan cómo seguir estas políticas.
- ✓ Tecnología – para garantizar que el proceso ha sido eficaz en su implementación.

A menos que se adopte un enfoque integral, sólo probar la aplicación técnica de una aplicación no destapará la gestión o vulnerabilidades operacionales que podrían estar presentes. Probando a las personas, políticas y procesos, una organización puede encontrar temas que después se manifiesten en defectos en la tecnología, así como erradicar errores tempranamente e identificar la causa de los defectos.

Además, sólo probando algunas de las cuestiones técnicas que pueden estar presentes en un sistema resultará en una evaluación de seguridad incompleta e inexacta.

6.1 Guía de pruebas para SDLC – Fase de planificación

6.1.1 Defina un SDLC

Antes del inicio del desarrollo de aplicaciones, un SDLC adecuado debe ser definido donde la seguridad es inherente en cada etapa.

6.1.2 Revisar las políticas y estándares

Asegúrese de que existen adecuadas políticas, estándares y documentación en la organización. La documentación es extremadamente importante ya que da a los equipos las pautas de desarrollo y las políticas que pueden seguir.

La gente sólo puede hacer lo correcto si sabe lo que es lo correcto.Si la aplicación se desarrollara en Java, es esencial que exista un estándar de codificación segura de Java. Si la aplicación debe utilizar la criptografía, es esencial que haya un estándar de criptografía. Ninguna política o norma puede cubrir cada situación que enfrentará el equipo de desarrollo. Al documentar las cuestiones comunes y predecibles, habrá menos decisiones que necesiten ser hechas durante el proceso de desarrollo.

6.1.3 Desarrolle criterios de mediciones y métricas y asegure su trazabilidad

Antes de que comience el desarrollo, planifique el plan de medición. Definir los criterios que deben medirse proporciona visibilidad de los defectos tanto en el proceso como en el producto. Es esencial definir las métricas antes de que comience el desarrollo, ya que puede haber la necesidad de modificar el proceso con el fin de capturar los datos.

6.2 Guía de pruebas para SDLC – Fase de diseño

6.2.1 Revisar los requisitos de seguridad

Los requisitos de seguridad definen cómo funciona una aplicación desde una perspectiva de seguridad. Es esencial que los requerimientos de seguridad sean probados. Probar, en este caso, significa verificar los

supuestos que se hacen en los requisitos y las pruebas para ver si hay diferencias en las definiciones de los requisitos.

Por ejemplo, si hay un requisito de seguridad que indica que los usuarios deben estar registrados antes de que puedan acceder a la sección de documentos de un sitio web, ¿esto significa que el usuario debe estar registrado en el sistema o que el usuario esté autenticado? Asegúrese de que los requerimientos sean muy claros. Al buscar brechas de necesidades, considere mirar los mecanismos de seguridad tales como:

- ✓ Administración de usuarios.
- ✓ Autenticación.
- ✓ Autorización.
- ✓ Confidencialidad de datos.
- ✓ Integridad.
- ✓ Responsabilidad.
- ✓ Administración de sesiones.
- ✓ Seguridad de transporte.
- ✓ Segregación de sistema de información en niveles.
- ✓ Cumplimiento de legislación y estándares (incluidas las normas de privacidad, gubernamentales e industria).

6.2.2 Revise el diseño y arquitectura

Las aplicaciones deben tener una arquitectura y diseño documentados. Esta documentación puede incluir modelos, documentos textuales y otros artefactos similares. Es esencial probar estos artefactos para asegurar que el diseño y la arquitectura de la aplicación mantienen el nivel adecuado de seguridad según lo definido en los requisitos.

Identificar fallas de seguridad en la fase de diseño no sólo es uno de los momentos más eficientes en costo para identificar fallas, sino que puede ser uno de los momentos más eficaces para realizar cambios. Por ejemplo, si se identifica que el diseño exige autorización para las decisiones en varios lugares, puede ser apropiado considerar un componente de autorizaciones centralizado. Si la aplicación realiza una validación de datos en múltiples

lugares, puede ser apropiado desarrollar un marco de validación central (es decir, la validación de correcciones en un solo lugar, en vez de cientos de lugares; es mucho más económico).

Si se descubren debilidades, éstas deben ser entregadas al arquitecto del sistema para buscar enfoques alternativos.

6.2.3 Cree y revise modelos UML

Una vez que el diseño y la arquitectura estén completos, construya modelos de Lenguaje de Modelaje Unificado (UML) que describan cómo funciona la aplicación. En algunos casos, estos ya pueden estar disponibles. Use estos modelos para confirmar con los diseñadores de sistemas una comprensión exacta de cómo funciona la aplicación. Si se descubren debilidades, éstas deben ser entregadas al arquitecto del sistema para buscar enfoques alternativos.

6.2.4 Cree y revise modelos de amenazas

Provisto con la revisión del diseño y la arquitectura de los modelos UML, y habiendo aclarado exactamente cómo funciona el sistema, lleve a cabo un ejercicio de modelaje de amenazas. Desarrolle escenarios realistas de las amenazas. Analice el diseño y la arquitectura para asegurar que estas amenazas han sido mitigadas, aceptadas por el negocio o asignadas a una tercera persona, como una empresa de seguros. Cuando las amenazas identificadas no tengan estrategias de mitigación, revise el diseño y la arquitectura con el arquitecto de sistemas para modificar el diseño.

6.3 Guía de pruebas para SDLC – Fase de desarrollo

En teoría, el desarrollo es la aplicación de un diseño. Sin embargo, en el mundo real, muchas decisiones de diseño se realizan durante el desarrollo de la codificación. Son, a menudo, decisiones pequeñas que eran demasiado detalladas para ser descritas en el diseño, o temas donde no se ofrecieron políticas o estándares de orientación. Si el diseño y la arquitectura no fueran adecuados, el desarrollador se enfrentará a muchas decisiones. Si hay normas y políticas insuficientes, el desarrollador se enfrentará aún a más

decisiones.

6.3.1 Camine a través del código

El equipo de seguridad debería realizar una "caminata" a través del código con los desarrolladores y, en algunos casos, los arquitectos del sistema. Una caminata a través del código es un recorrido de alto nivel a través del código, donde los desarrolladores pueden explicar la lógica y el flujo del código implementado. Esta caminata permite al equipo de revisión de código obtener una comprensión general del código y permite a los desarrolladores explicar por qué ciertas cosas se desarrollaron de la manera en que lo hicieron. El propósito no es realizar una revisión de código, sino entender en un nivel alto el flujo, el diseño y la estructura del código que compone la aplicación.

6.3.2 Revisiones de código

Armado con una buena comprensión de cómo está estructurado el código y por qué ciertas cosas fueron codificadas de la manera en que lo fueron, el evaluador puede examinar ahora el código real en busca de defectos de seguridad. Las revisiones de código estático validan el código con una serie de listas de verificación, que incluyen:

- ✓ Requerimientos del negocio para la disponibilidad, confidencialidad e integridad.
- ✓ Guía OWASP o listado Top 10 para exposiciones técnicas (dependiendo de la profundidad de la revisión).
- ✓ Cuestiones específicas relacionadas con el lenguaje o el framework utilizado, tales como para PHP, JAVA o Microsoft ASP.NET, etc.
- ✓ Los requisitos específicos de la industria, tales como Sarbanes-Oxley 404, COPPA, ISO/IEC 27002, APRA, HIPAA, las guías de Visa Merchant u otros regímenes normativos.

En términos del retorno sobre los recursos invertidos (sobre todo tiempo), las revisiones de código estático producen rendimientos de mayor calidad que cualquier otro método de revisión de seguridad y dependen menos

en la habilidad del revisor. Sin embargo, no son una solución milagrosa y necesitan ser consideradas cuidadosamente dentro de un régimen de prueba de amplio espectro.

6.4 Guía de pruebas para SDLC – Fase de implementación

6.4.1 Prueba de aplicación y penetración

Habiendo probado los requisitos, analizado el diseño y realizada la revisión de código, se puede suponer que todos los temas han sido cubiertos. Esperemos que este sea el caso, pero realizar pruebas de penetración de la aplicación después de que se ha implementado proporciona una última comprobación para asegurarse de que nada se ha escapado.

6.4.2 Prueba de gestión de configuración

La prueba de penetración de la aplicación debe incluir la comprobación de cómo la infraestructura fue implementada y asegurada. Aunque la aplicación puede ser segura, un pequeño aspecto de la configuración podría estar todavía en una fase de instalación por defecto y ser vulnerable a la explotación.

6.5 Guía de pruebas para SDLC – Fase de mantenimiento

6.5.1 Revisión de la gestión de la conducta operacional

Debe existir un proceso implantado que detalle cómo se maneja tanto la parte operativa de la aplicación como la infraestructura.

6.5.2 Realice pruebas de salud de la conducta periódicamente

Las pruebas de salud de la conducta deben realizarse mensual o trimestralmente, tanto sobre la aplicación como sobre la infraestructura para garantizar que no hayan aparecido nuevos riesgos de seguridad y que el nivel de seguridad esté todavía intacto.

6.5.3 Garantice la verificación después del cambio

Después de que cada cambio ha sido aprobado y probado en el entorno de control de calidad e implementado en el entorno de producción, es de vital importancia que el cambio sea comprobado para asegurarse de que el nivel de seguridad no ha sido afectado por él. Esto debe estar integrado dentro del proceso de gestión del cambio.

Flujo de trabajo de pruebas SDLC

Flujo de trabajo de pruebas OWASP

7. Pruebas de seguridad para las aplicaciones web

Las pruebas de seguridad nunca serán una ciencia exacta donde se puede definir una lista completa de todos los temas posibles que deben ser probados. De hecho, las pruebas de seguridad son sólo una técnica apropiada para probar la seguridad de aplicaciones web bajo ciertas circunstancias.

El conjunto de pruebas activas se han dividido en 11 categorías para un total de 91 controles:

- ✓ Recopilación de información.
- ✓ Pruebas de gestión de configuración e implementación.
- ✓ Pruebas de gestión de identidad.
- ✓ Pruebas de autenticación.
- ✓ Pruebas de autorización.
- ✓ Pruebas de gestión de sesión.
- ✓ Pruebas de validación de ingreso.
- ✓ Manejo de errores.
- ✓ Criptografía.
- ✓ Pruebas de lógica del negocio.
- ✓ Pruebas del punto de vista del cliente.

7.1 Pruebas de recopilación de información

Entender la configuración implementada del servidor que aloja la aplicación web es casi tan importante como la aplicación misma de pruebas de seguridad. Después de todo, una cadena de aplicaciones sólo es tan fuerte como su eslabón más débil. Las plataformas de aplicación son amplias y variadas, pero algunos errores clave de configuración de la plataforma pueden comprometer la aplicación de la misma manera que una aplicación no segura puede comprometer al servidor.

7.1.1 Mediante un motor de búsqueda (OTG-INFO-001)

Descripción

Existen elementos directos e indirectos para el descubrimiento y reconocimiento mediante motores de búsqueda. Los métodos directos se refieren a buscar en los índices y el contenido asociado al caché. Los métodos indirectos se refieren a información sensible de la configuración y diseño mediante búsquedas en foros, grupos de noticias y sitios de licitación web.

Una vez que un robot de motores de búsqueda ha terminado de arrastrarse, comienza la indexación de la página web basada en etiquetas y atributos asociados, tales como<TITLE>, con el fin de devolver los resultados de búsqueda relevantes. Si el archivo robots.txt no está actualizado durante

la vida útil del sitio web y en línea HTML las meta etiquetas, que indican a los robots que no generen índices del contenido, no han sido utilizadas, entonces es posible que los índices incluyan contenido web cuya inclusión no estaba prevista por parte de los propietarios. Los propietarios de páginas web pueden utilizar el mencionado robots.txt, meta etiquetas HTML, autenticación y herramientas proporcionados por los motores de búsqueda para eliminar dicho contenido.

Objetivos de la prueba

Para entender qué información sensible del diseño y configuración de la aplicación/sistema/organización está expuesta directamente (en la página web de la organización) o indirectamente (en un sitio web de terceros).

Herramientas

- ✓ Foundstone sitedigger.
- ✓ Google hacking.
- ✓ Punkspider.
- ✓ Shodan.
- ✓ Foca.

Remediación

Considere cuidadosamente la sensibilidad de la información del diseño y configuración antes de publicarla en línea. Periódicamente revise la sensibilidad de la información del diseño y configuración existente que está publicada en línea.

7.1.2 Huellas digitales en el servidor web (OTG-INFO-002)

Descripción

El utilizar huellas digitales en el servidor web es una tarea fundamental para el evaluador de penetración. Conocer la versión y el tipo de un servidor web en ejecución permite a los evaluadores determinar

vulnerabilidades conocidas y cómo explotarlas adecuadamente para usarlas durante la prueba.

Hay varios proveedores diferentes y versiones de servidores web en el mercado hoy. Conocer el tipo de servidor web que está siendo probado ayuda significativamente en el proceso de prueba, y también puede cambiar el curso de la prueba.

Esta información se puede derivar enviando al servidor web comandos específicos y analizando la respuesta de salida, como cada versión de software del servidor web puede responder de manera distinta a estos comandos. Sabiendo cómo responde cada tipo de servidor web a comandos específicos y manteniendo esta información en una base de datos de huellas digitales de servidores web, un evaluador de penetración puede enviar estos comandos al servidor web, analizar la respuesta y compararla con la base de datos de firmas conocidas.

Objetivos de la prueba

Encontrar la versión y el tipo de servidor web en ejecución para determinar vulnerabilidades conocidas y la manera de explotarlas para usar durante la prueba.

Herramientas

- ✓ Httprint.
- ✓ Httprecon.
- ✓ Netcraft.

Remediación

Proteja la capa de presentación del servidor web detrás de un proxy inverso endurecido. Ofusque la capa de presentación de los encabezados del servidor web.

- ✓ Apache.
- ✓ IIS.

7.1.3 Revisión de meta archivos del servidor web en busca de fugas de información (OTG-INFO-003)

Descripción

Esta sección describe cómo probar el archivo robots.txt en busca de fugas de información de la(s) ruta(s) al directorio de la aplicación o de la carpeta de la aplicación web. Además, también puede crearse la lista de directorios que deben ser evitados por los robots o rastreadores como una dependencia de rutas de ejecución del mapa a través de la aplicación (OTG-INFO-007).

Objetivos de la prueba

Fuga de información de la ruta o rutas al directorio o carpeta de la aplicación web. Crear la lista de directorios que deben ser evitados por los robots o rastreadores.

Herramientas

- ✓ Curl.
- ✓ Wget.
- ✓ Rockspider.

7.1.4 Enumere las aplicaciones en el servidor web (OTG-INFO-004)

Descripción

Un paso fundamental en las pruebas de vulnerabilidades en aplicaciones web es averiguar qué aplicaciones particulares están alojadas en un servidor web. Muchas aplicaciones tienen vulnerabilidades y estrategias de ataque conocidas que pueden ser explotadas para obtener el control remoto para explotar los datos. Además, muchas aplicaciones se encuentran a menudo mal configuradas o no están actualizadas, debido a la percepción de que sólo se utilizan "internamente" y, por lo tanto, no existe amenaza.

Con la proliferación de servidores web virtuales, la relación tipo 1:1-

tradicional entre una dirección IP y un servidor web está perdiendo gran parte de su significado original. No es raro tener varios sitios web o aplicaciones cuyos nombres simbólicos resulten en la misma dirección IP. Este escenario no se limita a entornos de alojamiento (hosting), sino que también se aplica a los entornos corporativos.

Objetivos de la prueba

Enumerar las aplicaciones que existen dentro del ámbito en un servidor web.

Herramientas

- ✓ Herramientas de búsqueda DNS como nslookup, dig y similares.
- ✓ Motores de búsqueda (Google, Bing y otros motores de búsqueda principales).
- ✓ Servicios especializados relacionados con DNS basado en la web: ver texto.
- ✓ Nmap.
- ✓ Nessus Vulnerability Scanner.
- ✓ Nikto.

7.1.5 Comentarios sobre el sitio web y los metadatos en busca de fugas de información (OTG-INFO-005)

Descripción

Es muy común e incluso recomendable para los programadores el incluir comentarios detallados y metadatos en su código fuente. Sin embargo, los comentarios y metadatos incluidos en el código HTML podrían revelar información interna que no debería estar disponible a atacantes potenciales. Los comentarios y metadatos deben ser revisados con el fin de determinar si hay fugas de información.

Objetivos de la prueba

Revise los comentarios y metadatos de la página web para entender mejor la aplicación y encontrar cualquier fuga de información.

Herramientas

- ✓ Wget.
- ✓ Eyeballs
- ✓ Curl.

7.1.6 Identificar puntos de entrada de la aplicación (OTG-INFO-006)

Descripción

Enumerar la aplicación y su superficie de ataque es un precursor clave antes que cualquier prueba de fondo puede llevarse a cabo, ya que permite al evaluador identificar probables áreas de debilidad. Esta sección pretende ayudar a identificar y mapear las áreas dentro de la aplicación que deben investigarse una vez que la enumeración y el mapeo se han completado.

Objetivos de la prueba

Entender cómo se forman las solicitudes y las respuestas típicas de la aplicación.

Herramientas

- ✓ OWASP: Zed Attack Proxy (ZAP).
- ✓ OWASP: WebScarab.
- ✓ Burp Suite.
- ✓ CAT.

7.1.7 Crear mapas de las rutas de ejecución a través de la aplicación (OTG-INFO-007)

Descripción

Antes de comenzar las pruebas de seguridad, es de suma importancia entender la estructura de la aplicación. Sin una comprensión profunda de la distribución de la aplicación, es poco probable que sea probada exhaustivamente.

Objetivos de la prueba

Crear mapas de la aplicación de destino y comprender los principales flujos de trabajo.

Herramientas

- ✓ Zed Attack Proxy (ZAP).
- ✓ Spreadsheet software.
- ✓ Diagramming software.

7.1.8 Framework referencial para el uso de huellas digitales en aplicaciones web (OTG-INFO-008)

Descripción

El framework web de huellas digitales es una subtarea importante del proceso de recolección de información. Conocer el tipo de framework puede automáticamente dar una gran ventaja si este framework ya ha sido probado mediante pruebas de penetración. No son sólo las vulnerabilidades conocidas en versiones sin parches, sino configuraciones erróneas específicas en el framework y la estructura de archivo conocido que hace tan importante el proceso de marcar con huellas digitales.

Se utilizan varios proveedores diferentes y versiones de los frameworks web. La información al respecto de estos ayuda significativamente en el proceso de pruebas y también puede ayudar a cambiar el curso de la prueba. Dicha información puede ser derivada de un cuidadoso análisis de ciertos lugares comunes. La mayoría de los frameworks web tienen varios marcadores en esos lugares, lo que ayuda a un atacante a detectarlos. Esto es básicamente lo que todas las herramientas automáticas hacen: buscar un marcador desde una ubicación predefinida y luego compararlo con la base de

datos de firmas conocidas. Para mayor precisión se utilizan, generalmente, varios marcadores.

Objetivos de la prueba

El tipo de framework web usado, así como tener una mejor comprensión de la metodología de pruebas de seguridad.

Herramientas

Una lista de herramientas generales y muy conocidas se presenta a continuación. También hay un sinfín de otras utilidades, así como herramientas de huellas digitales basadas en frameworks.

- ✓ Whatweb.
- ✓ Blindelephant.
- ✓ Wappalyzer.

Remediación

El consejo general es usar varias de las herramientas descritas anteriormente y verificar los registros para entender exactamente lo que ayuda a un atacante a revelar el framework de la web. Mediante la realización de múltiples análisis después de que se han hecho cambios para ocultar las rutas del framework, es posible alcanzar un mejor nivel de seguridad y asegurarse de que el framework no puede ser detectado por análisis automático. A continuación, se presentan algunas recomendaciones específicas, de acuerdo a la ubicación del marcador del framework y algunos enfoques adicionales interesantes.

7.1.9 Aplicación de huellas digitales para web (OTG-INFO-009)

Descripción

No hay nada nuevo bajo el sol, y casi cada aplicación web que se puede pensar en desarrollar ya ha sido desarrollada. Con la gran cantidad de proyectos de software libre y de código abierto que son desarrollados y

desplegados activamente alrededor del mundo, es muy probable que una prueba de seguridad enfrentará un sitio objetivo que es total o parcialmente dependiente de una de estas aplicaciones muy conocidas (por ejemplo, Wordpress, phpBB, Mediawiki, etc.). Conocer los componentes de las aplicaciones web que se están probando ayuda significativamente en el proceso de prueba y se reduce drásticamente el esfuerzo requerido durante la prueba. Estas aplicaciones web conocidas tienen encabezados HTML, cookies y estructuras de directorios que se pueden enumerar para identificar la aplicación.

Objetivo de la prueba

Identificar la aplicación web y la versión para determinar vulnerabilidades conocidas y la formas de explotarlas apropiadamente para usar durante la prueba.

Herramientas

Una lista de herramientas generales y muy conocidas se presenta a continuación. También hay un sinfín de otras utilidades, así como herramientas de huellas digitales basadas en frameworks.

- ✓ Whatweb.
- ✓ Blindelephant.
- ✓ Wappalyzer.

7.1.10 Crear un mapa de la arquitectura de la aplicación (OTG-INFO-010)

Descripción

La complejidad de la infraestructura de servidores web interconectados y heterogéneos puede incluir cientos de aplicaciones web y hace de la gestión de configuración y de la revisión un paso fundamental en la prueba e implementación de cada aplicación. De hecho, se necesita sólo una vulnerabilidad para socavar la seguridad de toda la infraestructura, e incluso problemas pequeños y sin importancia aparente pueden convertirse

en serios riesgos para otra aplicación en el mismo servidor.

Para abordar estos problemas, es de suma importancia llevar a cabo una profunda revisión de la configuración y problemas de seguridad conocidos. Antes de realizar un examen a fondo es necesario crear un mapa de la red y de la arquitectura de la aplicación. Los diferentes elementos que conforman la infraestructura necesitan ser determinados para entender cómo interactúan con una aplicación web y cómo ellos afectan a la seguridad.

7.2 Pruebas de gestión de configuración e implementación

7.2.1 Prueba la configuración de la infraestructura y red (OTG-CONFIG-001)

Descripción

La complejidad intrínseca de una infraestructura de servidor web interconectada y heterogénea, que puede incluir cientos de aplicaciones web, hace de la gestión de la configuración y revisión un paso fundamental en la prueba e implementación de cada aplicación. Toma sólo una vulnerabilidad el socavar la seguridad de toda la infraestructura y problemas e incluso problemas pequeños y aparentemente sin importancia pueden convertirse en serios riesgos para otra aplicación en el mismo servidor. Para abordar estos problemas, es de suma importancia llevar a cabo una profunda revisión de la configuración y problemas de seguridad conocidos, después de haber creado un mapa de toda la arquitectura.

Una gestión apropiada de la configuración la infraestructura del servidor de web es muy importante para preservar la seguridad de la propia aplicación. Si elementos como el software del servidor web, los servidores de base de datos de back-end o los servidores de autenticación no está revisados y asegurados, podrían presentar riesgos no deseados o introducir nuevas vulnerabilidades que podrían comprometer a la propia aplicación.

Los siguientes pasos deben tomarse para poner a prueba la infraestructura de gestión de configuración:

- ✓ Los diferentes elementos que conforman la infraestructura deben ser determinados con el fin de comprender cómo interactúan con una aplicación web y cómo afectan a su seguridad.
- ✓ Todos los elementos de la infraestructura deben revisarse para asegurarse de que no contienen vulnerabilidades conocidas.
- ✓ Debe hacerse una revisión de las herramientas administrativas usadas para dar mantenimiento a los diferentes elementos.
- ✓ Los sistemas de autenticación necesitan ser revisados para asegurarse que sirven a las necesidades de la aplicación y que no pueden ser manipulados por los usuarios externos para obtener el acceso.
- ✓ Una lista de puertos definidos que se requieren para la aplicación debe recibir mantenimiento y debe guardarse con un control de cambios.

Después de haber elaborado un mapa de los diferentes elementos que conforman la infraestructura (vea mapa de red y arquitectura de la aplicación), es posible revisar la configuración de cada elemento encontrado y probarlos en busca de cualquier vulnerabilidad conocida.

Herramientas administrativas

Cualquier infraestructura de servidor web requiere la existencia de herramientas administrativas para dar mantenimiento y actualizar la información utilizada por la aplicación. Esta información incluye contenido estático (páginas web, archivos gráficos), código fuente de la aplicación, bases de datos de autenticación de usuario, etc. Las herramientas administrativas serán diferentes según el sitio, la tecnología o el software utilizado. Por ejemplo, algunos servidores se gestionan mediante interfaces administrativas, que son estos mismos servidores web (como el servidor web iPlanet) o serán administradas por archivos de texto con la configuración (en caso del Apache) que usen un sistema operativo GUI tools (al usar el servidor IIS o ASP.Net de Microsoft).

En la mayoría de los casos se controlará la configuración del servidor

con diferentes herramientas de mantenimiento de archivos utilizados por el servidor web, los cuales son administrados a través de servidores FTP, WebDAV, sistemas de archivos de red (NFS, CIFS) u otros mecanismos. Obviamente, el sistema operativo de los elementos que componen la arquitectura de la aplicación también se gestionará utilizando otras herramientas. Las aplicaciones también pueden tener interfaces administrativas incrustadas en ellas, que se utilizan para gestionar los datos mismos de la aplicación (usuarios, contenido, etc.).

Después de haber elaborado mapas de las interfaces administrativas utilizadas para administrar las diferentes partes de la arquitectura, es importante revisarla, ya que, si un atacante obtiene acceso a cualquiera de ellas, entonces puede comprometer o dañar la arquitectura de la aplicación. Para ello es importante:

- ✓ Determinar los mecanismos que controlan el acceso a estas interfaces y sus vulnerabilidades asociadas. Esta información puede estar disponible en línea.
- ✓ Cambiar el usuario y contraseña generados automáticamente.

Algunas empresas optan por no gestionar todos los aspectos de sus aplicaciones de servidor web, pero pueden tener otros equipos gestionando el contenido entregado por la aplicación web. Esta empresa externa puede solamente proporcionar partes de los contenidos (actualizaciones de noticias o promociones) o puede administrar el servidor de web totalmente (incluyendo contenido y código). Es común encontrar interfaces administrativas disponibles desde Internet en estas situaciones, ya que usar el Internet es más barato que tener una línea dedicada que conecte a la empresa externa únicamente con la infraestructura de la aplicación a través de una interfaz de gestión. En esta situación, es muy importante comprobar si las interfaces administrativas son vulnerables a ataques.

7.2.2 Prueba la configuración de la plataforma de aplicaciones (OTG-CONFIG-002)

Descripción

Es importante una adecuada configuración de los elementos individuales que conforman la arquitectura de una aplicación, para prevenir errores que podrían comprometer la seguridad de la arquitectura entera. Revisar y probar la configuración es una tarea fundamental en la creación y mantenimiento de una arquitectura. Esto es porque muchos sistemas diferentes generalmente cuentan con configuraciones genéricas que podrían no ser adecuadas para la tarea que se llevará a cabo en el sitio específico donde están instaladas.

Mientras que la instalación típica del servidor de la web y de la aplicación contendrá mucha funcionalidad (como ejemplos de aplicación, documentación, páginas de prueba) lo que no es esencial debe retirarse antes de la implementación para evitar la explotación de estos después de la instalación.

7.2.3 Prueba el manejo de archivos de extensiones en busca de información sensible (OTG-CONFIG-003)

Descripción

Los archivos de extensiones se utilizan en los servidores web para determinar fácilmente qué tecnologías, idiomas y accesorios (plugins) deben utilizarse para cumplir con la solicitud web. Si bien este comportamiento es compatible con los RFC y estándares Web, utilizar las extensiones de archivo estándar proporciona al evaluador de penetración la información útil sobre las tecnologías subyacentes que se utilizan en una aplicación web y simplifica enormemente la tarea de determinar el escenario de ataque a usar para estas tecnologías en particular. Además, un error de configuración de los servidores web fácilmente podría revelar información confidencial sobre las credenciales de acceso.

El control de extensiones a menudo se utiliza para validar los archivos que serán cargados, que pueden conducir a resultados inesperados debido a que el contenido no es el esperado, o por manejo de nombres de archivo inesperado del OS.

Determinar cómo los servidores web manejan las peticiones

correspondientes a archivos con diferentes extensiones puede ayudar a comprender el comportamiento del servidor web según el tipo de archivos a los que se accede. Por ejemplo, puede ayudar a entender cuáles extensiones de archivo se devuelven como texto simple frente a aquellos que causan alguna ejecución en el lado del servidor. Estos últimos son indicativos de las tecnologías, idiomas o plugins que son utilizados por los servidores web o servidores de aplicaciones y pueden proporcionar información adicional de cómo está diseñada la aplicación web. Por ejemplo, una extensión de ".pl" es generalmente asociada con un soporte Perl del lado del servidor. Sin embargo, la extensión de archivo sola puede ser engañosa y no completamente concluyente.

Herramientas

Los escáneres de vulnerabilidad, tales como Nessus y Nikto, comprueban la existencia de directorios web conocidos. Pueden permitir que el evaluador descargue la estructura del sitio web, lo cual es útil cuando se intenta determinar la configuración de los directorios web y cómo son atendidas las extensiones de archivo individuales. Otras herramientas que pueden utilizarse para este propósito incluyen:

- ✓ Wget.
- ✓ Curl.
- ✓ Google for “web mirroring tools”.

7.2.4 Revisar archivos viejos, copias de seguridad y archivos no referenciados en busca de información sensible (OTG-CONFIG-004)

Descripción

Mientras que la mayoría de los archivos en un servidor web son manejados directamente por el servidor, no es raro encontrar archivos no referenciados u olvidados que pueden utilizarse para obtener información importante acerca de la infraestructura o las credenciales. Los escenarios más comunes incluyen la presencia de viejas versiones renombradas de archivos modificados, archivos de inclusión que se cargan en el idioma de su

preferencia y pueden ser descargados como fuente, o incluso copias de seguridad manuales o automáticas o en forma de archivos comprimidos. Los archivos de copias de seguridad también pueden ser generados automáticamente por el sistema de archivos subyacente donde se aloja la aplicación, una característica que se conoce generalmente como "instantáneas".

Todos estos archivos podrán conceder acceso al evaluador a trabajos internos, puertas traseras, interfaces administrativas o incluso credenciales para conectarse a la interfaz administrativa o al servidor de base de datos. Una fuente importante de vulnerabilidades se encuentra en los archivos que no tienen nada que ver con la aplicación, pero se crean como consecuencia de la edición de archivos de la aplicación, o después de crear copias de seguridad o dejando en el árbol de la red viejos archivos del árbol o archivos no referenciados. Realizar ediciones ¨"en el sitio" u otras acciones administrativas en servidores web en producción sin darse cuenta puede dejar copias de seguridad, ya sean generadas automáticamente por el editor mientras se editan archivos, o por el administrador quien está comprimiendo un conjunto de archivos para crear una copia de seguridad.

En general, exponer el código del lado del servidor es una mala idea. No sólo está usted innecesariamente exponiendo la lógica del negocio, sino que, sin saberlo, usted puede estar exponiendo información relacionada con la aplicación, lo que puede ayudar a un atacante (nombres de ruta de acceso, estructuras de datos, etc.). Por no mencionar el hecho de que hay muchos scripts que tienen incrustado el usuario y contraseña en texto claro (que es una práctica imprudente y muy peligrosa).

Herramientas

- ✓ Las herramientas de evaluación de vulnerabilidad tienden a incluir verificaciones a directorios web que tienen nombres estándar (como “admin”,“test”, “backup”, etc.) y a reportar cualquier directorio web que permite la indexación. Si usted no puede conseguir un listado de directorios, debe intentar comprobar extensiones de respaldo probables. Compruebe,

por ejemplo, Nessus (nessus.org), Nikto2(cirt.net) o su nuevo derivado Wikto (sensepost.com), que también es compatible con Google hacking based strategies.

- ✓ Las herramientas robot: wget (gnu.org, interlog.com); Sam Spade (samspade.org); Spike Proxy incluyen una función de rastreador del sitio web (immunitysec.com); Xenu (home.snafu.de); Curl (curl.haxx.se). Algunos de ellos también se incluyen en las distribuciones estándar de Linux.
- ✓ Las herramientas de desarrollo web suelen incluir instalaciones para identificar los enlaces rotos y los archivos no referenciados.

Remediación

Para garantizar una estrategia de protección efectiva, la prueba debe estar compuesta por una política de seguridad que específicamente prohíbe las prácticas peligrosas tales como:

- ✓ Editar los archivos en el lugar del servidor web o sistemas de archivos del servidor de aplicaciones. Este es un mal hábito particular, ya que es probable que, sin querer, los editores generen archivos de respaldo. Es sorprendente ver que esto se hace frecuentemente, incluso en grandes organizaciones. Si definitivamente necesita editar archivos en un sistema en producción, asegúrese de no dejar atrás cualquier cosa que no esté explícitamente planificada y recuerde que lo está haciendo bajo su propio riesgo.
- ✓ Revise cuidadosamente cualquier otra actividad realizada en los sistemas de archivos expuestos por el servidor web, como las actividades de la administración en el punto. Por ejemplo, si usted necesita de vez en cuando tomar una instantánea de un par de directorios (que no se debe hacer en un sistema en producción), puede sentirse tentado a comprimirlos primero. Tenga cuidado de no dejar atrás esos archivos.
- ✓ Las políticas de gestión de configuración apropiada deberían ayudar a no dejar por ahí archivos obsoletos y sin referencia.

- ✓ Las aplicaciones deben ser diseñadas para no crear (o depender de) archivos almacenados debajo de los árboles de directorios web atendidos por el servidor web. Los archivos de datos, archivos de registro, archivos de configuración, etc. deben almacenarse en directorios no accesibles por el servidor web, para contrarrestar la posibilidad de divulgación de información (sin mencionar la modificación de los datos si los permisos del directorio web permiten escritura).
- ✓ Las instantáneas de sistema de archivos no deben ser accesibles a través de la web si la raíz del documento es un sistema de archivos que utiliza esta tecnología. Configure el servidor web para negar el acceso a dichos directorios.

7.2.5 Infraestructura de enumeración e interfaces de administración de aplicaciones (OTG-CONFIG-005)

Descripción

Las interfaces del administrador se pueden presentar en la aplicación o en el servidor de aplicaciones para permitir que determinados usuarios puedan llevar a cabo actividades privilegiadas en el sitio. Se deben realizar pruebas para revelar sí y cómo esta funcionalidad privilegiada puede ser accesada por un usuario no autorizado o estándar. Una aplicación puede requerir una interfaz de administrador para habilitar un usuario con privilegios con acceso a funciones que pueden realizar cambios de cómo funciona el sitio. Tales cambios pueden incluir:

- ✓ Provisionamiento de cuentas de usuario.
- ✓ Diseño y diagramación del sitio.
- ✓ Manipulación de datos.
- ✓ Cambios en la configuración.

En muchas instancias, dichas interfaces no tienen suficientes controles para protegerlas de accesos no autorizados. La prueba está dirigida a descubrir estas interfaces de administrador y acceder a funcionalidades para estos usuarios privilegiados.

Herramientas

- ✓ Dirbuster este proyecto OWASP actualmente inactivo sigue siendo una gran herramienta para forzar directorios y archivos en el servidor.
- ✓ THC-HYDRA es una herramienta que permite forzar muchas interfaces, incluyendo la autenticación HTTP basada en formularios.
- ✓ Un forzador es mucho mejor cuando usa un buen diccionario, por ejemplo, el diccionario de netsparker.

7.2.6 Pruebe los métodos HTTP (OTG-CONFIG-006)

Descripción

HTTP ofrece una serie de métodos que pueden utilizarse para realizar acciones en el servidor web. Muchos de los métodos están diseñados para ayudar a los desarrolladores a implementar y probar aplicaciones HTTP. Estos métodos HTTP pueden utilizarse para fines malvados si el servidor web está mal configurado. Además, el Cross Site Tracing (XST), una forma de escritura de sitios cruzados que utiliza el método TRACE del servidor HTTP, es examinado. Aunque GET y POST son los métodos más comunes que se utilizan para acceder a la información proporcionada por un servidor web, el protocolo de transferencia de hipertexto (HTTP) permite varios otros métodos (y algunos menos conocidos). RFC 2616 (que describe al HTTP versión 1.1 que es el estándar hoy en día) define los siguientes ocho métodos:

- ✓ HEAD.
- ✓ GET.
- ✓ POST.
- ✓ PUT.
- ✓ DELETE.
- ✓ TRACE.
- ✓ OPTIONS.
- ✓ CONNECT.

Herramientas

- ✓ NetCat.
- ✓ CURL.

7.2.7 Pruebe el HTTP strict transport security (OTG-CONFIG-007)

Descripción

El encabezado HTTPS Strict Transport Security (HSTS) es un mecanismo que tienen los sitios web para comunicarse con los navegadores web. Todo el tráfico intercambiado con un dominio debe siempre ser enviado mediante https; esto ayudará a proteger la información de que se envié mediante peticiones no cifradas. Considerando la importancia de esta medida de seguridad, es importante verificar que el sitio web utilice este encabezado HTTP, para garantizar que todos los datos viajan encriptados desde el navegador al servidor.

La función HTTP Strict Transport Security (HSTS) permite a una aplicación web informar al navegador, mediante el uso de un encabezado de respuesta especial, que nunca debe establecer una conexión con los servidores de dominio especificados mediante HTTP. En su lugar debe establecer automáticamente todas las solicitudes de conexión para acceder al sitio a través de HTTPS. El encabezado HTTP Strict Transport Security utiliza dos directivas:

- ✓ Max-age: para indicar el número de segundos en el que el navegador debe convertir automáticamente todas las solicitudes de HTTP a HTTPS.
- ✓ IncludeSubDomains: para indicar que todos los subdominios de la aplicación web deben utilizar HTTPS.

7.2.8 Pruebe la política de dominio cruzado RIA (OTG-CONFIG-008)

Descripción

Aplicaciones Enriquecidas de Internet (RIA) han adoptado los archivos de políticas de Adobe crossdomain.xml para permitir el acceso controlado de dominio cruzado para consumo de datos y servicios, utilizando tecnologías como Oracle Java, Silverlight y Adobe Flash. Por lo tanto, un dominio puede conceder acceso remoto a sus servicios desde un dominio diferente. Sin embargo, a menudo los archivos de políticas que describen las restricciones de acceso se configuran pobremente. Una configuración pobre de los archivos de directivas permite ataques de Cross-site Request Forgery (Falsificación de peticiones de sitios cruzados) y puede permitir a terceros acceder a los datos sensibles para el usuario.

Un archivo de políticas de dominios cruzados especifica los permisos que un cliente web como Java, Adobe Flash, Adobe Reader, etc. utilizan para acceder a datos entre dominios diferentes. Para Silverlight, Microsoft adoptó un subconjunto del crossdomain.xml de Adobe y, además, creó su propio archivo de directivas entre dominios: clientaccesspolicy.xml. Cada vez que un cliente web detecta que un recurso tiene que ser solicitado a otro dominio, primero buscará un archivo de políticas en el dominio de destino para determinar si puede realizar peticiones de dominio cruzado, incluyendo encabezados, y si se permiten los enlaces basados en tomas de conexión (socket-based connections).

Los archivos maestros de políticas se encuentran en la raíz del dominio. Un cliente puede recibir instrucciones para cargar un archivo de políticas diferentes, pero siempre comprobará el archivo maestro de política primero, para asegurarse de que el archivo maestro de políticas permite el archivo de políticas solicitado.

Herramientas

- ✓ Nikto.
- ✓ OWASP Zed Attack Proxy Project.
- ✓ W3af.

7.3 Pruebas de gestión de identidad

7.3.1 Pruebe las definiciones de roles (OTG-IDENT-001)

Descripción

Es común en las empresas modernas definir funciones de sistema para la gestión de usuarios y autorización de recursos del sistema. En la mayoría de las implementaciones de sistema, se espera que existan al menos dos funciones: los administradores y usuarios regulares. El primero representa un papel que permite el acceso a la funcionalidad privilegiada e información sensible; el segundo que representa un papel que permite el acceso a información y funcionalidad del negocio regular. Los roles bien desarrollados deben estar alineados con los procesos de negocio que están soportados por la aplicación.

Es importante recordar que la autorización obligatoria no es la única manera de gestionar el acceso a los objetos del sistema. En entornos más confiables, donde la confidencialidad no es crítica, controles más suaves como el flujo de trabajo de aplicación y registro de auditoría pueden cubrir los requisitos de integridad de los datos, mientras no restrinjan el acceso del usuario a la funcionalidad o la creación de estructuras de roles más complejas, que son difíciles de manejar.

Es importante considerar el principio de "Ricitos de Oro" cuando hablamos de la ingeniería de roles. Definir muy pocos papeles y amplios papeles (exponiendo a los usuarios de esta manera al acceso de funcionalidades que no requieren) es tan malo como muchos papeles o hechos muy ajustados a la medida de cada usuario (y así restringir el acceso de los usuarios a las funcionalidades que requieren).

Objetivo de prueba

Validar los diferentes roles en el sistema, definidos dentro de la aplicación. Defina y separe lo suficiente cada sistema y rol de negocio para gestionar un acceso adecuado a la información y funcionalidad del sistema.

Herramientas

Si bien el enfoque más exhaustivo y exacto para completar esta prueba es llevarla a cabo manualmente, las herramientas de spidering también son útiles. Inicie la sesión con cada rol en orden (no olvide excluir el vínculo cerrar sesión desde la herramienta de spidering).

Remediación

La remediación de los problemas puede tomar las siguientes formas:

- ✓ Ingeniería de roles.
- ✓ Crear mapas de los roles del negocio a los roles del sistema.
- ✓ Separación de funciones.

7.3.2 Pruebe el proceso de registro del usuario (OTG-IDENT-002)

Descripción

Algunos sitios web ofrecen un proceso de registro del usuario, que automatiza (o semi-automatiza) la creación del acceso al sistema para los usuarios. Los requisitos de identidad para el acceso varían desde identificación positiva a ninguna en absoluto, dependiendo de los requisitos de seguridad del sistema. Muchas aplicaciones públicas automatizan totalmente el registro y el proceso de provisionamiento porque el tamaño de la base de usuarios hace que sea imposible manejarla manualmente. Sin embargo, muchas aplicaciones corporativas provisionan usuarios manualmente, por lo que este tipo de prueba puede no ser aplicable.

Objetivo de la prueba

- ✓ Verifique que los requisitos de identidad para registro de usuarios estén alineados con los requerimientos de seguridad y negocio.
- ✓ Valide el proceso de registro.

Herramientas

Un proxy HTTP puede ser una herramienta útil para probar este control.

Remediación

Implemente una identificación y verificación de requisitos que corresponden a los requisitos de seguridad de la información que las credenciales protegen.

7.3.3 Pruebe el proceso de creación de cuentas (OTG-IDENT-003)

Descripción

El aprovisionamiento de cuentas presenta una oportunidad para un atacante de crear una cuenta válida sin la aplicación de una correcta identificación y proceso de autorización.

Objetivo de la prueba

Verifique qué cuentas pueden aprovisionar otras cuentas y de qué tipo.

Herramientas

Aunque el enfoque más exhaustivo y exacto para completar esta prueba es llevarla a cabo manualmente, las herramientas de proxy HTTP también pueden ser útiles.

7.3.4 Pruebas de enumeración de cuentas y adivinanza de cuentas de usuario (OTG-IDENT-004)

Descripción

La visión de esta prueba es verificar si es posible reunir un conjunto de nombres válidos de usuario interactuando con el mecanismo de autenticación de la aplicación. Esta prueba será útil para las pruebas de fuerza

bruta, en las que el evaluador verifica si, dado un nombre de usuario válido, es posible encontrar la contraseña correspondiente.

A menudo, las aplicaciones web revelan cuándo existe un nombre de usuario en el sistema, ya sea como consecuencia de la mala configuración o como una decisión de diseño. Por ejemplo, a veces, cuando se envían credenciales equivocadas, recibimos un mensaje que indica que el nombre de usuario está presente en el sistema o la contraseña proporcionada es incorrecta. La información obtenida puede utilizarla un atacante para obtener una lista de los usuarios en el sistema. Esta información puede utilizarse para atacar la aplicación web, por ejemplo, a través de un ataque forzado o un ataque por defecto de nombre de usuario y contraseña.

Herramientas

- ✓ WebScarab: OWASP_WebScarab_Project
- ✓ CURL.
- ✓ PERL.

Remediación

Asegúrese de que la aplicación devuelve mensajes de error genéricos, consistentes en respuesta a nombres de cuenta válidos, contraseñas u otras credenciales de usuario, ingresados durante el proceso de registro. Asegúrese que las cuentas de pruebas del sistema y cuentas por defecto se eliminen antes de lanzar el sistema a producción (o exponiéndola a una red no confiable).

7.3.5 Pruebe las políticas de nombre de usuario débiles o no segura (OTG-IDENT-005)

Descripción

Los nombres de usuario de cuentas están a menudo altamente estructurados (por ejemplo, el nombre de la cuenta de Joe Bloggs es jbloggs y el nombre de la cuenta de Fred Nurks es fnurks) y los nombres de cuentas válidos pueden ser adivinados fácilmente.

Objetivo de la prueba

Determine si una estructura de nombres de cuenta constante hace que la aplicación sea vulnerable a la enumeración de la cuenta. Determine si los mensajes de error de la aplicación permiten la enumeración de la cuenta.

Remediación

Asegúrese que la aplicación devuelva mensajes de error genéricos consistentes en respuesta a nombres de cuenta inválidos, contraseñas u otras credenciales de usuario introducidas durante el proceso registro.

7.4 Pruebas de autenticación

7.4.1 Pruebas del transporte de credenciales en una canal encriptado (OTG-AUTHN-001)

Descripción

Probar el transporte de credenciales significa comprobar que los datos de autenticación del usuario se transfieren a través de un canal encriptado para evitar ser interceptados por usuarios maliciosos. El análisis se centra simplemente en tratar de entender si los datos viajan sin encriptar desde el navegador web al servidor, o si la aplicación web toma las medidas de seguridad apropiadas al usar protocolos como HTTPS. El protocolo HTTPS se construye sobre TLS/SSL para encriptar los datos que se transmiten y asegurar que el usuario es enviado hacia el sitio deseado. Claramente, el hecho de que el tráfico se encuentre cifrado no necesariamente significa que es totalmente seguro. La seguridad depende también del algoritmo utilizado y la robustez de las claves que la aplicación está utilizando.

En la actualidad, el ejemplo más común de este problema es la página de registro en una aplicación web. El evaluador debe comprobar que las credenciales del usuario se transmitan a través de un canal encriptado. Para iniciar una sesión en un sitio web, el usuario generalmente tiene que llenar un formulario simple que transmite los datos insertados a la aplicación web con el método POST. Lo que es menos evidente es que estos datos se pueden

transmitir mediante el protocolo HTTP, que transfiere los datos de una manera insegura, como texto transparente, o utilizando el protocolo HTTPS, que cifra los datos durante la transmisión.

Para complicar más las cosas, existe la posibilidad de que el sitio tenga la página de inicio accesible a través de HTTP (haciéndonos creer que la transmisión es insegura), pero en realidad envía datos a través de HTTPS. Esta prueba se hace para asegurarse que un atacante no pueda recuperar información sensible simplemente husmeando en la red con una herramienta de olfateo (sniffer).

Herramientas

- ✓ WebScarab.
- ✓ OWASP Zed Attack Proxy (ZAP).

7.4.2 Pruebas de credenciales por defecto (OTG-AUTHN-002)

Descripción

En la actualidad, las aplicaciones web a menudo hacen uso de software popular de código abierto o comercial que puede ser instalado en servidores con configuración mínima o personalización para requisitos particulares del administrador del servidor. Por otra parte, muchos dispositivos de hardware (routers de red y servidores de base de datos) ofrecen configuración basada en web o interfaces administrativas.

A menudo estas aplicaciones, una vez instaladas, no están configuradas correctamente y las credenciales predeterminadas proporcionadas para la autenticación inicial y configuración nunca son cambiadas. Estas credenciales predeterminadas son bien conocidas por los evaluadores de penetración y, por desgracia, también por atacantes maliciosos que pueden utilizarlas para tener acceso a varios tipos de aplicaciones.

Además, en muchas situaciones, cuando se crea una nueva cuenta

en una aplicación, una contraseña por defecto (con algunas características estándar) se genera. Si esta contraseña es predecible y el usuario no la cambia en el primer acceso, esto puede llevar a un atacante a obtener acceso no autorizado a la aplicación.

La causa principal de este problema puede ser identificada como:

- ✓ Personal técnico sin experiencia que no es consciente de la importancia de cambiar las contraseñas por defecto en componentes de la infraestructura instalada o dejar la contraseña por defecto para "facilidad de mantenimiento".
- ✓ Programadores que dejan puertas traseras para tener fácil acceso y probar su aplicación y después olvidan eliminarlas.
- ✓ Aplicaciones con cuentas predeterminadas fijas incorporadas con un usuario y contraseña predefinido.
- ✓ Aplicaciones que no obligan al usuario a cambiar las credenciales por defecto después de la primera sesión.

Herramientas

- ✓ Burp Intruder.
- ✓ THC Hydra.
- ✓ Brutus.
- ✓ Nikto 2.

7.4.3 Pruebas para determinar un mecanismo de bloqueo débil (OTG-AUTHN-003)

Descripción

Los mecanismos de bloqueo se utilizan para mitigar los ataques de fuerza bruta o adivinanza de contraseñas. Las cuentas se bloquean normalmente después de tres a cinco intentos de inicio de sesión sin éxito y solo pueden ser desbloqueadas después de un periodo determinado de tiempo, a través de la intervención de un administrador. Los mecanismos de bloqueo de cuenta requieren un equilibrio entre la protección de cuentas de acceso no autorizado y proteger a los usuarios de una negativa al acceso

autorizado.

Objetivo de la prueba

- ✓ Evaluar la capacidad del mecanismo de bloqueo de cuentas para mitigar el ingreso forzado adivinando contraseñas.
- ✓ Evaluar la resistencia del mecanismo de liberación para abrir sin autorización la cuenta.

Remediación

Aplique mecanismos de desbloqueo de cuentas dependiendo del nivel de riesgo. En orden de menor a mayor seguridad:

- ✓ Bloqueo y desbloqueo temporizado.
- ✓ Desbloqueo con autoservicio (desbloqueo que envía un correo electrónico a la dirección de correo electrónico registrada).
- ✓ Desbloqueo manual por un administrador.
- ✓ Desbloqueo manual por un administrador con identificación de usuario positiva.

7.4.4 Pruebas para eludir el esquema de autenticación (OTG-AUTHN-004)

Descripción

Mientras que la mayoría de las aplicaciones requieren autenticación para tener acceso a información privada o para ejecutar las tareas, no todos los métodos de autenticación son capaces de proporcionar una seguridad adecuada. La negligencia, ignorancia o una simple subvaloración de las amenazas de seguridad, a menudo resultan en esquemas de autenticación que pueden evitarse simplemente obviando el registro en la página y llamando directamente a una página interna que se supone debe accederse sólo después de que se realizó la autenticación.

Además, a menudo es posible sortear las medidas de autenticación alterando las solicitudes y engañando a la aplicación a pesar de que el usuario

ya está autenticado. Esto se puede lograr modificando el parámetro de URL determinado, mediante la manipulación de la forma o por falsificación de las sesiones. Los problemas relacionados con el esquema de autenticación pueden encontrarse en diferentes etapas del ciclo de vida de desarrollo de software (SDLC), como las fases de diseño, desarrollo e implementación:

- ✓ En los errores de la fase de diseño, se puede incluir una definición equivocada de las secciones de la aplicación a proteger, la opción de no aplicar protocolos de encriptación fuertes para asegurar la transmisión de las credenciales y muchos más.
- ✓ En los errores de la fase de desarrollo, se puede incluir la aplicación incorrecta de la funcionalidad de validación de entrada o sin seguir las mejores prácticas de seguridad para el idioma específico.
- ✓ En la fase de implementación de la aplicación, puede haber problemas durante la instalación de la aplicación (actividades de instalación y configuración) debido a la falta de habilidades técnicas requeridas o por falta de una buena documentación.

Herramientas

- ✓ WebScarab.
- ✓ WebGoat.
- ✓ OWASP Zed Attack Proxy (ZAP).

7.4.5 Pruebas para recordatorios de contraseñas vulnerables (OTG-AUTHN-005)

Descripción

Los navegadores a veces preguntarán al usuario si desea recordar la contraseña que acaba de ingresar. El navegador entonces almacenará la contraseña e ingresará automáticamente cada vez que el mismo formulario de autenticación sea visitado. Esto es una conveniencia para el usuario. Además, algunos sitios web ofrecen funcionalidades personalizadas de

"Recuérdame" para permitir que los usuarios mantengan su sesión en un sistema de cliente específico.

Tener al navegador guardando contraseñas no es sólo conveniente para los usuarios finales, sino también para un atacante. Si un atacante puede tener acceso al navegador de la víctima (por ejemplo, a través de un ataque de Cross Site Scripting, o a través de un ordenador compartido), pueden recuperar las contraseñas almacenadas. No es extraño que los navegadores almacenen las contraseñas de una manera fácilmente recuperable, sino que, incluso, si el navegador almacena contraseñas encriptadas que sólo pueden ser recuperadas mediante el uso de una contraseña maestra, un atacante podría recuperar la contraseña visitando el formulario de autenticación de la aplicación web de destino, introducir el usuario de la víctima, y dejar que el navegador introduzca la contraseña.

Remediación

Asegúrese que ninguna credencial sea almacenada en texto claro, o que sean fácilmente recuperables en forma codificada o encriptada en cookies.

7.4.6 Pruebas para buscar la debilidad de memoria caché en el navegador (OTG-AUTHN-006)

Descripción

En esta fase el evaluador comprueba que la aplicación indique correctamente al navegador que no recuerde datos sensibles. Los navegadores pueden almacenar información con fines de almacenamiento en caché e historia. El almacenamiento en caché se utiliza para mejorar el rendimiento; así la información que apareció previamente no necesita descargarse otra vez. Se utilizan mecanismos de historia para conveniencia del usuario, por lo que él puede ver exactamente lo que vio en el momento de obtener el recurso.

Si se muestra información sensible al usuario (como su dirección,

datos de tarjeta de crédito, número de seguro social o usuario), esta información podría ser almacenada con fines de almacenamiento en caché o de historia y, por lo tanto, ser recuperables examinando la caché del navegador o pulsando el botón "Atrás" del navegador.

Herramientas

- ✓ OWASP Zed Attack Proxy.
- ✓ Firefox add-on CacheViewer2.

7.4.7 Pruebas para determinar las políticas de contraseñas débiles (OTG-AUTHN-007)

Descripción

El mecanismo de autenticación más frecuente y más fácilmente administrado es una contraseña estática. La contraseña representa las llaves del dominio, pero a menudo es devaluada por los usuarios en nombre de la facilidad de uso. En cada uno de los últimos hacks de alto perfil que han revelado las credenciales de usuario, se lamenta que las contraseñas más comunes son: 123456, password y qwerty.

Objetivo de la prueba

Determine la resistencia de la aplicación contra ataques de fuerza bruta o adivinanza de contraseña usando diccionarios de contraseñas disponibles mediante la evaluación de los requerimientos de longitud, complejidad, reutilización y caducidad de las contraseñas.

Remediación

Para mitigar el riesgo de contraseñas fácilmente adivinables facilitando el acceso no autorizado, hay dos soluciones: introducir controles de autenticación adicionales (es decir, autenticación de dos factores) o introducir una política de contraseñas fuertes. El más simple y más barato de estos es la introducción de una política de contraseña fuerte que asegura la longitud, la complejidad, la reutilización y la caducidad de la contraseña.

7.4.8 Pruebas para determinar la seguridad débil de pregunta y respuesta (OTG-AUTHN-008)

Descripción

A menudo llamadas preguntas y respuestas "secretas", las preguntas y respuestas de seguridad se utilizan frecuentemente para recuperar contraseñas olvidadas (véase Pruebas para determinar un cambio débil de contraseña o funciones de restablecimiento (OTG-AUTHN-009)), o como seguridad adicional por encima de la contraseña.

Típicamente se generan con la creación de la cuenta y requieren que el usuario seleccione algunas de las preguntas previamente generadas y provea una respuesta adecuada. Puede permitir al usuario generar sus propios pares de preguntas y respuestas. Ambos métodos son propensos a inseguridades. Idealmente, las preguntas de seguridad deben generar respuestas que sólo son conocidas por el usuario y no pueden ser predichas o descubiertas por nadie más. Esto es más difícil de lo que suena.

Las preguntas y respuestas de seguridad se basan en el secreto de la respuesta. Las preguntas y respuestas deben elegirse de modo que las respuestas son sólo conocidas por el titular de la cuenta. Sin embargo, aunque muchas respuestas no sean públicamente conocidas, la mayoría de las preguntas que implementan los sitios web promueven respuestas que son de carácter privado.

Preguntas previamente generadas

La mayoría de preguntas previamente generadas son de naturaleza bastante simple y pueden llevar a respuestas inseguras. Por ejemplo:

- ✓ Las respuestas pueden ser conocidas por los familiares o amigos cercanos del usuario, por ejemplo, "¿Cuál es el apellido de soltera de su madre?", "¿Cuál es su fecha de nacimiento?".
- ✓ Las respuestas pueden ser fácilmente predecibles, e.g. "¿Cuál es su color favorito?", "¿Cuál es su equipo favorito de béisbol?".
- ✓ Las respuestas pueden ser atacadas con fuerza bruta, por

ejemplo, "¿Cuál es el nombre de su profesora favorita de secundaria?" La respuesta está probablemente en alguna lista fácilmente descargable de nombres populares y, por lo tanto, un ataque de fuerza bruta simple puede secuenciarse en un script.

- ✓ Las respuestas pueden ser públicamente visibles, por ejemplo, ¿cuál es su película favorita?" La respuesta puede encontrarse fácilmente en la página de perfil de redes sociales del usuario.

Preguntas generadas por el usuario

El problema de pedir a los usuarios que generen sus propias preguntas es que les permite generar interrogantes muy inseguras, o incluso desviar la idea de tener una pregunta de seguridad en primer lugar. Aquí están algunos ejemplos del mundo real que ilustran este punto:

- ✓ “Cuánto es 1+1?”.
- ✓ “¿Cuál es tu nombre de usuario?”.
- ✓ “Mi clave es M3@t$p1N”.

7.4.9 Pruebas para determinar un cambio débil de contraseña o funciones de restablecimiento (OTG-AUTHN-009)

Descripción

El cambio de contraseña y la función de restablecimiento de una aplicación es un autoservicio de cambio de contraseña o un mecanismo de restablecimiento para los usuarios. Este mecanismo de autoservicio permite a los usuarios cambiar o restablecer rápidamente la contraseña sin que un administrador intervenga. Cuando se cambian las contraseñas se cambian típicamente dentro de la aplicación. Cuando las contraseñas se restablecen son presentadas dentro de la aplicación o por correo electrónico al usuario. Esto puede indicar que las contraseñas se almacenan en texto plano o formato desencriptable.

Objetivo de la prueba

- ✓ Determine la resistencia de la aplicación a la subversión del proceso de cambio de la cuenta permitiendo a una persona cambiar la contraseña de una cuenta.
- ✓ Determine la resistencia de la función de restablecimiento de contraseñas contra el que puedan eludir o adivinar.

Remediación

El cambio de contraseña o función de restablecimiento es una función sensible y requiere algún tipo de protección, tal como que los usuarios tengan que volver a autenticarse o que se presente al usuario pantallas de confirmación durante el proceso.

7.4.10 Pruebas para determinar la autenticación más débil en un canal alternativo (OTG-AUTHN-010)

Descripción

Incluso si los mecanismos de autenticación primaria no incluyen vulnerabilidades, puede ser que las vulnerabilidades existan en canales alternativos de autenticación legítima para la misma cuenta del usuario. Deben realizarse pruebas para identificar canales alternativos y, conforme a la prueba de evaluación, identificar las vulnerabilidades.

Los canales alternativos de interacción del usuario podrían utilizarse para eludir el canal primario o exponer información que puede utilizarse para ayudar a un ataque contra el canal primario. Algunos de estos canales pueden ser aplicaciones web independientes, mediante nombres o rutas de alojamiento diferentes. Por ejemplo:

- ✓ Página web estándar.
- ✓ Sitio web optimizado para dispositivos móviles o dispositivos específicos.
- ✓ Sitio web de accesibilidad optimizada.
- ✓ Sitios web de país e idioma alternativos.

- ✓ Sitios web paralelos que utilizan el mismo usuario (por ejemplo, otra página web que ofrece diferentes funcionalidades de la misma organización, un sitio web de un socio con el que se comparten cuentas de usuario).
- ✓ Desarrollo, prueba, UAT (Prueba de aceptación del usuario) y puesta en escena de las versiones de la página web estándar.

Pero también podría haber otro tipo de aplicaciones o procesos del negocio:

- ✓ Aplicación para dispositivos móviles.
- ✓ Aplicación de escritorio.
- ✓ Operadores de centros de llamadas (call center).
- ✓ Sistemas de respuesta de voz interactiva o sistemas de árboles de llamadas telefónicas.

Remediación

Asegúrese de que se aplique una política de autenticación consistente en todos los canales para que sean igualmente seguros.

7.5 Pruebas de autorización

7.5.1 Prueba transversal de directorio de archivos incluidos (OTG-AUTHZ-001)

Descripción

Muchas aplicaciones web usan y administran archivos como parte de su operación diaria. Usando métodos de validación de entrada que no han sido bien diseñados o implementados, un agresor podría aprovechar el sistema para leer o escribir archivos que no están diseñados para ser accesibles. En situaciones particulares, podría ser posible ejecutar un código arbitrario o comandos del sistema.

Tradicionalmente, los servidores web y aplicaciones web implementan mecanismos de autenticación para controlar el acceso a archivos y recursos. Los servidores web tratan de limitar los archivos de los usuarios dentro de un

"directorio raíz" o "raíz del documento web ", que representa un directorio físico en el sistema de archivos. Los usuarios deben considerar este directorio como el directorio base en la estructura jerárquica de la aplicación web.

La definición de los privilegios se realiza mediante listas de control de acceso (ACL) que identifican qué usuarios o grupos se supone que deben tener acceso, modificar o ejecutar un archivo en el servidor. Estos mecanismos están diseñados para evitar que usuarios malintencionados tengan acceso a archivos sensibles (por ejemplo, el archivo común /etc/passwd en una plataforma tipo UNIX) o para evitar la ejecución de comandos del sistema.

Muchas aplicaciones web utilizan secuencias de comandos de servidor para incluir diferentes tipos de archivos. Es muy común utilizar este método para administrar imágenes, plantillas, cargar textos estáticos y así sucesivamente. Desafortunadamente, estas aplicaciones exponen las vulnerabilidades de seguridad si los parámetros de entrada (es decir, parámetros de los formularios, valores de cookies) no son validados correctamente.

Herramientas

- ✓ DotDotPwn.
- ✓ Path Traversal Fuzz Strings.
- ✓ Web Proxy (Burp Suite, Paros, WebScarab, Zed Attack Proxy (ZAP)).
- ✓ Enconding/Decoding tools.
- ✓ String searcher “grep”.

7.5.2 Prueba para eludir el esquema de autorización (OTG-AUTHZ-002)

Descripción

Este tipo de prueba se centra en comprobar cómo se implementó el esquema de autorización para que cada rol o privilegio obtenga acceso a funciones reservadas y recursos. Para cada rol específico que el evaluador

tiene durante la evaluación, para cada función y solicitud que la aplicación ejecuta durante la fase posterior a la autenticación, es necesario verificar:

- ✓ ¿Es posible acceder a ese recurso, incluso si el usuario no está autenticado?
- ✓ ¿Es posible tener acceso a ese recurso después de la desconexión?
- ✓ ¿Es posible acceder a las funciones y los recursos que deben ser accesibles a un usuario que tiene un rol diferente o un privilegio?

Intente acceder a la aplicación como un usuario administrativo y siga todas las funciones administrativas.

- ✓ ¿Es posible acceder a funciones administrativas si el evaluador está registrado como un usuario con privilegios estándar?
- ✓ ¿Es posible utilizar estas funciones administrativas como un usuario con un rol diferente y para quien esa acción debería ser negada?

Herramientas

- ✓ OWASP WebScarab: OWASP WebScarab Project.
- ✓ OWASP Zed Attack Proxy (ZAP).

7.5.3 Pruebas para determinar el escalamiento de privilegios (OTG-AUTHZ-003)

Descripción

Esta sección describe el problema del escalamiento de privilegios de una etapa a otra. Durante esta fase, el evaluador deberá verificar que no es posible para un usuario modificar sus privilegios o roles dentro de la aplicación, de manera que podría permitir ataques de escalada de privilegios.

El escalamiento de privilegios se produce cuando un usuario obtiene acceso a más recursos o funcionalidad que la que normalmente se le permite,

y dicha elevación o cambios debían haber sido evitados por la aplicación. Esto es generalmente causado por un defecto en la aplicación. El resultado es que la aplicación realiza las acciones con más privilegios que los que el desarrollador o administrador del sistema querían adjudicar.

El grado de escalamiento depende de los privilegios que el atacante está autorizado a poseer y que se pueden obtener en una explotación exitosa. Por ejemplo, un error de programación que permite a un usuario privilegios extras después de la autenticación correcta limita el grado de escalamiento, porque el usuario ya está autorizado a tener algo de privilegios. Asimismo, un atacante remoto que obtiene privilegios de superusuario sin ninguna autenticación presenta un mayor grado de escalamiento.

Generalmente, las personas se refieren al escalamiento vertical cuando es posible tener acceso a recursos en cuentas más privilegiadas (por ejemplo, adquirir privilegios administrativos para la aplicación) y en escalamiento horizontal cuando es posible acceder a los recursos de una cuenta configurada de manera similar (por ejemplo, en una aplicación de banca en línea, al acceder a la información relacionada con un usuario diferente).

Herramientas

- ✓ OWASP WebScarab: OWASP WebScarab Project.
- ✓ OWASP Zed Attack Proxy (ZAP).

7.5.4 Pruebas de las referencias de objetos directos inseguros (OTG-AUTHZ-004)

Descripción

Las Referencias de Objetos Directos Inseguros ocurren cuando una aplicación ofrece acceso directo a objetos basados en la información suministrada por el usuario. Como resultado de esta vulnerabilidad, los atacantes pueden omitir la autorización y acceder a recursos en el sistema directamente, por ejemplo, registros de la base de datos o archivos.

Las Referencias de Objetos Directos Inseguros permiten a los atacantes omitir la autorización y acceder a los recursos modificando directamente el valor de un parámetro para apuntar directamente a un objeto. Tales recursos pueden ser entradas de bases de datos que pertenecen a otros usuarios, archivos en el sistema y más. Esto es causado porque la aplicación toma la información ingresada por el usuario y la utiliza para recuperar un objeto sin realizar las comprobaciones de autorización suficientes.

Herramientas

- ✓ OWASP WebScarab: OWASP WebScarab Project.
- ✓ OWASP Zed Attack Proxy (ZAP).

7.6 Pruebas de la gestión de sesión

Uno de los componentes básicos de cualquier aplicación basada en la web es el mecanismo que controla y mantiene el estado de un usuario para interactuar con él. Esto se refiere a aquello como gestión de sesiones y se define como el conjunto de todos los controles que rigen el estado completo de interacción entre un usuario y la aplicación basada en la web. Esto cubre ampliamente cualquier cosa, desde cómo se realiza la autenticación del usuario, a lo que sucede al salir del sistema.

HTTP es un protocolo sin estado, lo que significa que los servidores web responden a solicitudes de clientes sin vincularlos entre sí. Incluso la lógica de la aplicación más simple requiere que múltiples solicitudes de un usuario se asocien entre sí dentro de una "sesión". Esto requiere soluciones de terceros a través de cualquier middleware estándar, o con una implementación de un desarrollador hecha a la medida. Los más populares entornos de aplicaciones web, como ASP y PHP, proporcionan a los desarrolladores rutinas de sesión incorporada. Algún tipo de ficha de identificación normalmente se emitirá, y se denominará "Identificador de sesión" o Cookie.

Hay numerosas maneras en que una aplicación web puede interactuar con un usuario. Cada una depende de la naturaleza de los requerimientos del

sitio, la seguridad y la disponibilidad de la aplicación. Mientras hayan aceptado las mejores prácticas para desarrollo de aplicaciones, tales como las descritas en la guía de OWASP Construyendo Aplicaciónes Web Seguras, es importante que se considere la seguridad de las aplicaciones en el contexto de las necesidades y expectativas del proveedor.

7.6.1 Pruebas del esquema de gestión de sesión (OTG-SESS-001)

Descripción

Para evitar la autenticación continua para cada página de un sitio web o servicio, las aplicaciones web implementan diversos mecanismos para almacenar y validar las credenciales durante un intervalo de tiempo determinado. Estos mecanismos se conocen como manejo de sesiones y aunque son importantes para aumentar la facilidad del uso de la aplicación y amigables con el usuario, pueden ser aprovechados por un evaluador de penetración para obtener acceso a una cuenta de usuario, sin necesidad de proporcionar las credenciales correctas.

En esta prueba, el evaluador debe comprobar que las cookies y otras fichas de sesión se crean de una manera segura e impredecible. Un atacante que es capaz de predecir y falsificar una cookie débil, fácilmente puede secuestrar las sesiones de usuarios legítimos.

Las cookies se utilizan para implementar la gestión de la sesión y se describen en detalle en el RFC 2965. En resumen, cuando un usuario accede a una aplicación que necesita hacer seguimiento de las acciones y la identidad de ese usuario a través de múltiples peticiones, una cookie (o cookies) son generadas por el servidor y enviadas al cliente. El cliente enviará la cookie al servidor en todas las conexiones siguientes hasta que esta expire o se destruya. Los datos almacenados en la cookie pueden proporcionar al servidor un gran abanico de información sobre quién es el usuario, qué acciones ha realizado hasta ahora, cuáles son sus preferencias, etc.; por lo tanto, le da un estado a un protocolo sin estado como es HTTP.

Un ejemplo típico es proporcionado por un carrito de compras en

línea. Durante toda la sesión de un usuario, la aplicación debe hacer un seguimiento de su identidad, su perfil, los productos que ha elegido para comprar, la cantidad, los precios individuales, descuentos, etc. Las cookies son una manera eficiente de almacenar y transmitir esta información una y otra vez (otros métodos son parámetros de URL y campos ocultos).

Debido a la importancia de los datos que almacenan, las cookies son vitales para la seguridad general de la aplicación. Poder manipular las cookies puede resultar en un secuestro de las sesiones de usuarios legítimos, obtener mayores privilegios en una sesión activa y, en general, influir en las operaciones de la aplicación de una manera no autorizada.

En esta prueba, el evaluador podrá comprobar si las cookies emitidas a los clientes pueden resistirse a una amplia gama de ataques dirigidos a interferir con las sesiones de usuarios legítimos y con la propia aplicación. El objetivo general es ser capaces de falsificar una cookie que sea considerada válida por la aplicación y que proporcione algún tipo de acceso no autorizado (secuestro de sesión, escalada de privilegios, etc).

Generalmente, los pasos principales del patrón de ataque son los siguientes:

- ✓ Recolección de cookies: recolectar un número suficiente de muestras de cookies.
- ✓ Ingeniería inversa de cookies: análisis del algoritmo de generación de cookies.
- ✓ Manipulación de cookies: falsificar una cookie válida para llevar a cabo el ataque. Este último paso podría requerir un gran número de intentos, dependiendo de cómo la cookie haya sido creada (ataque de fuerza bruta de cookie).

Herramientas

- ✓ OWASP Zed Attack Proxy Project (ZAP): owasp.org - features a session token analysis mechanism.
- ✓ Burp Sequencer: portswigger.net
- ✓ Foundstone CookieDigger: mcafee.com

- ✓ YEHG's JHijack: owasp.org

7.6.2 Pruebas de los atributos de las cookies (OTG-SESS-002)

Descripción

Las cookies son a menudo un vector de ataque clave para usuarios maliciosos (normalmente dirigidas hacia otros usuarios) y la aplicación siempre debe tomar las debidas diligencias para proteger las cookies. Esta sección examina cómo una aplicación puede tomar las precauciones necesarias al asignar las cookies y cómo se prueba que estos atributos se han configurado correctamente.

La importancia del uso seguro de las cookies no puede estar subestimada, especialmente en aplicaciones web dinámicas, que necesitan mantener el estado a través de un protocolo sin estado como HTTP. Para entender la importancia de las cookies, es imprescindible entender para qué se usan principalmente. Dentro de las funciones primarias se encuentran, generalmente, las de ser utilizadas como una autorización de sesión y ficha de autenticación o como un contenedor de datos temporales. Así, si un atacante pudiera adquirir una ficha de sesión (por ejemplo, mediante la explotación de una vulnerabilidad en un script de sitios cruzados), entonces podría utilizar esta cookie para secuestrar una sesión válida.

Además, las cookies se establecen para mantener el estado entre varias solicitudes. Dado que HTTP no tiene estado, el servidor no puede determinar si una solicitud que recibe es parte de una sesión actual o el comienzo de una nueva sesión sin ningún tipo de identificador. Este identificador es muy comúnmente una cookie, aunque también son posibles otros métodos. Hay muchos tipos diferentes de aplicaciones que necesitan hacer un seguimiento del estado de la sesión a través de múltiples solicitudes. La primera que viene a la mente es una tienda online.

Una vez que el evaluador tiene una comprensión de cómo se establecen las cookies, cuándo se configuran, para qué se utilizan, por qué se utilizan y su importancia, debería echar un vistazo a qué atributos se pueden

establecer en una cookie y cómo comprobar si son seguras. La siguiente es una lista de los atributos que se pueden establecer para cada cookie y lo que significan. La siguiente sección se centrará en cómo probar para cada atributo.

- ✓ Segura: este atributo indica al navegador que sólo envíe la cookie si la solicitud se remite mediante un canal seguro como HTTPS. Esto ayudará a proteger el envió de la cookie por solicitudes no encriptadas. Si se puede accede a la aplicación a través de HTTP y HTTPS, entonces existe la posibilidad de que la cookie pueda mandarse en texto limpio.
- ✓ HttpOnly: este atributo se utiliza para ayudar a prevenir ataques como crosssite scripting, ya que no permite que puedan acceder a la cookie a través de un script del lado del cliente como JavaScript. Tenga en cuenta que no todos los navegadores admiten esta funcionalidad.
- ✓ Dominio: este atributo se utiliza para comparar contra el dominio del servidor en el que se solicita la dirección URL. Si el dominio coincide o si es un subdominio, entonces el atributo de ruta de acceso se comprobará a continuación.

Herramientas

Intercepting Proxy:

- ✓ OWASP Zed Attack Proxy Project.

Browser Plug-in:

- ✓ “TamperIE” for Internet Explorer: bayden.com
- ✓ Adam Judson: “Tamper Data” for Firefox: addons.mozilla.org

7.6.3 Pruebas de fijación de sesión (OTG-SESS-003)

Descripción

Cuando una aplicación no renueva su(s) cookie(s) de sesión después de una autenticación exitosa, podría ser posible encontrar una vulnerabilidad

de fijación de sesión y forzar a un usuario a utilizar una cookie conocida por el atacante. En ese caso, un atacante podría robar la sesión del usuario (secuestro de sesión). Las vulnerabilidades de fijación de sesión ocurren cuando:

- ✓ Una aplicación web autentica a un usuario sin invalidar primero el ID de sesión existente, de tal modo que continúa usando el identificador de sesión ya asociado con el usuario.
- ✓ Un atacante es capaz de forzar un Identificador de sesión conocido sobre un usuario para que, una vez que el usuario se autentica, el atacante tenga acceso a la sesión autenticada.

En una explotación de vulnerabilidades genéricas de fijación de sesión, un atacante crea una nueva sesión en una aplicación web y registra el identificador de sesión asociado. El atacante, entonces, hace que la víctima se autentique en el servidor utilizando el mismo identificador de sesión, lo que da al atacante acceso a la cuenta del usuario a través de la sesión activa.

Además, el problema descrito anteriormente es difícil para los sitios que emiten un identificador de sesión sobre HTTP y luego redireccionan al usuario a un formulario de inicio de sesión en HTTPS. Si el identificador de sesión no es regenerado tras la autenticación, el atacante puede husmear y robar el identificador y luego usarlo para secuestrar la sesión.

Herramientas

- ✓ Hijacking tool.
- ✓ OWASP WebScarab.

7.6.4 Pruebas para determinar la exposición de las variables de sesión (OTG-SESS-004)

Descripción

Las fichas de sesión (Cookie, SessionID, Hidden Field), si se exponen, generalmente permitirán a un atacante hacerse pasar por una víctima y acceder a la aplicación ilegítimamente. Es importante que estén protegidas de

intrusos en todo momento, especialmente mientras están en tránsito entre el navegador del cliente y los servidores de las aplicaciones.

Esta información se refiere a cómo la seguridad en el transporte se aplica a la transferencia de datos sensibles del identificador de sesión en general, y puede ser más estricta que las políticas de transporte y almacenamiento en caché de los datos atendidos por el sitio. Utilizando un proxy interceptor, es posible conocer lo siguiente sobre cada petición y respuesta:

- ✓ Protocol used (e.g., HTTP vs. HTTPS).
- ✓ HTTP Headers.
- ✓ Message Body (e.g., POST or page content).

Cada vez que los datos del identificador de sesión pasan entre el cliente y el servidor, el protocolo, caché, las directivas y cuerpo de privacidad deben ser revisadas. La seguridad del transporte se refiere a la circulación del identificador de sesión por peticiones GET o POST, cuerpo de los mensajes u otros medios, mediante solicitudes HTTP válidas.

Para probar las vulnerabilidades de encriptación y reutilización de las fichas de sesión:

La protección contra intrusos es proporcionada a menudo por una encriptación SSL, pero puede incorporar otros túneles o encriptados. Cabe señalar que la encriptación o hash criptográfico del identificador de sesión debe considerarse por separado del encriptado del transporte, ya que es el identificador de sesión en sí el que se está protegiendo, no los datos que pueden ser representados por él.

Tenga en cuenta también que si hay un elemento en el sitio donde se realiza un seguimiento del usuario a través de un identificador de sesión, pero la seguridad no está presente (por ejemplo, observando qué documentos públicos descarga un usuario registrado), es esencial que se utilice un identificador de sesión diferente. El identificador de sesión, por lo tanto, debe

ser monitoreado mientras el usuario cambia entre elementos seguros y no seguros para garantizar que se utiliza uno diferente.

Resultado esperado

Cada vez que la autenticación es exitosa, el usuario debe esperar recibir:

- ✓ Una ficha de sesión diferente.
- ✓ Una ficha enviada a través de un canal encriptado cada vez que se hace una solicitud HTTP.

7.6.5 Pruebas de CSRF (OTG-SESS-005)

Descripción

CSRF es un ataque que obliga a un usuario a ejecutar acciones no deseadas en una aplicación web en la que actualmente él o ella se encuentra autenticado(a). Con un poco de ayuda de la ingeniería social (como enviar un enlace a través de un correo electrónico o chat), un atacante puede forzar a los usuarios de una aplicación web a ejecutar acciones escogidas por el atacante.

Un ataque CSRF exitoso puede comprometer los datos del usuario final y la operación cuando se dirige a un usuario normal. Si la cuenta de administrador es el usuario final objetivo, un ataque CSRF puede comprometer a toda la aplicación de web.

CSRF se basa en lo siguiente:

- ✓ [1] El comportamiento del navegador web con respecto al manejo de información relacionada con la sesión como las cookies y la información de autenticación http.
- ✓ [2] El conocimiento del atacante sobre urls válidos de aplicaciones web.
- ✓ [3] La gestión de sesión de la aplicación basada únicamente en

la información que es conocida por el navegador.

- ✓ [4] La existencia de etiquetas HTML cuya presencia permite un acceso inmediato a un recurso http(s) por ejemplo, la etiqueta de imagen img.

Los puntos 1, 2 y 3 son esenciales para que la vulnerabilidad esté presente, mientras que el punto 4 es un accesorio y facilita la explotación real, pero no es estrictamente necesario.

Herramientas

- ✓ OWASP ZAP.
- ✓ CSRF Tester.
- ✓ Cross Site Requester.
- ✓ Cross Frame Loader.
- ✓ Pinata-csrf-tool.

Remediación

Las siguientes defensas se dividen entre las recomendaciones para los usuarios y para los desarrolladores.

Usuarios

Ya que las vulnerabilidades CSRF son, al parecer, generalizadas, se recomienda seguir las mejores prácticas para mitigar el riesgo. Algunas acciones de mitigación son:

- ✓ Cierre la sesión inmediatamente después de usar una aplicación web.
- ✓ No permita que el navegador guarde el nombre de usuario/contraseñas y no permita que los sitios web "recuerden" la información de registro.
- ✓ No utilice el mismo navegador para acceder a aplicaciones sensibles y para navegar libremente por Internet. Si es necesario, haga ambas cosas en la misma máquina y con distintos navegadores.

Tanto el correo/navegador como los entornos de lector/navegador integrados en HTML plantean riesgos adicionales, puesto que simplemente ver un mensaje de correo o un mensaje de noticias puede conducir a la ejecución de un ataque.

Desarrolladores

Agregue información relacionada a la sesión a la URL. Lo que hace posible el ataque es el hecho de que la sesión es identificada como única por la cookie, que se envía automáticamente por el navegador. Tener otra información específica de la sesión que se genera a nivel de la URL dificulta al atacante conocer la estructura de las URL a atacar. Otras defensas, mientras no resuelvan el problema, contribuyen a hacer más difícil la explotación:

- ✓ Usar solicitudes POST en vez de solicitudes GET. Puesto que las solicitudes POST se pueden simular mediante JavaScript, esto hace que sea más complejo montar un ataque.
- ✓ Lo mismo sucede con páginas de confirmación intermedias (tales como: "¿está seguro que desea hacer esto?"). Estas pueden ser evitadas por un atacante, aunque harán su trabajo un poco más complejo. Por lo tanto, no dependa únicamente de estas medidas para proteger su aplicación.
- ✓ Los mecanismos de cierre automático de sesión mitigan, de alguna manera, la exposición a estas vulnerabilidades, aunque en última instancia depende del contexto (un usuario que trabaja todo el día en una aplicación web bancaria vulnerable tiene obviamente un mayor riesgo que un usuario que utiliza la misma aplicación ocasionalmente).

7.6.6 Pruebas de la funcionalidad del cierre de sesión(OTG-SESS-006)

Descripción

El cierre de sesión es una parte importante del ciclo de vida de la sesión. Reducir al mínimo la vida útil de las fichas de sesión disminuye la

probabilidad de un ataque de secuestro de sesión exitoso. Esto puede verse como un control contra la prevención de otros ataques como el Cross Site Scripting (CSS) o Cross Site Request Forgery (CSRF). Se conoce que este tipo de ataques dependen de que un usuario tenga una sesión autenticada en ese momento. No tener una terminación de sesión segura sólo aumenta la superficie de ataque para cualquiera de estos ataques.

Una terminación de sesión segura requiere, por lo menos, de los siguientes componentes:

- ✓ La disponibilidad de controles de interfaz de usuario que permiten al usuario desconectarse manualmente.
- ✓ La terminación de la sesión después de un período específico de tiempo sin actividad (tiempo de caducidad de sesión).
- ✓ Una correcta anulación del estado de la sesión desde el servidor.

Hay múltiples cuestiones que pueden prevenir la terminación eficaz de una sesión. Para la seguridad ideal de la aplicación web, un usuario debe ser capaz de terminar en cualquier momento a través de la interfaz de usuario. Cada página debe contener un botón de cierre de sesión en un lugar directamente visible. Las funciones de cierre de sesión confusas o ambiguas podrían causar desconfianza en el usuario sobre dicha funcionalidad.

Otro error común en la terminación de la sesión es que en la ficha de sesión del cliente se establece un nuevo valor, mientras que en el estado del servidor permanece activa y puede ser reutilizada restableciendo la cookie de sesión al valor anterior. A veces sólo se muestra un mensaje de confirmación al usuario sin que tenga que realizar ninguna acción adicional. Esto debe evitarse.

A los usuarios de navegadores web a menudo no les importa que una aplicación está todavía abierta y simplemente cierran el navegador o la pestaña. Una aplicación web debe considerar este comportamiento y terminar la sesión automáticamente del lado del servidor después de un tiempo

definido.

El uso de un sistema de registro único (single sign-on ó SSO) en lugar de un esquema de autenticación de aplicaciones específicas, a menudo causa la coexistencia de múltiples sesiones que deben terminarse por separado. Por ejemplo, la terminación de la sesión de la aplicación específica no termina la sesión en el sistema SSO. Navegar de vuelta hacia el portal SSO ofrece al usuario la posibilidad de volver a iniciar una sesión en la aplicación donde la sesión fue terminada poco antes. Por otro lado, una función de registro en un sistema SSO no necesariamente causa el cierre de sesión en las aplicaciones interconectadas.

Herramientas

- ✓ Burp Suite – Repeater.

7.6.7 Pruebas del tiempo de cierre de sesión (OTG-SESS-007)

Descripción

En esta fase, los evaluadores comprueban que la aplicación cierra automáticamente la sesión cuando un usuario ha permanecido inactivo durante un cierto periodo de tiempo, asegurando que no se pueda "reutilizar" la misma sesión y que ningún dato sensible permanezca almacenado en la caché del navegador.

Todas las aplicaciones deben implementar un tiempo de cierre de sesión por inactividad. Este tiempo de cierre define el período que una sesión se mantendrá activa en caso de que no haya actividad por parte del usuario. Cierre e invalide la sesión una vez excedido el período de inactividad definida desde la última petición HTTP recibida por la aplicación web para un determinado identificador de sesión. El tiempo de cierre más adecuado debe ser un equilibrio entre la seguridad (menor tiempo de espera) y usabilidad (mayor tiempo de espera), y mucho depende del nivel de sensibilidad de los datos manejados por la aplicación.

Por ejemplo, un tiempo de cierre de sesión de 60 minutos para un foro público puede ser aceptable, pero tanto tiempo sería demasiado en una aplicación de banca en casa (donde se recomienda un tiempo máximo de cierre de quince minutos). En todo caso, cualquier aplicación que no impone un cierre de sesión basado en el tiempo de espera debe considerarse insegura, a menos que tal comportamiento sea exigido por un requisito funcional específico. El tiempo de inactividad limita la ventana de oportunidad que un atacante tiene para adivinar y usar un identificador de otro usuario. Bajo ciertas circunstancias podría proteger a las computadoras públicas para que reutilicen una sesión válida. Sin embargo, si el atacante es capaz de secuestrar una sesión determinada, el tiempo de inactividad no limita las acciones del atacante, ya que puede generar periódicamente actividad dentro de la sesión para mantenerla activa por períodos de tiempo más largos.

La gestión de la caducidad y administración de tiempo de cierre de sesión debe ser realizada obligatoriamente desde el lado del servidor si algunos datos que se encuentran en control del cliente se utilizan para hacer cumplir el tiempo de cierre de sesión. Por ejemplo, usando valores de cookies u otros parámetros del cliente para hacer el seguimiento de las referencias de tiempo (p. ej. el número de minutos desde la hora de registro), un atacante podría manipular estos para extender la duración de la sesión.

Como resultado, la aplicación tiene que medir el tiempo de inactividad en el servidor y, una vez transcurrido el lapso de espera, automáticamente invalida la sesión del usuario actual y borra todos los datos almacenados en el cliente.

7.6.8 Pruebas de sesión puzzling (OTG-SESS-008)

Descripción

La sobrecarga de variables de sesión (también conocida como Session Puzzling) es una vulnerabilidad al nivel de la aplicación que permite a un atacante realizar una variedad de acciones maliciosas que incluyen, pero no se limitan a:

- ✓ Esquivar los mecanismos de autenticación de la aplicación y hacerse pasar por usuarios legítimos.
- ✓ Elevar los privilegios de una cuenta de usuario maliciosa, en un entorno que, de lo contrario, sería considerado infalible.
- ✓ Saltarse las fases de calificación en los procesos de fases múltiples, incluso si el proceso incluye todas las restricciones a nivel de código comúnmente recomendadas.
- ✓ Manipular los valores del lado del servidor de forma indirecta para que no puedan ser predichos o detectados.
- ✓ Ejecutar ataques tradicionales en lugares previamente inaccesibles, o incluso que se consideran seguros.

Esta vulnerabilidad se produce cuando una aplicación utiliza la misma variable de sesión para más de un propósito. Potencialmente un atacante puede acceder a páginas en un orden no previsible por parte de los desarrolladores para que la variable de sesión se configure en un contexto y luego se utilice en otro.

Por ejemplo, un atacante puede utilizar la sobrecarga de variables de sesión para eludir los mecanismos de autenticación de la aplicación, que garantizan la autenticación mediante la validación de la existencia de variables de sesión que contengan valores relacionados con la identidad, que normalmente se almacenan en la sesión después de un proceso de autenticación exitoso. Esto significa que, primero, un atacante tiene acceso a una ubicación en la aplicación que establece el contexto de la sesión y, luego, accede a lugares privilegiados que examinan este contexto.

Por ejemplo, un vector de ataque de evasión de autenticación podría ser ejecutado mediante el ingreso a un punto de entrada de acceso público (por ejemplo, una página de recuperación de contraseña) que rellena la sesión con una variable de sesión idéntica, basada en valores fijos o en información ingresada por un usuario.

Remediación

Las variables de sesión deben ser utilizadas con una sola finalidad.

7.7 Pruebas de validación de entradas

La debilidad de seguridad de las aplicaciones web más común es el no poder validar correctamente los datos de entrada que vienen del cliente o del medio ambiente antes de usarlo. Esta debilidad conduce a casi todas las vulnerabilidades principales en aplicaciones web, tales como cross site scripting, inyección SQL, inyección de intérprete, ataques ataques locale/Unicode, ataques al sistema de archivo y desbordamiento de búfer.

Nunca se debe confiar en los datos de una entidad externa o del cliente, ya que pueden ser arbitrariamente adulterados por un atacante. "Todas las entradas son malignas", dice Michael Howard en su famoso libro "Writing Secure Code" (Escribiendo Código Seguro). Esta es la regla número uno. Lamentablemente, las aplicaciones más complejas a menudo tienen un gran número de puntos de entrada, lo que hace difícil para que un desarrollador haga cumplir esta regla.

7.7.1 Pruebas de reflexión cross site scripting (OTG-INPVAL-001)

Descripción

La reflexión del Cross-site Scripting (XSS) ocurre cuando un atacante inyecta un código ejecutable en el navegador, dentro de una sola respuesta HTTP. El ataque inyectado no se almacena dentro de la aplicación, es no-persistente y sólo afecta a los usuarios que abren una página web de terceros o un vínculo diseñado con mala intención. La cadena de ataque se incluye como parte del diseño de los parámetros URL o HTTP, que se procesan incorrectamente por la aplicación y se devuelven a la víctima.

Las reflexiones de XSS son los tipos de ataque XSS más frecuentes encontrados en la naturaleza. Los ataques de reflexión XSS también son conocidos como ataques XSS no persistentes y, puesto que la carga de ataque es entregada y ejecutada mediante una sola solicitud y respuesta, también se les conoce como XSS de primer orden o tipo 1.

Cuando una aplicación web es vulnerable a este tipo de ataques, esta

pasará datos de entrada no validados mediante solicitudes al cliente. El modus operandi común del ataque incluye un paso de diseño, en el que el atacante crea y prueba una URI ofensiva; un paso de ingeniería social, en el cual ella convence a sus víctimas para que carguen esta URI en sus navegadores y la eventual ejecución del código ofensivo utilizando el navegador de la víctima.

Comúnmente, el código del intruso es escrito en el lenguaje Javascript, pero también se utilizan otros lenguajes, por ejemplo, ActionScript y VBScript. Los atacantes suelen aprovechar estas vulnerabilidades para instalar registradores de claves, robar las cookies de la víctima, realizar robos del portapapeles y cambiar el contenido de la página (por ejemplo, enlaces de descarga).

Una de las principales dificultades en la prevención de vulnerabilidades XSS es la correcta codificación de caracteres. En algunos casos, el servidor web o la aplicación web podrían no estar filtrando algunas codificaciones de caracteres.

Herramientas

- ✓ OWASP CAL9000: CAL9000 es una colección de herramientas de prueba de seguridad en aplicaciones web, que complementa el conjunto actual de proxys web y escáneres automatizados. Se presenta como una referencia en:
- ✓ PHP Charset Encoder(PCE): Esta herramienta le permite codificar textos arbitrarios desde y hacia 65 clases de conjuntos de caracteres. Además, se proporcionan algunas funciones de codificación destacadas de JavaScript.
- ✓ HackVertor: Ofrece varias docenas de codificaciones flexibles para ataques de manipulación de cadena avanzada.
- ✓ WebScarab: WebScarab es un framework para analizar las aplicaciones que se comunican mediante los protocolos HTTP y HTTPS.
- ✓ Burp Proxy: Burp Proxy es un servidor proxy HTTP/S interactivo que sirve para atacar o probar aplicaciones web.
- ✓ OWASP Zed Attack Proxy (ZAP): ZAP es una herramienta de

penetración integrada, fácil de usar para encontrar vulnerabilidades en aplicaciones web. Está diseñada para ser utilizada por personas con un amplio espectro de experiencia en seguridad y por eso es ideal para desarrolladores y evaluadores funcionales que son novatos realizando pruebas de penetración. ZAP ofrece escáneres automatizados, así como un conjunto de herramientas que permiten encontrar las vulnerabilidades de seguridad manualmente.

7.7.2 Pruebas para cross site scripting almacenados (OTG-INPVAL-002)

Descripción

El Cross-site Scripting almacenado (XSS) es el tipo más peligroso de Cross Site Scripting. Las aplicaciones web que permiten a los usuarios almacenar datos están potencialmente expuestas a este tipo de ataques. Este capítulo ilustra ejemplos de escenarios relacionados con la inyección y explotación de Cross-site Scripting almacenado.

El XSS almacenado se produce cuando una aplicación web reúne la información ingresada por un usuario que puede ser malicioso y esa información es almacenada para su uso posterior. La información almacenada no se filtra correctamente. Como consecuencia, los datos maliciosos aparecerán como parte del sitio web y se ejecutarán en el navegador del usuario con los privilegios de la aplicación web. Puesto que esta vulnerabilidad, por lo general, implica al menos dos peticiones a la aplicación, esto puede también llamarse XSS de segundo orden.

Esta vulnerabilidad se puede utilizar para llevar a cabo una serie de ataques basados en el navegador incluyendo:

- ✓ Secuestro del navegador de otro usuario.
- ✓ Captura de información confidencial vista por usuarios de la aplicación.
- ✓ Pseudo desfiguración de la aplicación.

- ✓ Escaneo de puertos en hosts internos ("internos" en relación a los usuarios de la aplicación web).
- ✓ Explotación basada en el navegador de entrega dirigida.
- ✓ Otras actividades maliciosas.

Un XSS almacenado no necesita un enlace malicioso para ser explotado. Una explotación exitosa se produce cuando un usuario visita una página con un XSS almacenado. Las fases siguientes se relacionan con una situación de ataque XSS almacenado normal:

- ✓ El atacante almacena el código malicioso en la página vulnerable.
- ✓ El usuario se autentica en la aplicación.
- ✓ El usuario visita páginas vulnerables.
- ✓ El código malicioso se ejecuta en el navegador del usuario.

Herramientas

- ✓ OWASP CAL9000: CAL9000 es una colección de herramientas de prueba de seguridad en aplicaciones web, que complementa el conjunto actual de proxys web y escáneres automatizados. Se presenta como una referencia en:
- ✓ PHP Charset Encoder(PCE): Esta herramienta le permite codificar textos arbitrarios desde y hacia 65 clases de conjuntos de caracteres. Además, se proporcionan algunas funciones de codificación destacadas de JavaScript.
- ✓ HackVertor: Ofrece varias docenas de codificaciones flexibles para ataques de manipulación de cadena avanzada.
- ✓ BeEF: BeEF es el explotador del framework del navegador (browser exploitation framework); una herramienta profesional para demostrar las vulnerabilidades del navegador en tiempo real.
- ✓ Burp Proxy: Burp Proxy es un servidor proxy HTTP/S interactivo que sirve para atacar o probar aplicaciones web.
- ✓ OWASP Zed Attack Proxy (ZAP): ZAP es una herramienta de penetración integrada, fácil de usar para encontrar

vulnerabilidades en aplicaciones web. Está diseñada para ser utilizada por personas con un amplio espectro de experiencia en seguridad y por eso es ideal para desarrolladores y evaluadores funcionales que son novatos realizando pruebas de penetración. ZAP ofrece escáneres automatizados, así como un conjunto de herramientas que permiten encontrar las vulnerabilidades de seguridad manualmente.

7.7.3 Pruebas de manipulación de verbos en HTTP (OTG-INPVAL-003)

Descripción

La especificación de HTTP incluye métodos de peticiones que no son estándar, como GET y POST. Un servidor de quejas de estándares web puede responder a estos métodos alternativos de una manera no prevista por los desarrolladores. Aunque la descripción común es manipulación, el estándar de HTTP 1.1 se refiere a estas solicitudes como métodos HTTP distintos.

La especificación completa de HTTP 1.1 define los siguientes métodos válidos de solicitudes HTTP:

- ✓ OPTIONS
- ✓ GET
- ✓ HEAD
- ✓ POST
- ✓ PUT
- ✓ DELETE

Si están habilitadas, las extensiones Web Distributed Authoring and Version (WebDAV) permiten muchos métodos HTTP más:

- ✓ PROPFIND
- ✓ PROPPATCH
- ✓ MKCOL
- ✓ COPY

- ✓ MOVE
- ✓ LOCK

Sin embargo, la mayoría de aplicaciones web sólo necesitan responder a solicitudes GET y POST, y proporcionar datos del usuario en la cadena de consulta de la dirección URL o anexa a la solicitud. El link normal de estilo desencadena una solicitud GET; el formulario de datos se envía mediante <form method="’POST’"></form> y desencadena las peticiones POST. Los formularios sin un método definido también envían los datos vía solicitudes GET, por defecto.

Curiosamente, los demás métodos válidos de HTTP no son compatibles con el estándar de HTML. Cualquier método HTTP distinto a GET o POST debe ser contactado fuera del documento HTML. Sin embargo, los contactos JavaScript y AJAX pueden enviar métodos distintos a GET y POST.

Si se requieren métodos como HEAD u OPTIONS para su aplicación, esto aumenta substancialmente la carga de la prueba. Cada acción dentro del sistema deberá verificarse para que estos métodos alternativos no activen acciones sin una apropiada autenticación ni revelen información sobre el contenido o funcionamiento de la aplicación web. Si es posible, limite el uso del método HTTP alternativo a una sola página que no contenga ninguna acción por parte del usuario, tal como la página principal (ejemplo: index.html).

7.7.4 Pruebas de contaminación de parámetros HTTP (OTG-INPVAL-004)

Descripción

Abastecer múltiples parámetros HTTP con el mismo nombre puede causar que una aplicación interprete los valores de maneras imprevistas. Al explotar estos efectos, un atacante puede ser capaz de esquivar la validación de entrada, provocar errores de aplicación o modificar valores de variables internas. Ya que la contaminación de parámetros HTTP (HTTP Parameter Pollution [abreviado HPP]) afecta los cimientos de la construcción de las tecnologías web, existen los ataques del lado del servidor y del cliente.

Las normas actuales de HTTP no incluyen una guía sobre cómo interpretar los parámetros de entrada múltiples con el mismo nombre. Por ejemplo, RFC 3986 simplemente define el concepto de cadena de consulta (Query String) como una serie de valores de campo pareados y RFC 2396 define clases de caracteres de la cadena de consulta inversa y sin reservas. Sin determinar un estándar, los componentes de la aplicación web manejan este caso extremo de diversas maneras (véase la siguiente tabla para más detalles).

Por sí misma, esta no es necesariamente una indicación de una vulnerabilidad. Sin embargo, si el desarrollador no está consciente del problema, la presencia de parámetros duplicados puede producir un comportamiento anómalo en la aplicación, que puede ser potencialmente aprovechado por un atacante. Comúnmente en la seguridad, los comportamientos inesperados suelen ser una fuente habitual de las debilidades que podrían conducir a ataques por contaminación de parámetros HTTP en este caso. Para presentar de mejor manera esta clase de vulnerabilidades y el resultado de los ataques de HPP, es interesante analizar algunos ejemplos de la vida real que han sido descubiertos en el pasado.

Herramientas

- ✓ OWASP ZAP HPP Passive/Active Scanners.
- ✓ HPP Finder (Chrome Plugin).

7.7.5 Pruebas de inyección de SQL (OTG-INPVAL-005)

Descripción

Un ataque de inyección SQL consiste en la inserción o "inyección" de una consulta SQL parcial o completa a través de los datos de entrada o transmitidos desde el cliente (navegador) hacia la aplicación web. Un ataque exitoso de inyección SQL puede leer datos sensibles de la base de datos, modificar datos de la base de datos (insertar/actualizar/eliminar), ejecutar operaciones administrativas de la base de datos (por ejemplo, apagar el

DBMS), recuperar el contenido de un archivo existente en el sistema de archivos DBMS o escribir archivos en el sistema de archivos y, en algunos casos, emitir comandos al sistema operativo. Los ataques de inyección SQL son un tipo de ataque de inyección en los que los comandos SQL se inyectan en la entrada de datos planos para afectar la ejecución de instrucciones SQL predefinidas.

Los ataques de inyección SQL pueden dividirse en las siguientes tres clases:

- ✓ Dentro de Banda (Inband): se extraen los datos utilizando el mismo canal que se utiliza para inyectar el código SQL. Este es el tipo más sencillo de ataque, en el que los datos recuperados se presentan directamente en la página web de la aplicación.
- ✓ Fuera de banda (Out-of-band): se recuperan los datos utilizando un canal diferente (por ejemplo, se genera un correo electrónico con los resultados de la consulta y se envía al evaluador).
- ✓ Inferencial o ciega (Inferential or Blind): no hay una transferencia real de datos, pero el evaluador es capaz de reconstruir la información enviando peticiones particulares y observando el comportamiento resultante del servidor de base de datos.

Un ataque exitoso de inyección SQL requiere que el atacante cree una consulta SQL sintácticamente correcta. Si la aplicación devuelve un mensaje de error generado por una consulta incorrecta, puede ser más fácil para un atacante reconstruir la lógica de la consulta original y, por lo tanto, entender cómo realizar la inyección correctamente. Sin embargo, si la aplicación oculta los detalles del error, entonces el evaluador debe ser capaz de invertir la lógica de la consulta original.

Sobre las técnicas de inyección SQL para explotar los defectos, hay cinco técnicas comunes. También las técnicas, a veces, se pueden utilizar de

forma combinada (e.g. operador de unión y fuera de banda):

- ✓ Operador de unión (Union Operator): se puede utilizar cuando ocurre la falla de inyección SQL en una instrucción SELECT, que permite combinar dos consultas en un único resultado o un conjunto de resultados.
- ✓ Booleano (Boolean): Utilice las condiciones booleanas para verificar si ciertas condiciones son verdaderas o falsas.
- ✓ Basadas en el error (Error based): esta técnica fuerza a la base de datos a generar un error, dando la información al atacante o evaluador con lo que puede refinar su inyección.
- ✓ Fuera de banda(out-of-band): técnica utilizada para recuperar datos utilizando un canal diferente (por ejemplo, hacer una conexión HTTP para enviar los resultados a un servidor web).
- ✓ Tiempo de retardo (Time Delay): Utilice comandos de base de datos (por ejemplo sleep) para retrasar las respuestas en consultas condicionales. Es útil cuando el atacante no tiene algún tipo de respuesta (resultado, salida o error) de la aplicación.

Herramientas

- ✓ SQL Injection Fuzz Strings.
- ✓ OWASP SQLiX.
- ✓ Multiple DBMS SQL Injection tool.
- ✓ Sqlmap, automatic SQL injection tool.
- ✓ Sqlninja, SQL Server Takeover Tool.
- ✓ Automated SQL Injection Tool.
- ✓ MySqloit, MySql Injection takeover tool.

Pruebas en Oracle

Descripción

Las aplicaciones web basadas en PL/SQL son habilitadas por el Gateway PL/SQL, que es el componente que traduce las solicitudes web en consultas de base de datos. Oracle ha desarrollado una serie de implementos

de software, que van desde el primer producto para escuchar en la web al módulo de Apache mod_plsql y al servidor web de base de datos XML (XDB). Todos tienen sus propias peculiaridades y problemas. Los productos que utilizan el Gateway PL/SQL incluyen, pero no se limitan tan solo al servidor HTTP de Oracle, eBusiness Suite, Portal, HTMLDB, WebDB y Oracle Application Server.

Cómo funciona el Gateway PL/SQL

Esencialmente el Gateway PL/SQL simplemente actúa como un servidor proxy que toma la solicitud del usuario web y se lo pasa al servidor de la base de datos donde se ejecuta.

- ✓ [1] El servidor web acepta una petición enviada por un cliente web y determina si debe ser procesada por el Gateway PL/SQL.
- ✓ [2] El Gateway PL/SQL procesa la solicitud extrayendo el nombre del paquete solicitado, procedimiento y variables.
- ✓ [3] El paquete y procedimiento solicitados son envueltos en un bloque de PL/SQL anónimo y enviados al servidor de la base de datos.
- ✓ [4] El servidor de la base de datos ejecuta el procedimiento y envía los resultados al Gateway como HTML.
- ✓ [5] El Gateway envía la respuesta, mediante el servidor web, al cliente.

Es importante entender que el código PL/SQL no existe en el servidor web, sino en el servidor de la base de datos. Esto significa que cualquier debilidad en el Gateway PL/SQL o cualquier debilidad en la aplicación de PL/SQL, cuando se explota, da al atacante un acceso directo al servidor de la base de datos; ninguna cantidad de firewalls evitará esto.

Herramientas

- ✓ SQLInjector.
- ✓ Orascan (Oracle Web Application VA scanner), NGS SQuirrel

(Oracle RDBMS VA Scanner).

Pruebas en Mysql

Descripción

Las vulnerabilidades de inyección SQL ocurren cuando los datos ingresados se utilizan en la construcción de una consulta SQL sin ser adecuadamente limitados o desinfectados. El uso de SQL dinámico (la construcción de consultas SQL de concatenación de cadenas) abre la puerta a estas vulnerabilidades. La inyección SQL permite a un atacante acceder a los servidores SQL. Permite la ejecución de código SQL bajo los privilegios del usuario utilizado para conectarse a la base de datos. El servidor MySQL tiene particularidades que hacen que algunas explotaciones necesiten modificarse especialmente para esta aplicación.

MySQL viene, por lo menos,con cuatro versiones que se utilizan en la producción a nivel mundial, 3.23.x, 4.0.x, 4.1.x y 5.0.x. Cada versión introduce un nuevo conjunto de características.

- ✓ From Version 4.0: UNION.
- ✓ From Version 4.1: Subqueries.
- ✓ From Version 5.0: Stored procedures, Stored functions and the view named INFORMATION_SCHEMA.
- ✓ From Version 5.0.2: Triggers.

Cabe señalar que en las versiones de MySQL anteriores a 4.0.x, solo los ataques de inyección ciega booleanos o basados en el tiempo podrían ser utilizados, puesto que la funcionalidad de subconsulta de instrucciones UNION no estaba implementada.

Herramientas

- ✓ SQLsus.
- ✓ Mysqloit.
- ✓ Sqlmap.

Pruebas del servidor SQL

Descripción

Las vulnerabilidades de inyección SQL ocurren cuando el ingreso de datos se utiliza en la construcción de una consulta SQL sin que sea adecuadamente limitada o desinfectada. El uso de SQL dinámico (la construcción de consultas SQL mediante la concatenación de cadenas) abre la puerta a estas vulnerabilidades. La Inyección SQL permite a un atacante acceder a los servidores SQL y ejecutar códigos SQL bajo los privilegios del usuario utilizado para conectarse a la base de datos.

Como se explica en la inyección de SQL, un ataque de inyección SQL requiere dos cosas: un punto de entrada y una explotación para entrar. Cualquier parámetro, controlado por el usuario, que se procesa en la aplicación podría estar ocultando una vulnerabilidad. Esto incluye:

- ✓ Los parámetros de aplicación en las cadenas de consulta (por ejemplo, solicitud GET).
- ✓ Los parámetros de aplicación incluidos como parte del cuerpo de una solicitud POST.
- ✓ Información relacionada con el navegador (por ejemplo, agente usuario, referido)
- ✓ Información relacionada con el host (por ejemplo, nombre de host, IP).
- ✓ Información relacionada con la sesión (por ejemplo, identificación del usuario, cookies).

Herramientas

- ✓ Sqlninja.
- ✓ Sqlmap.

Pruebas en PostgreSQL

Descripción

Las técnicas de inyección SQL para PostgreSQL tienen las siguientes características:

- ✓ El conector de PHP permite ejecutar múltiples instrucciones utilizándolo; como un separador de declaraciones.
- ✓ Las instrucciones SQL pueden truncarse anexando el comentario char: --.
- ✓ LÍMIT y OFFSET pueden utilizarse en una instrucción SELECT para recuperar una parte del conjunto de resultados generado por la consulta.

Pruebas para MS Access

Descripción

Las vulnerabilidades de inyección SQL ocurren cuando se usan los ingresos suministrados por el usuario durante la construcción de una consulta SQL sin estar adecuadamente limitada o desinfectada. Esta clase de vulnerabilidades permite a un atacante ejecutar códigos SQL bajo los privilegios del usuario que utiliza para conectarse a la base de datos. En esta sección, se discutirán las técnicas relevantes de inyección SQL que utilizan características específicas de Microsoft Access.

Desafortunadamente, MS Access no es compatible con operadores comunes que se utilizan tradicionalmente durante las pruebas de inyección, incluyendo:

- ✓ No comments characters.
- ✓ No stacked queries.
- ✓ No LIMIT operator.
- ✓ No SLEEP or BENCHMARK alike operators.
- ✓ And many others.

Sin embargo, es posible emular las funciones mediante la combinación de varios operadores o mediante el uso de técnicas alternativas. Como se mencionó, no es posible utilizar el truco de insertar los caracteres /*,

-- o # para truncar la consulta. Sin embargo, afortunadamente podemos evitar esta limitación mediante la inyección de un carácter 'null'. Al utilizar un byte nulo %00 en una consulta SQL resulta que MS Access ignora todos los caracteres restantes. Esto puede explicarse considerando que todas las cadenas son NULL en la representación interna utilizada por la base de datos. Cabe destacar que el carácter 'null' a veces puede causar problemas también, ya que puede truncar cadenas al nivel del servidor web. En esas situaciones, sin embargo, podemos emplear otro carácter: 0x16 (% 16 en formato URL codificado).

También hay muchas otras funciones que pueden utilizarse al realizar pruebas de inyección SQL, que incluyen, pero no limitan a:

- ✓ ASC: Obtiene el valor ASCII de un carácter que se pasa como ingreso.
- ✓ CHR: Obtiene el carácter del valor ASCII pasado como ingreso.
- ✓ LEN: Devuelve la longitud de la cadena pasada como parámetro.
- ✓ IIF: Es la estructura IF, por ejemplo, la siguiente instrucción IIF(1=1 'a', 'b') return 'a'.
- ✓ MID: Esta función le permite extraer una subcadena, por ejemplo, la siguiente instrucción mid('abc',1,1) return 'a'.
- ✓ TOP: Esta función le permite especificar el número máximo de resultados que la consulta debe devolver desde la parte superior. Por ejemplo, TOP 1 devolverá sólo una fila.
- ✓ LAST: Esta función se utiliza para seleccionar sólo la última fila de un conjunto de filas. Por ejemplo, la siguiente consulta SELECT last(*) FROM users devolverá sólo la última fila del resultado.

Pruebas de inyección Nosql

Descripción

Las bases de datos NoSQL ofrecen restricciones de consistencia más sueltas que las bases de datos SQL tradicionales. Por requerir menos restricciones relacionales y comprobaciones de coherencia, las bases de

datos NoSQL a menudo ofrecen beneficios por rendimiento y escala. Sin embargo, estas bases de datos aún son potencialmente vulnerables a ataques de inyección, incluso si no están usando la sintaxis SQL tradicional. Debido a que estos ataques de inyección de NoSQL pueden ejecutarse dentro de un lenguaje procesal, en lugar de un lenguaje SQL declarativo, los impactos potenciales son mayores que en una inyección SQL tradicional.

Las comunicaciones de base de datos NoSQL son escritas en el lenguaje de programación de la aplicación, una comunicación API personalizada o formateada según un acuerdo común (como XML, JSON, LINQ, etc.). Los registros maliciosos que apuntan a dichas especificaciones podrían no desencadenar los controles primarios de desinfección de la aplicación. Hoy encontramos más de 150 bases de datos NoSQL disponibles para usarlas dentro de una aplicación, y que proporcionan API en una variedad de lenguajes y modelos de relación. Cada una ofrece diferentes características y restricciones. Ya que no hay un lenguaje común entre ellas, una inyección de código de ejemplo no aplica a través de las bases de datos NoSQL. Por esta razón, cualquier persona que va a probar los ataques de inyección de NoSQL tendrá que familiarizarse con la sintaxis, modelo de datos y lenguaje de programación relacionada para poder elaborar pruebas específicas.

Los ataques de inyección de NoSQL pueden ejecutarse en diferentes áreas de una aplicación que la inyección de SQL tradicional. Mientras la inyección SQL se ejecutaría dentro del motor de la base de datos, las variantes NoSQL pueden ejecutarse dentro de la capa de aplicación o la capa de base de datos, dependiendo de la API de NoSQL y el modelo de datos usado. Por lo general, los ataques de inyección de NoSQL se ejecutarán donde la cadena de ataque es analizada, evaluada o concatenada con una comunicación a la API de NoSQL.

7.7.6 Pruebas de inyección de Ldap (OTG-INPVAL-006)

Descripción

El Protocolo Ligero de Acceso de Directorios (LDAP) se utiliza para almacenar información sobre usuarios, hosts y otros muchos objetos. La

inyección LDAP es un ataque de lado del servidor, que podría permitir que información sensible acerca de usuarios y hosts en una estructura LDAP pueda divulgarse, modificarse o insertarse. Esto se hace mediante la manipulación de parámetros de ingreso luego de pasarlos a las funciones de búsqueda interna, agregar y modificar.

Una aplicación web podría utilizar LDAP para permitir a los usuarios autenticarse o buscar información de otros usuarios dentro de una estructura corporativa. El objetivo de los ataques de inyección LDAP es inyectar meta caracteres de filtros de búsqueda LDAP en consultas que serán ejecutadas por la aplicación.

Las condiciones booleanas y agregaciones de grupo en un filtro de búsqueda LDAP pueden aplicarse utilizando los siguientes metacaracteres:

Metachar	Meaning
&	Boolean AND
|	Boolean OR
!	Boolean NOT
=	Equals
~=	Approx
>=	Greater than
<=	Less than
*	Any character
()	Grouping parenthesis

Una explotación exitosa de una vulnerabilidad de inyección LDAP podría permitir al evaluador:

- ✓ Acceso a contenido no autorizado.
- ✓ Evadir las restricciones de las aplicaciones.
- ✓ Reunir información no autorizada.
- ✓ Agregar o modificar objetos dentro de la estructura del árbol LDAP.

Herramientas

- ✓ Softerra ldap browser.

7.7.7 Pruebas de inyección de Orm (OTG-INPVAL-007)

Descripción

La inyección de ORM es un ataque de inyección SQL contra un modelo de objetos de acceso de datos generado mediante ORM. Desde el punto de vista del evaluador, este ataque es virtualmente idéntico a un ataque de inyección SQL. Sin embargo, la vulnerabilidad de inyección existe en el código generado por la herramienta ORM. Un ORM es una herramienta de mapeo relacional de objetos. Se utiliza para acelerar el desarrollo orientado a objetos dentro de la capa de acceso de datos de las aplicaciones de software, incluyendo aplicaciones web.

Los beneficios de utilizar una herramienta ORM incluyen la rápida generación de una capa de objeto para comunicarse con una base de datos relacional, plantillas de código estandarizado para estos objetos y, generalmente, un conjunto de funciones seguras para protegerse contra ataques de inyección SQL. Los objetos ORM generados pueden utilizar SQL o, en algunos casos, una variante de SQL, para realizar las operaciones CRUD (Create, Read, Update, Delete); en español: crear, leer, actualizar, eliminar, en una base de datos. Es posible, sin embargo, para una aplicación web que utiliza objetos generados mediante ORM, ser vulnerable a ataques de inyección SQL si los métodos permiten parámetros de entrada sin desinfectar.

Las herramientas ORM incluyen Hibernate para Java, NHibernate para. .NET, ActiveRecord para Ruby on Rails, EZPDO para PHP y muchos otros. La mayoría de herramientas ORM proporcionan funciones seguras para escapar el ingreso del usuario. Sin embargo, si estas funciones no se usan y el programador utiliza funciones personalizadas que aceptan ingresos del usuario, es posible ejecutar un ataque de inyección SQL.

Herramientas

- ✓ Hibernate.
- ✓ Nhibernate.

7.7.8 Pruebas de inyección de Xml (OTG-INPVAL-008)

Descripción

La prueba de inyección XML se da cuando un evaluador intenta inyectar un documento XML a la aplicación. Si el analizador XML falla en la validación de datos dentro de su contexto, la prueba dará un resultado positivo. En primer lugar, una comunicación de estilo XML se define y se explican los principios de funcionamiento. Luego, el método de descubrimiento por el cual tratamos de insertar metacaracteres XML. Una vez que se logra el primer paso, el evaluador tendrá alguna información sobre la estructura XML, por lo que será posible intentar inyectar datos y etiquetas XML (inyección de etiquetas).

El primer paso para probar una aplicación en busca de una vulnerabilidad de inyección XML consiste en intentar insertar metacaracteres XML. Los metacaracteres XML son:

- ✓ Comilla simple (Single quote): '-cuando no ha sido desinfectado, este carácter podría lanzar una excepción durante el análisis del XML, si el valor inyectado va a ser parte de un valor de atributo en una etiqueta.
- ✓ Comilla Doble (Double quote): “ - este carácter tiene el mismo significado que la comilla simple y puede utilizarse si el valor del atributo está encerrado entre comillas dobles.
- ✓ Paréntesis angular(Angular parentheses): > y < - Aumentando un paréntesis angular abierto o cerrado en el ingreso de datos de un usuario.
- ✓ Etiqueta de comentario (Comment tag): <!--/--> - Esta secuencia de caracteres se interpreta como el inicio/final de un comentario.
- ✓ (Ampersand): & - El signo ampersand es usado en la sintaxis

XML para representar entidades. El formato de una entidad es '&symbol;'. Una entidad es asignada con un carácter en el conjunto de caracteres Unicode.

- ✓ Delimitadores de sección CDATA: <![CDATA[/]]> - Las secciones CDATA se utilizan para escaparse de bloques de texto que contengan caracteres que, de lo contrario, serían reconocidos como marcas. En otras palabras, los caracteres dentro de una sección CDATA no son analizadas por un evaluador de XML.

Entidad externa

El conjunto de entidades válidas puede ampliarse mediante la creación de nuevas entidades. Si la definición de una entidad es un URI, a la entidad se le llama una entidad externa. A menos que se configure para hacer lo contrario, las entidades externas fuerzan al evaluador XML para que acceda al recurso especificado por el URI, por ejemplo, un archivo en el equipo local o en un sistema remoto. Este comportamiento expone a la aplicación a ataques de XML external entity (XXE), que puede utilizarse para denegar el servicio del sistema local, obtener acceso no autorizado a los archivos en el equipo local, analizar equipos remotos y denegar el servicio de sistemas remotos.

Herramientas

- ✓ XML Injection Fuzz Strings.

7.7.9 Pruebas de inyección de Ssi (OTG-INPVAL-009)

Descripción

Los servidores web suelen dar a los desarrolladores la posibilidad de añadir pequeñas piezas de código dinámico dentro de páginas HTML estáticas, sin tener que lidiar con todos los derechos de los lenguajes del servidor o del cliente. Esta característica es encarnada cuando el servidor incluye (SSI). En la prueba de inyección SSI, probamos si es posible inyectar en los datos de la aplicación que serán interpretados por mecanismos del SSI.

Una explotación exitosa de esta vulnerabilidad permite a un atacante inyectar código en páginas HTML o incluso realizar ejecución remota de códigos.

Server-Side Includes son directivas que el servidor web analiza antes de servir la página al usuario. Representan una alternativa para escribir programas CGI o incrustar código utilizando lenguajes de scripting del lado del servidor, cuando sólo hay que realizar tareas muy simples. Las implementaciones comunes de SSI proporcionan comandos para incluir archivos externos, configurar e imprimir las variables del entorno CGI del servidor web y ejecutar scripts CGI externos o comandos del sistema.

Entonces, si el soporte de SSI del servidor web está habilitado, el servidor analizará estas directivas. En la configuración predeterminada, por lo general, la mayoría de servidores web no permiten el uso de la directiva exec para ejecutar comandos del sistema.

Como en toda situación de una mala validación de entrada, los problemas surgen cuando el usuario de una aplicación web puede proporcionar datos que hacen que la aplicación o el servidor web se comporten de manera imprevista. Con respecto a la inyección SSI, el atacante podría aportar datos que, si son ingresados por la aplicación (o tal vez directamente por el servidor) en una página generada dinámicamente, se puede analizar como una o más directivas SSI.

Si tenemos acceso al código fuente de la aplicación, fácilmente podemos averiguar:

- ✓ [1] Si se usan directivas SSI, el servidor web va a tener el soporte SSI activo. Hacer una inyección de SSI es, por lo menos, un problema potencial que debe investigarse.
- ✓ [2] Donde se manejan los datos ingresados por el usuario, el contenido de las cookies y los encabezados HTTP. La lista completa de vectores de entrada puede determinarse rápidamente.
- ✓ [3] Cómo se manejan los datos ingresados, qué tipo de filtro se realiza, qué caracteres no permiten pasar la aplicación y cuántos tipos de codificación se consideran.

Herramientas

- ✓ Web Proxy Burp Suite.
- ✓ Paros.
- ✓ OWASP WebScarab.
- ✓ String searcher.

7.7.10 Pruebas de inyección de Xpath (OTG-INPVAL-010)

Descripción

XPath es un lenguaje que ha sido diseñado y desarrollado, sobre todo, para dirigirse a las piezas de un documento XML. En la prueba de inyección XPath, probamos si es posible inyectar la sintaxis de XPath en una solicitud interpretada por la aplicación, permitiendo a un atacante ejecutar consultas XPath controladas por el usuario. Cuando se explota con éxito, esta vulnerabilidad podría permitir a un atacante eludir mecanismos de autenticación o información sin la debida autorización.

Las aplicaciones web utilizan constantemente bases de datos para almacenar y acceder a los datos que necesitan para sus operaciones. Históricamente, las bases de datos relacionales han sido, en gran medida, la tecnología más común para el almacenamiento de datos, pero, en los últimos años, hemos sido testigos de una creciente popularidad de las bases de datos que organizan los datos mediante el lenguaje XML. Tal como las bases de datos relacionales se acceden a través del lenguaje SQL, las bases de datos XML utilizan XPath como su lenguaje de consulta estándar.

Ya que, desde un punto de vista conceptual, XPath es muy similar a SQL en sus propósitos y usos, un resultado interesante es que los ataques de inyección XPath siguen la misma lógica que los ataques de inyección SQL. En algunos aspectos, XPath es incluso más poderoso que el SQL estándar, ya que todo su poder está ya presente en sus especificaciones, mientras que un gran número de técnicas que pueden utilizarse en un ataque de inyección SQL depende de las características del dialecto SQL usado por la base de datos de destino. Esto significa que los ataques de inyección XPath pueden ser mucho más adaptables y ubicuos. Otra de las ventajas de un ataque de

inyección XPath es que, a diferencia del SQL, no se aplica ningún ACL ya que nuestra consulta puede acceder a todas las partes del documento XML.

Como en un ataque de inyección SQL común, hemos creado una consulta que siempre se evalúa como verdadera, lo que significa que la aplicación autenticará al usuario incluso si no ha proporcionado un nombre de usuario o una contraseña. Y como en un ataque de inyección SQL comun, con la inyección XPath, el primer paso es insertar una comilla sencilla (') en el campo que vamos a probar, introducir un error de sintaxis en la consulta y así comprobar si la aplicación devuelve un mensaje de error.

Si no hay ningún conocimiento acerca de los detalles internos de datos XML, y si la aplicación no proporciona mensajes de error útiles que nos ayuden a la reconstrucción de su lógica interna, es posible realizar un ataque de inyección ciega de XPath, cuyo objetivo es reconstruir toda la estructura de datos. La técnica es similar a la inyección de SQL basada en la inferencia, ya que el método consiste en inyectar el código que crea una consulta que devuelve un bit de información. La inyección ciega de XPath se explica más detalladamente por Amit Klein en el documento “Blind XPath Injection”.

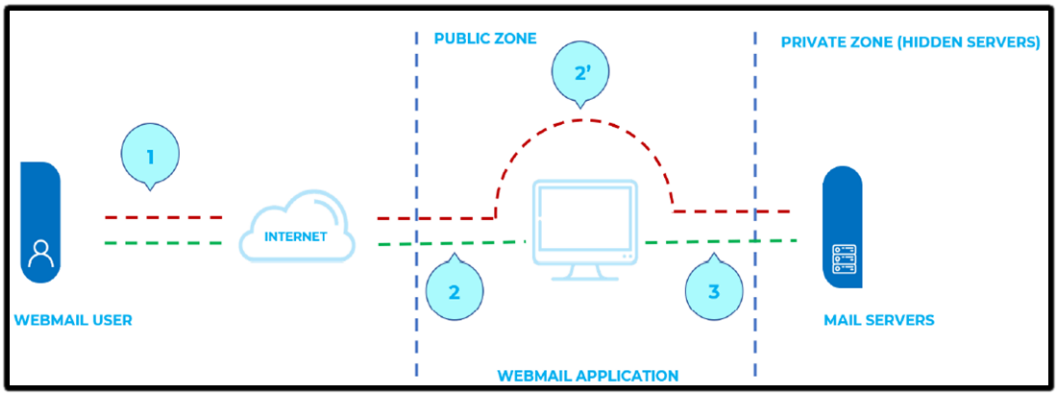
7.7.11 Pruebas de inyección de Imap/Smtp (OTG-INPVAL-011)

Descripción

Esta amenaza afecta a todas las aplicaciones que se comunican con servidores de correo (IMAP/SMTP), generalmente aplicaciones de webmail. El objetivo de esta prueba es verificar la capacidad de inyectar comandos IMAP/SMTP arbitrarios en los servidores de correo, debido a que el ingreso de datos no está correctamente desinfectado.

La técnica de inyección IMAP/SMTP es más eficaz si el servidor de correo no es directamente accesible desde el Internet. Donde una comunicación completa con el servidor de correo de acceso restringido sea posible, se recomienda realizar pruebas directas. Una inyección IMAP/SMTP permite acceder a un servidor de correo que, de otra manera, no sería directamente accesible desde Internet. En algunos casos, estos sistemas internos no tienen el mismo nivel de seguridad y rigurosidad en la

infraestructura que se aplica a los servidores web de acceso frontal. Por lo tanto, los resultados de los servidores de correo pueden ser más vulnerables a ataques por parte de usuarios finales (vea el esquema presentado a continuación)



La figura 1 muestra el flujo de tráfico visto generalmente al utilizar tecnologías de webmail. Los pasos 1 y 2 son el usuario interactuando con el cliente de correo web, mientras que el paso 3 es el evaluador evadiendo al cliente webmail e interactuando directamente con los servidores de correo de acceso restringido (back-end). Esta técnica permite una amplia variedad de acciones y ataques. Las posibilidades dependen del tipo y ámbito de la inyección y la tecnología del servidor de correo que se está probando.

Referencias

- ✓ RFC 0821.
- ✓ RFC 3501.
- ✓ Vicente Aguilera Díaz: “MX Injection: Capturing and Exploiting Hidden Mail Servers”.

7.7.12 Pruebas de inyección de código (OTG-INPVAL-012)

Descripción

En la prueba de inyección de código, un evaluador envía información que es procesada por el servidor web como código dinámico o como un archivo incluido. Estas pruebas pueden dirigirse a varios motores de secuencias de scripting del servidor, como ASP o PHP. Una correcta validación de la entrada y prácticas de codificación seguras deben emplearse

para protegerse de estos ataques.

Pruebas para determinar la inclusión de documentos locales

Descripción

La vulnerabilidad de inclusión de documentos permite al atacante incluir un documento, normalmente explotando un mecanismo de "inclusión de documento dinámica" implementado en la aplicación objetivo. La vulnerabilidad ocurre debido al uso de datos ingresados por el usuario sin una apropiada validación. Esto puede conducir a algo como mostrar el contenido del archivo, pero dependiendo de la gravedad, también puede llevar a:

- ✓ Ejecución de código en el servidor web.
- ✓ Ejecución de código en el lado del cliente como JavaScript que puede conducir a otros ataques tales como cross site scripting (XSS).
- ✓ Negación de servicio (DoS).
- ✓ Despliegue de información sensible.

La inclusión de archivos locales (también conocida como LFI) es el proceso de incluir archivos, que ya están presentes localmente en el servidor, a través de la explotación de las vulnerabilidades en los procedimientos de inserción implementados en la aplicación. Esta vulnerabilidad se produce, por ejemplo, cuando una página recibe, como datos, la ruta del archivo que debe estar incluida y este dato no ha sido correctamente desinfectado, permitiendo que caracteres de salto de directorio (como dot-dot-slash) sean inyectados. Aunque la mayoría de ejemplos apuntan a scripts PHP vulnerables, debemos tener en cuenta que también es común en tecnologías como JSP, ASP y otras.

Remediación

La solución más eficaz para eliminar las vulnerabilidades de inclusión de archivo es evitar pasar datos ingresados por el usuario a cualquier filesystem/framework API. Si esto no es posible, la aplicación puede mantener una lista blanca de archivos que pueden ser incluidos por la página y luego utilizar un identificador (por ejemplo, el número de índice) para tener acceso

al archivo seleccionado. Cualquier solicitud que contenga un identificador inválido será rechazada; de esta manera, no hay ninguna superficie de ataque para que usuarios malintencionados puedan manipular la ruta.

Pruebas para la inclusión remota de archivos

Descripción

La vulnerabilidad de inclusión de archivos permite que un atacante incluya un archivo, generalmente, aprovechando un mecanismo de "inclusión dinámica de archivos" implementado en la aplicación de destino. La vulnerabilidad se produce debido al uso de datos suministrados por el usuario sin una validación adecuada. Esto puede llevar a que se muestre el contenido del archivo, pero dependiendo de la gravedad, también puede llevar a:

- ✓ Ejecución de código en el servidor web.
- ✓ Ejecución de código en el lado del cliente como JavaScript que nos llevaría a otros ataques tales como cross site scripting (XSS).
- ✓ Negación de servicio (DoS).
- ✓ Publicación de información sensitiva.

La inclusión remota de archivos (también conocida como RFI) es el proceso de incluir archivos remotos a través de la explotación de las vulnerabilidades en los procedimientos de inserción implementados en la aplicación. Esta vulnerabilidad se produce, por ejemplo, cuando una página recibecomo datos, la ruta del archivo que debe estar incluida y este dato no ha sido correctamente desinfectado, permitiendo que una URL externa se inyecte. Aunque la mayoría de ejemplos apuntan a scripts PHP vulnerables, debemos tener en cuenta que también es común en tecnologías como JSP, ASP y otras.

Remediación

La solución más eficaz para eliminar las vulnerabilidades de inclusión de archivo es evitar pasar datos enviados por el usuario a cualquier filesystem/framework API. Si esto no es posible, la aplicación puede mantener una lista blanca de archivos que pueden ser incluidos por la página y luego

utilizar un identificador (por ejemplo, el número de índice) para tener acceso al archivo seleccionado. Cualquier solicitud que contenga un identificador inválido será rechazada, de esta manera no hay ninguna superficie de ataque para que usuarios malintencionados puedan manipular la ruta.

7.7.13 Pruebas de inyección de comandos (OTG-INPVAL-013)

Descripción

La Inyección de comandos del sistema operativo es una técnica utilizada a través de una interfaz web para ejecutar comandos del sistema operativo en un servidor web. El usuario provee comandos del sistema operativo a través de una interfaz web para ejecutar comandos del sistema operativo. Cualquier interfaz que no esté correctamente desinfectada está sujeta a esta debilidad. Con la habilidad de ejecutar comandos en el sistema operativo, el usuario puede subir programas maliciosos o incluso obtener contraseñas. La inyección de comandos del sistema operativo es prevenible cuando la seguridad se acentúa durante el diseño y desarrollo de las aplicaciones.

Herramientas

- ✓ OWASP ZAP.

Remediación

Desinfección

Los formularios de datos y la URL deben desinfectarse de caracteres no válidos. Una "lista negra" de caracteres es una opción, pero puede ser difícil pensar en todos los caracteres contra los que hay que validar. También pueden haber algunos que no han sido descubiertos todavía. Para validar la entrada del usuario se debe crear una "lista blanca" que contenga sólo caracteres permitidos. Caracteres que faltaron, así como las amenazas desconocidas, deben ser eliminados de esta lista.

Permisos

La aplicación web y sus componentes deben ejecutarse bajo permisos estrictos que no permitan la ejecución de comandos del sistema operativo.

7.7.14 Pruebas la saturación de búfer (OTG-INPVAL-014)

Descripción

Para conocer más acerca de las vulnerabilidades de saturación de búfer, consulte las páginas de saturación de búfer.

- ✓ Vea el artículo OWASP sobre ataques de saturación de búfer.
- ✓ Vea el artículo OWASP sobre vulnerabilidades de saturación de búfer.

Diferentes tipos de vulnerabilidades de saturación de búfer tienen diferentes métodos de prueba. Aquí están los métodos de prueba para los tipos comunes de vulnerabilidades de saturación de búfer.

- ✓ Pruebas de vulnerabilidad de saturación del heap.
- ✓ Pruebas de vulnerabilidad de saturación de pilas de datos.
- ✓ Pruebas de vulnerabilidad de cadena de formato.

Pruebas de saturación de Heap

Descripción

Heap es un segmento de memoria que se utiliza para almacenar datos dinámicamente asignados y variables globales. Cada fragmento de la memoria en el heap consiste en etiquetas de límite que contienen información de gestión de memoria. Cuando un búfer basado en heap se satura, se sobrescribe la información de control en estas etiquetas. Cuando la rutina de gestión del heap libera el búfer, una sobrescritura de la dirección de la memoria da lugar a una infracción de acceso. Cuando la saturación se ejecuta de manera controlada, la vulnerabilidad permitirá a un adversario sobrescribir en una ubicación de memoria deseada con un valor controlado por el usuario. En la práctica, un atacante podrá sobrescribir funciones de dirección y varias direcciones almacenadas en estructuras como GOT, .dtors o TEB con la dirección de una carga maliciosa útil.

Hay numerosas variantes de la vulnerabilidad de saturación de heap (corrupción del heap) que puede permitir cualquier cosa, desde sobrescribir punteros de función a la explotación de las estructuras de gestión de memoria

para la ejecución de código arbitrario. Localizar la saturación de heap requiere una revisión más detallada en comparación con la saturación de pilas de datos, ya que hay ciertas condiciones que deben existir en el código de estas vulnerabilidades para que sean explotables.

Herramientas

- ✓ OllyDbg: “Un depurador basado en windows utilizado para el análisis de vulnerabilidades de saturación de búfer”.
- ✓ Spike, un fuzzer framework que puede utilizarse para explorar las vulnerabilidades y realizar pruebas largas.
- ✓ Brute Force Binary Tester (BFB), un controlador binario proactivo.
- ✓ Metasploit, un framework de desarrollo, explotación y evaluación rápido.

Pruebas de saturación de pila de datos

Descripción

La saturación de pila de datos se produce cuando se copian datos de tamaño variable en búferes de longitud ubicados en la pila de datos del programa sin ninguna comprobación de los límites. Las vulnerabilidades de esta clase se consideran generalmente de alta severidad, ya que su explotación permitiría, sobre todo, la ejecución de código arbitrario o negación de servicio. Aunque rara vez se encuentra en plataformas interpretadas, el código escrito en C y lenguajes similares es, a menudo, montado con instancias de esta vulnerabilidad. De hecho, casi todas las plataformas son vulnerables a saturación de pila de datos con las siguientes excepciones notables:

- ✓ J2EE – siempre y cuando no se invoquen métodos nativos o comunicación con el sistema.
- ✓ .NET – siempre que el código inseguro o no administrado no sea invocado (tal como el usado en P/Invoke or COM Interop).
- ✓ PHP – mientras que no se comuniquen con los programas externos y las extensiones PHP vulnerables escritas en C o C++ las cuales pueden sufrir de saturación de pila de datos.

Las vulnerabilidades de saturación de pila de datos a menudo permiten a un atacante tomar directamente el control del puntero de instrucción y, por lo tanto, alterar la ejecución del programa y ejecutar el código arbitrario. Además de sobrescribir el puntero de instrucción, resultados similares también se pueden obtener sobreescribiendo otras variables y estructuras, como los controladores de excepciones, que se encuentran en la pila de datos.

Herramientas

- ✓ OllyDbg: “Un depurador basado en windows, utilizado para el análisis de vulnerabilidades de saturación de búfer”.
- ✓ Spike, un fuzzer framework que puede utilizarse para explorar las vulnerabilidades y realizar pruebas largas.
- ✓ Brute Force Binary Tester (BFB), un controlador binario proactivo.
- ✓ Metasploit, un framework de desarrollo, explotación y evaluación rápido.

Pruebas para la cadena de formato

Descripción

El problema surge por la utilización de datos ingresados por el usuario, que no han sido filtrados, como el parámetro de formato de cadena en ciertas funciones C que realizan el formateo, como printf(). Varios lenguajes de estilo C disponen de un formato de salida por medio de funciones como printf (), fprintf () etc.

El formateo se rige a un parámetro de estas funciones como un especificador del tipo de formato, normalmente %s, %c etc. La vulnerabilidad se presenta cuando se contactan funciones de formato que tienen una inadecuada validación de parámetros y datos controlados por el usuario.

Enumerar el proceso de la pila de datos

Esto permite a un adversario ver la organización del proceso vulnerable de apilamiento, mediante el suministro de cadenas de formato como, %x o %p, que puede conducir a la fuga de información sensible. También puede ser utilizado para extraer valores de "canario" cuando la

aplicación está protegida con un mecanismo de seguridad para la pila de datos. Junto con una saturación de pila de datos, esta información puede utilizarse para saltarse el protector de apilamiento.

Control del flujo de ejecución

Esta vulnerabilidad también puede facilitar la ejecución de código arbitrario, ya que permite escribir cuatro bytes de datos en una dirección suministrada por el adversario. El especificador %n es útil para sobrescribir varios punteros de función en la memoria con la dirección de la carga maliciosa. Cuando se contactan estos punteros de función sobrescritos, al ejecutarse pasan el código malicioso.

Negación de servicio

Si el adversario no está en condiciones de suministrar código malicioso para su ejecución, la aplicación vulnerable puede ser bloqueada suministrando una secuencia de %x seguido de %n.

Herramientas

- ✓ ITS4: “Una herramienta de análisis estático del código para la identificación de vulnerabilidades de cadenas de formato con código fuente”.
- ✓ Un constructor de cadenas de explotación para errores de formato.

7.7.15 Pruebas de las vulnerabilidades incubadas (OTG-INPVAL-015)

Descripción

También conocidos como ataques persistentes, la prueba de incubación es un método de prueba complejo que necesita más de una vulnerabilidad de validación de datos para que funcione. Las vulnerabilidades incubadas se utilizan típicamente para llevar a cabo ataques de "abrevadero" contra usuarios legítimos de aplicaciones web. Las vulnerabilidades incubadas tienen las siguientes características:

- ✓ En primer lugar, el vector de ataque debe ser constante y debe

almacenarse en la capa de persistencia. Esto ocurrirá únicamente si una validación de datos débil está presente o los datos llegaron al sistema a través de otro canal como una consola de administración o directamente a través de un proceso de datos restringidos.

- ✓ Segundo, una vez que el vector de ataque ha sido "contactado", este necesita ejecutarse de una manera satisfactoria. Por ejemplo, un ataque XSS incubado requiere una validación de salida de datos débil para que el script pueda ser entregado al cliente de una manera ejecutable.

La explotación de algunas vulnerabilidades, o incluso características funcionales de una aplicación web, permitirá a un atacante plantar una sección de datos que más tarde se recuperará mediante un usuario desprevenido u otro componente del sistema, explotando así alguna vulnerabilidad.

En una prueba de penetración, los ataques incubados pueden utilizarse para evaluar la importancia de ciertos errores, utilizando el tema de la seguridad particular que encontró para construir un ataque basado en el lado del cliente; este se utiliza generalmente para atacar a un gran número de víctimas al mismo tiempo (es decir, a todos los usuarios que navegan por el sitio).

Servidor mal configurado

Algunos servidores web presentan una interfaz de administración que puede permitir a un atacante cargar componentes activos de su elección en el sitio. Esto podría ser el caso con un servidor Apache Tomcat que no obliga a utilizar credenciales fuertes para acceder a su gestor de aplicaciones web (o si los evaluadores de edición han sido capaces de obtener credenciales válidas para el módulo de administración por otros medios). En este caso, puede cargarse un archivo WAR y una nueva aplicación web desplegarse en el sitio, que no sólo permita al evaluador de edición ejecutar localmente el código a su elección en el servidor, sino también para plantar una aplicación en el sitio de confianza, al que los usuarios regulares del sitio puedan acceder (probablemente con un mayor grado de confianza que al acceder a un sitio diferente).

Como también debería ser obvio, la habilidad de cambiar el contenido de la página web en el servidor a través de cualquier vulnerabilidad que puede ser explotable en el host dará al atacante permisos de escritura en el webroot, y también será útil cuando se quiera plantar un ataque incubado semejante en las páginas del servidor web (en realidad, se trata de un método de propagación de la infección conocido para algunos gusanos de servidores web).

Herramientas

- ✓ XSS-proxy.
- ✓ Paros.
- ✓ Burp Suite.
- ✓ Metasploit.

7.7.16 Pruebas para verificar la separación/contrabando de HTTP (OTG-INPVAL-016)

Descripción

Estos ataques aprovechan características específicas del protocolo HTTP, ya sea mediante la explotación de las debilidades de la aplicación web o peculiaridades, en la manera en que los diferentes agentes interpretan los mensajes HTTP. Esta sección analizará dos diferentes ataques dirigidos a encabezados HTTP específicos:

- ✓ Separación HTTP (HTTP splitting).
- ✓ Contrabando HTTP (HTTP smuggling).

El primer ataque explota la falta de desinfección de datos ingresados que permite a un intruso insertar caracteres CR y LF en los encabezados de la respuesta de la aplicación y 'separar' la respuesta en dos mensajes HTTP diferentes. El objetivo del ataque puede variar desde un envenenamiento del caché hasta un cross site scripting.

En el segundo ataque, el atacante explota el hecho de que algunos mensajes HTTP especialmente diseñados pueden ser analizados e interpretados de diferentes maneras según el agente que las reciba. El contrabando de HTTP requiere cierto nivel de conocimiento sobre los

diferentes agentes que están manejando los mensajes HTTP (servidor web, proxy, firewall).

Separación HTTP

Ayuda enormemente a una explotación exitosa de separación HTTP si se sabe algunos detalles de la aplicación web y del destino del ataque. Por ejemplo, diferentes objetivos pueden utilizar diversos métodos para decidir cuándo termina el primer mensaje HTTP y cuándo inicia el segundo. Algunos utilizan los límites del mensaje, como en el ejemplo anterior. Otros objetivos asumen que diferentes mensajes se transportarán en diferentes paquetes. Otros destinarán para cada mensaje un número de piezas de longitud predeterminada: en este caso, el segundo mensaje debe iniciar exactamente al principio del pedazo y, para ello, será necesario que el evaluador utilice relleno entre los dos mensajes.

Contrabando HTTP

Como se mencionó en la introducción, el contrabando HTTP aprovecha las diferentes maneras en que un mensaje HTTP especialmente diseñado puede ser analizado e interpretado por los diferentes agentes (navegadores, cachés web, aplicaciones de firewalls). Este relativamente nuevo tipo de ataque fue descubierto por Chaim Linhart, Amit Klein, Ronen Heled y Steve Orrin en 2005.

7.8 Manejo de errores

7.8.1 Pruebas de errores de código (OTG-ERR-001)

Descripción

A menudo, durante una prueba de penetración en aplicaciones web, nos encontramos con muchos errores de código generados por aplicaciones o servidores web. Es posible hacer que estos errores se muestren al utilizar una solicitud personalizada, creada especialmente con herramientas o manualmente. Estos códigos son muy útiles para los evaluadores de penetración durante sus actividades, ya que revelan mucha información sobre bases de datos, errores y otros componentes tecnológicos directamente relacionados con aplicaciones web.

El aspecto más importante para esta actividad es centrar la atención en estos errores, verlos como una colección de información que ayudará en los pasos de nuestro análisis. Una buena colección puede facilitar la eficiencia de la evaluación reduciendo el tiempo total tomado para realizar la prueba de penetración.

Los atacantes a veces usan motores de búsqueda para localizar errores que divulguen la información. Las búsquedas se realizan para encontrar los sitios erróneos como víctimas al azar, o es posible buscar errores en un sitio específico utilizando el motor de búsqueda, herramientas de filtro como las que se describen en (OTG-INFO -001).

Errores de servidor web

Un error común que podemos ver durante la prueba es el HTTP 404 Not Found. A menudo este código de error proporciona información útil sobre el servidor web subyacente y componentes asociados.

Este mensaje de error puede generarse por solicitar una URL no existente. Después del mensaje común que muestra una página no encontrada, hay información sobre la versión del servidor web, sistema operativo, módulos y otros productos utilizados. Esta información puede ser muy importante desde el punto de vista de identificar el tipo y versión del sistema operativo y aplicaciones. Otros códigos de respuesta HTTP tales como 400 Bad Request, 405 Method Not Allowed, 501 Method Not Implemented, 408 Request Time-out and 505 HTTP Version Not Supported pueden ser forzados por un atacante. Cuando reciben solicitudes especialmente diseñadas, los servidores web pueden proporcionar uno de estos códigos de error, dependiendo de su aplicación HTTP.

Errores de servidores de aplicaciones

Los errores de aplicación son devueltos por la misma aplicación más

que por el servidor web. Estos pueden ser mensajes de error de código del framework (ASP, JSP etc.) o pueden ser errores específicos devueltos por el código de la aplicación. Los errores detallados de la aplicación normalmente proporcionan información de las rutas del servidor, bibliotecas instaladas y versiones de aplicaciones.

Errores de base de datos

Los errores de base de datos son devueltos por el sistema de base de datos cuando hay un problema con la consulta o la conexión. Cada sistema de base de datos, como MySQL, Oracle o MSSQL, tiene su propio conjunto de errores. Esos errores pueden proporcionar información confidencial como las IP del servidor de base de datos, tablas, columnas y datos de inicio de sesión.

Manejo de errores en IIS y ASP .net

ASP.net es un framework común de Microsoft utilizado para desarrollar aplicaciones web. IIS es uno de los servidores web más utilizados. Los errores se producen en todas las aplicaciones, los desarrolladores intentan atrapar la mayoría de errores, pero es casi imposible cubrir cada excepción (sin embargo, es posible configurar el servidor web para suprimir que sean devueltos los mensajes de error detallado al usuario).

IIS utiliza un conjunto de páginas de error personalizadas que generalmente se encuentra en c:\winnt\help\iishelp\common para mostrar los errores como "404 page not found " al usuario.

Manejo de errores en Apache

Apache es un servidor HTTP común para atender páginas HTML y PHP. Por defecto, Apache muestra la versión del servidor, los productos instalados y el sistema operativo instalado dentro de las respuestas de error en HTTP. Las respuestas a los errores pueden configurarse y personalizarse a nivel global, por sitio o por directorio en el apache2.conf usando la Directiva ErrorDocument. Los administradores del sitio pueden gestionar sus propios errores con el archivo .htaccess si la directiva global AllowOverride está configurada correctamente en apache2.conf.

Manejo de errores en Tomcat

Tomcat es un servidor HTTP para alojar aplicaciones JSP y Java Servlet. Por defecto Tomcat muestra la versión del servidor en las respuestas de error HTTP. La personalización de las respuestas de error se puede configurar en el documento de configuración web.xml.

7.8.2 Análisis de trazas de pila de datos (OTG-ERR-002)

Descripción

Los rastros de pilas de datos no son vulnerabilidades por sí mismas, pero a menudo revelan información que es interesante para un atacante. Los atacantes intentan generar estos rastros de pila de datos mediante la alteración del ingreso a la aplicación web con peticiones HTTP con formato incorrecto y otros datos de ingreso. Si la aplicación responde con rastros de pilas de datos que no se manejan, podría revelar información útil a los atacantes. Esta información podría utilizarse en otros ataques. Proporcionar información de depuración como resultado de las operaciones que generan errores se considera una mala práctica por varias razones. Por ejemplo, puede contener información sobre el funcionamiento interno de la aplicación como rutas relativas del punto donde está instalada la aplicación o como se referencian los objetos internamente.

Hay una variedad de técnicas que pueden provocar que mensajes de excepción se envíen en una respuesta HTTP. Tenga en cuenta que en la mayoría de los casos se trata de una página HTML, pero las excepciones se pueden enviar como parte de las respuestas SOAP o REST también.

Algunas pruebas que se deben incluir son:

- ✓ Entrada no válida (como la entrada que no es coherente con la lógica de la aplicación).
- ✓ Las entradas que contienen caracteres no alfanuméricos o consultas de sintaxis.
- ✓ Entradas vacías.
- ✓ Entradas que son muy largas.

- ✓ Acceso a páginas internas sin autenticación.
- ✓ Esquivar el flujo de la aplicación.

Todas las pruebas anteriores podrían llevar a errores de aplicación que pueden contener restos de pila de datos. Se recomienda utilizar un comprobador aleatorio adicional a cualquier prueba manual.

Herramientas

- ✓ Zap proxy.

7.9 Criptografía

7.9.1 Pruebas de codificadores SSL/TLS débiles, protección de transporte de capas insuficiente (OTG-CRYPST-001)

Descripción

Los datos sensibles deben ser protegidos cuando se transmiten a través de la red. Dichos datos pueden incluir credenciales y tarjetas de crédito del usuario. Como regla general, si los datos se deben proteger cuando se almacenan, se deben proteger también durante la transmisión.

HTTP es un protocolo de texto y normalmente se asegura mediante un túnel SSL/TLS, dando como resultado tráfico en HTTPS. El uso de este protocolo asegura no sólo confidencialidad, sino también la autenticación. Los servidores se autentican mediante certificados digitales y también es posible utilizar el certificado de cliente para una autenticación mutua.

Aunque los cifrados de alto nivel se apoyan hoy en día y son utilizados normalmente, algún error de configuración en el servidor puede utilizarse para forzar el uso de un cifrado débil o, en el peor caso, ningún cifrado permitiendo a un atacante acceder al canal de comunicación supuestamente seguro. Otras configuraciones erróneas pueden usarse para un ataque de negación de servicio.

Asuntos comunes

Se produce una vulnerabilidad si se utiliza el protocolo HTTP para

transmitir información sensible (credenciales transmitidas en HTTP). La presencia de un servicio SSL/TLS es buena, pero aumenta la superficie de ataque y pueden existir las siguientes vulnerabilidades:

- ✓ Los protocolos SSL/TLS, cifrados, claves y renegociación, deben estar correctamente configurados.
- ✓ La validez del certificado debe estar garantizada.

Otras vulnerabilidades vinculadas a esto son:

- ✓ Debe actualizarse el software expuesto debido a la posibilidad de vulnerabilidades conocidas.
- ✓ Use banderas de seguridad para las Cookies de sesión.
- ✓ El uso de Seguridad estricta de transporte HTTP (HSTS).
- ✓ La presencia tanto de HTTP como HTTPS, lo cual se puede utilizar también para interceptar tráfico.
- ✓ La presencia de contenido mezclado de HTTPS y HTTP en la misma página, lo cual puede usarse para filtrar información.

Datos sensibles transmitidos en texto transparente

La aplicación no debe transmitir información sensible a través de canales sin codificar. Normalmente es posible encontrar la autenticación básica en HTTP, contraseña de ingreso o la cookie de sesión enviada a través de HTTP y, en general, otra información considerada sensible por las regulaciones, leyes o políticas organizacionales.

SSL/TLS Ciphers/Protocols/Keys débiles

Históricamente, ha habido limitaciones determinadas por el gobierno de Estados Unidos, que permiten la exportación del cifrado sólo de un tamaño de hasta 40 bits, una longitud de clave que podría ser rota y permitiría el descifrado de comunicaciones. Desde entonces, las regulaciones de exportación criptográfica se han allanado a un tamaño de clave máximo de 128 bits.

Es importante comprobar que la configuración de SSL ha sido usada para evitar la puesta en marcha del soporte criptográfico que podría ser

fácilmente derrotado. Para alcanzar este objetivo, los servicios basados en SSL no deberían ofrecer la posibilidad de escoger una suite de cifrado débil. Una suite de cifrado se especifica mediante un protocolo de cifrado (por ejemplo, DES, RC4, AES), la longitud de cifrado de clave (por ejemplo, 40, 56 o 128 bits) y un algoritmo de hash (SHA, MD5) utilizado para verificar la integridad.

Brevemente, los puntos clave para la determinación de la suite de cifrado son las siguientes:

- ✓ [1] El cliente envía al servidor un mensaje ClientHello, especificando, entre otros datos, el protocolo y las suites de cifrado que puede manejar. Tome en cuenta que un cliente es generalmente un navegador web (actualmente, el más popular es un cliente SSL), pero no necesariamente, ya que puede ser
- ✓ cualquier aplicación habilitada para SSL; lo mismo se aplica para el servidor, que puede no ser un servidor web, aunque este es el caso más común.
- ✓ [2] El servidor responde con un mensaje de ServerHello, que contiene la suite de protocolo y algoritmo de cifrado elegidos que serán utilizados para esta sesión (generalmente el servidor selecciona la suite de protocolo y algoritmo de cifrado más fuerte que soporta tanto el cliente como el servidor). Es posible (por ejemplo, por medio de las directivas de configuración) especificar qué suites de cifrado el servidor obedecerá. De esta manera, usted puede controlar si las conversaciones con los clientes aceptarán solamente un cifrado de 40 bits.
- ✓ [1] El servidor envía su mensaje de certificado y, si requiere autenticación del cliente, también envía un mensaje de CertificateRequest al cliente.
- ✓ [2] El servidor envía un mensaje ServerHelloDone y espera una respuesta del cliente.
- ✓ [3] Al recibir el mensaje de ServerHelloDone, el cliente verifica la validez del certificado digital del servidor.

Validez del certificado SSL – cliente y servidor

Al acceder a una aplicación web mediante el protocolo HTTPS, se establece un canal seguro entre el cliente y el servidor. Luego se establece la identidad de uno (el servidor) o de ambas partes (cliente y servidor) por medio de certificados digitales. Así, una vez determinada la suite de cifrado, el "SSL Handshake" continúa con el intercambio de los certificados:

- ✓ [1] El servidor envía su mensaje de certificado y, si se requiere autenticación de cliente, también envía al cliente un mensaje de CertificateRequest.
- ✓ [2] El servidor envía un mensaje ServerHelloDone y espera una respuesta del cliente.
- ✓ [3] Al recibir el mensaje de ServerHelloDone, el cliente verifica la validez del certificado digital del servidor.

Herramientas

- ✓ Qualys SSL Labs - SSL Server Test.
- ✓ Nessus vulnerability scanner.
- ✓ Test SSL server.
- ✓ Sslyze.
- ✓ Ssl audit.
- ✓ Ssl scan.
- ✓ Nmap.

7.9.2 Prueba de padding oracle (OTG-CRYPST-002)

Descripción

Un padding oracle es una función de la aplicación que decodifica datos encriptados que proporciona el cliente, por ejemplo, estado de sesión interna almacenado en el cliente y fuga el estado de la validez del padding después de descifrado. La existencia de un padding oracle permite a un atacante descifrar datos encriptados y cifrar datos arbitrarios sin conocer la clave utilizada para estas operaciones criptográficas. Esto puede llevar a la fuga de datos sensibles o a una vulnerabilidad de privilegios escalada si la aplicación asume la integridad de los datos cifrados.

Los bloques de cifrado encriptan los datos en bloques de tamaños determinados. Los tamaños de bloque utilizados por los cifradores comunes son de 8 y 16 bytes. Los datos cuyo tamaño no coincide con un múltiplo del tamaño del bloque de cifrado usado, tienen que rellenarse de una forma específica para que el decodificador pueda eliminar el padding. Un esquema común de padding utilizado es PKCS #7. Llena los bytes restantes con el valor de la longitud del padding.

Herramientas

- ✓ PadBuster.
- ✓ Python-paddingoracle.
- ✓ Poracle.
- ✓ Padding Oracle Exploitation Tool.

7.9.3 Prueba para el envió de información sensible por canales sin encriptar (OTG-CRYPST-003)

Descripción

Los datos sensibles deben ser protegidos al transmitirse a través de la red. Si los datos se transmiten a través de HTTPS o cifrados de alguna otra manera, el mecanismo de protección no debe tener limitaciones o vulnerabilidades, como se explica en el artículo más amplio. Pruebas de codificadores SSL/TLS débiles, protección de transporte de capas insuficiente (OTG-CRYPST-001).

Como regla general, si los datos deben protegerse cuando se almacenan, estos datos también deben protegerse durante la transmisión. Algunos ejemplos de datos sensibles son:

- ✓ Información utilizada en la autenticación (por ejemplo, credenciales, PINs, identificadores de sesión, Fichas, Cookie)
- ✓ Información protegida por leyes, regulaciones o políticas organizacionales específicas (por ejemplo, tarjetas de crédito, datos del cliente).

Si la aplicación transmite información sensible a través de canales sin

codificar, por ejemplo, HTTP se considera un riesgo de seguridad. Algunos ejemplos son de autenticación básica que envía credenciales de autenticación en texto simple mediante HTTP, credenciales de autenticación basadas en formularios enviados mediante HTTP, o la transmisión de texto simple o cualquier otra información considerada sensible de acuerdo a normas, leyes, política organizacional o lógica de la aplicación del negocio.

Herramientas

- ✓ Curl puede ser usado para revisar como buscar páginas manualmente.

7.10 Pruebas de lógica del negocio

7.10.1 Prueba de la validación de datos de la lógica del negocio (OTG-BUSLOGIC-001)

Descripción

En la prueba de validación de datos de la lógica del negocio, verificamos que la aplicación no permita a los usuarios insertar datos "no validados" en el sistema o la aplicación. Esto es importante porque, sin esta protección, los atacantes pueden insertar datos/información "no validada" en la aplicación/sistema en "puntos de entrega" donde el sistema/aplicación considera que los datos/información es "buena" y ha sido válida desde que los puntos de "entrada" realizaron la validación de datos como parte del flujo de trabajo de la lógica de negocio.

7.10.2 Prueba de la habilidad para manipular consultas (OTG-BUSLOGIC-002)

Descripción

En las pruebas de requerimientos de parámetros predictivos y manipulados, verificamos que la aplicación no permita a los usuarios enviar o modificar los datos de cualquier componente del sistema al que ellos no deben tener acceso, que estén accediendo en ese momento o de esa manera en particular. Esto es importante porque, sin esta protección, los atacantes pueden ser capaces de "engañar/trucar" la aplicación para que les permita entrar en secciones de la aplicación del sistema a las cuáles no deberían tener

acceso en ese momento en particular, eludiendo así el flujo de trabajo de la lógica de negocio de la aplicación.

7.10.3 Prueba de comprobación de integridad (OTG-BUSLOGIC-003)

Descripción

En la comprobación de integridad y prueba de evidencia de adulteración, verificamos que la aplicación no permita a los usuarios destruir la integridad o datos de cualquier parte del sistema. Esto es importante porque, sin estas protecciones, los atacantes pueden romper el flujo de trabajo de la lógica del negocio y cambiar o poner en peligro los datos del sistema de aplicación o encubrir acciones mediante la alteración de información, incluyendo archivos de registro.

7.10.4 Prueba del tiempo de procesamiento (OTG-BUSLOGIC-004)

Descripción

En las pruebas del tiempo de procesamiento, verificamos que la aplicación no permita a los usuarios manipular un sistema o adivinar su comportamiento basado en los tiempos de procesamiento de entrada o salida. Esto es importante porque, sin esta protección, los atacantes pueden ser capaces de controlar el tiempo de procesamiento y determinar las salidas basados en el tiempo o eludir la lógica del negocio de la aplicación al no completar transacciones o acciones en forma oportuna.

7.10.5 Prueba del número de veces que limita el uso de una función (OTG-BUSLOGIC-005)

Descripción

Al probar el límite de una función, verifique que la aplicación no permita a los usuarios utilizar partes de la aplicación o sus funciones, más veces de las requeridas por el flujo de la lógica de trabajo. Esto es importante porque, sin esta protección, los atacantes pueden utilizar una función o parte de la aplicación un mayor número de veces que el permitido por la lógica de negocio para obtener beneficios adicionales.

7.10.6 Pruebas para la evasión de los flujos de trabajo (OTG-BUSLOGIC-006)

Descripción

Al evadir el flujo de trabajo y burlar las pruebas de secuencia correcta, verificamos que la aplicación no permite a los usuarios realizar acciones fuera del flujo de proceso de negocios "aprobado/requerido". Esto es importante ya que, sin esta protección, los atacantes pueden ser capaces de evadir o burlar los flujos de trabajo y "controles" que les permite entrar prematuramente o saltarse las secciones de la aplicación "requeridas", y potencialmente permite que la acción/transacción termine sin completar el proceso de negocio, dejando al sistema con información de seguimiento incompleta en el backend.

7.10.7 Prueba de las defensas contra el mal uso de la aplicación (OTG-BUSLOGIC-007)

Descripción

En las pruebas de defensas contra el mal uso de la aplicación, verificamos que la aplicación no permita a los usuarios manipular la aplicación de una manera no deseada.

7.10.8 Prueba de la posibilidad de carga de tipos de archivos inesperados (OTG-BUSLOGIC-008)

Descripción

En la prueba de carga de tipos de archivos inesperados, verificamos que la aplicación no permita a los usuarios cargar tipos de archivos que el sistema no espera o requiera de acuerdo a la lógica del negocio. Esto es importante ya que, sin esta protección, los atacantes pueden ser capaces de enviar archivos inesperados tales como .exe o .php que se guardan en el sistema y luego se ejecutan contra el sistema o aplicación.

7.10.9 Prueba de la posibilidad de carga de archivos maliciosos (OTG-BUSLOGIC-009)

Descripción

Al probar la carga de archivos maliciosos, verifique que la aplicación no permita a los usuarios cargar archivos maliciosos o potencialmente dañinos al sistema y a su seguridad. Esto es importante ya que, sin esta protección, los atacantes pueden ser capaces de cargar archivos al sistema que pueden propagar virus, malware o incluso explotaciones como shellcode cuando se ejecutan.

Herramientas

Aunque existen herramientas para probar y verificar que los procesos del negocio funcionan correctamente en situaciones válidas, estas herramientas son incapaces de detectar vulnerabilidades lógicas. Por ejemplo, las herramientas no tienen posibilidad de detectar si un usuario es capaz de evitar el flujo de proceso del negocio a través de la edición de parámetros, predicción de nombres de recursos o escalada de privilegios para acceder a recursos restringidos ni tienen ningún mecanismo para ayudar a los evaluadores humanos para que sospechen de esta situación. Los siguientes son algunos tipos comunes de herramientas que pueden ser útiles en la identificación de temas relacionados con la lógica del negocio.

- ✓ HP Business Process Testing Software.

Intercepting Proxy - Para observar los bloques de solicitud y respuesta del tráfico HTTP.

- ✓ OWASP ZAP.
- ✓ Burp Proxy.
- ✓ Paros Proxy.

Web Browser Plug-ins - Para visualizar y modificar encabezados HTTP/HTTPS, parámetros de publicación y observar el DOM del navegador.

- ✓ Tamper Data.
- ✓ TamperIE.
- ✓ Firebug.

Herramientas de pruebas misceláneas

- ✓ Web Developer toolbar.

La extensión Web Developer añade un botón a la barra de herramientas en el navegador, con varias herramientas de desarrollo web. Este es el puerto oficial de la extención de Web Developer para Firefox.

- ✓ HTTP Request Maker.

Request Maker es una herramienta para pruebas de penetración. Con ésta usted puede capturar fácilmente solicitudes realizadas por páginas web, manipular la URL, encabezados, datos POST y, por supuesto, realizar nuevas solicitudes.

- ✓ Cookie Editor.

Edit This Cookie es un administrador de cookies. Usted puede añadir, borrar, editar, buscar, proteger y bloquear cookies.

- ✓ Session Manager.

Con Session Manager usted puede grabar rápidamente el estado de su navegador actual y recargarlo cuando sea necesario. Puede gestionar múltiples sesiones, renombrarlas o removerlas de la biblioteca de sesión. Cada sesión recuerda el estado del navegador en su momento de creación, es decir, las ventanas y pestañas abiertas. Una vez que se abre una sesión, el navegador se restaura a su estado original.

7.11 Pruebas del punto de vista del cliente

7.11.1 Prueba del cross site scripting basado en DOM (OTG-CLIENT-001)

Descripción

Cross site scripting basado en DOM es el nombre de facto para errores XSS que son el resultado del contenido activo del lado del navegador en una página, generalmente JavaScript, que obtiene ingresos del usuario y luego hace algo inseguro, lo que lleva a la ejecución de código inyectado. Este documento sólo habla de errores de JavaScript que conducen a XSS.

El DOM o Modelo de Objeto del Documento (Document Object Model) es el formato estructural utilizado para representar documentos en un navegador. El DOM permite scripts dinámicos como JavaScript para referenciar los componentes del documento como un campo de formulario o una cookie de sesión. El DOM también se utiliza en el navegador por seguridad, por ejemplo, para limitar a los scripts de dominios distintos el obtener las cookies de sesión desde otros dominios. Una vulnerabilidad XSS basada en DOM ocurre cuando se modifica el contenido activo, como una función de JavaScript; se modifica con una solicitud especialmente diseñada, tal como un elemento DOM que puede ser controlado por un atacante.

Recursos OWASP

- ✓ DOM based XSS Prevention Cheat Sheet.

7.11.2 Prueba de la ejecución de javascript (OTG-CLIENT-002)

Descripción

Una vulnerabilidad de inyección de JavaScript es un subtipo de Cross Site Scripting (XSS) que implica la capacidad para inyectar código JavaScript arbitrario que ejecuta la aplicación dentro del navegador de la víctima. Esta vulnerabilidad puede tener muchas consecuencias, como la divulgación de las cookies de sesión de un usuario, que podría ser utilizada para suplantar la identidad de la víctima o, más generalmente, puede permitir al atacante modificar el contenido de la página vista por la víctima o el comportamiento de la aplicación.

7.11.3 Prueba de inyección html (OTG-CLIENT-003)

Descripción

La inyección HTML es un tipo de inyección que se produce cuando un usuario es capaz de controlar un punto de entrada y puede inyectar código HTML arbitrario en una página web vulnerable. Esta vulnerabilidad puede tener muchas consecuencias, como la divulgación de las cookies de sesión de un usuario que podrían ser utilizadas para suplantar la identidad de la víctima o, más comúnmente, puede permitir al atacante modificar el contenido

de la página vista por las víctimas.

7.11.4 Prueba de redireccionamiento de la url del lado cliente (OTG-CLIENT-004)

Descripción

Comprobar el redireccionamiento de URL del lado cliente, también conocido como redirección abierta. Es un error de validación de entrada que se da cuando una aplicación acepta un ingreso controlado por el usuario, que especifica un enlace que lleva a una URL externa. Este tipo de vulnerabilidad podría utilizarse para realizar un ataque de phishing o redirigir a una víctima a una página de infección.

Herramientas

- ✓ DOMinator.

7.11.5 Pruebas de inyección de css (OTG-CLIENT-005)

Descripción

Una vulnerabilidad de inyección de CSS implica la capacidad de inyectar código CSS arbitrario en el contexto de un sitio web de confianza que se mostrará en el navegador de la víctima. El impacto de esta vulnerabilidad puede variar en función de la carga CSS suministrada: podría causar un Cross Site Scripting en circunstancias particulares, al robar datos realizando una extracción de los datos confidenciales o modificaciones de la interfaz del usuario.

7.11.6 Pruebas de la manipulación de recursos del lado del cliente (OTG-CLIENT-006)

Descripción

Una vulnerabilidad de manipulación de recursos del lado del cliente es un error de validación de los ingresos que se producen cuando una aplicación acepta las entradas controladas por un usuario que especifica la ruta hacia un recurso (por ejemplo, la fuente de un iframe, js, applet o el controlador de un XMLHttpRequest). Específicamente, esta vulnerabilidad consiste en la capacidad para controlar las direcciones URL que vinculan a

algunos recursos presentes en una página web. El impacto puede variar en función del tipo de elemento de la URL controlada por el atacante y, generalmente, se adopta para llevar a cabo ataques de Cross-Site Scripting.

Esta vulnerabilidad se produce cuando la aplicación emplea un URL controlado por el usuario para hacer referencia a recursos externos/internos. En estas circunstancias, es posible interferir con el comportamiento esperado de la aplicación, en el sentido de hacer que cargue y procese objetos maliciosos.

7.11.7 Pruebas para el intercambio de origen cruzado (OTG-CLIENT-007)

Descripción

La prueba de intercambio de recursos de origen cruzado o CORS es un mecanismo que permite a un navegador web realizar peticiones de "cross-domain" que utiliza la API XMLHttpRequest L2 de una manera controlada. En el pasado, la API XMLHttpRequest L1 sólo permitía que las solicitudes se enviaran dentro del mismo origen, ya que estaba restringida por la política del mismo origen.

Las solicitudes de origen cruzado tienen un encabezado de origen que identifica el dominio que inicia la petición y siempre se envía al servidor. CORS define el protocolo a utilizar entre un navegador web y un servidor para determinar si se permite una solicitud de origen cruzado. Con el fin de lograr este objetivo, hay algunos encabezados HTTP que participan en este proceso, que son compatibles con todos los navegadores que se detallan a continuación e incluyen: Origen, Acceso-Control-Solicitud-Método, Acceso-Control-Solicitud-Encabezados, Acceso-Control-Permiso-Origen, Acceso-Control-Permiso-Credenciales, Acceso-Control-Permiso-Métodos, Acceso-Control-Permiso-Encabezados.

Los mandatos de especificación CORS para las solicitudes no simples, como por ejemplo las solicitudes que no sean GET o POST o las solicitudes que utilizan credenciales, deben enviar una solicitud previa de OPTIONS para comprobar si el tipo de solicitud tendrá un impacto negativo en los datos. La solicitud previa al mandato comprueba los métodos, los

encabezados permitidos por el servidor y si se permiten las credenciales.

Herramientas

- ✓ OWASP Zed Attack Proxy (ZAP).

7.11.8 Pruebas de cross-site flashing (OTG-CLIENT-008)

Descripción

ActionScript es el lenguaje basado en ECMAScript, usado por las aplicaciones Flash cuando maneja necesidades interactivas. Hay tres versiones del lenguaje ActionScript. ActionScript 1.0 y ActionScript 2.0 son muy similares con ActionScript 2.0 que es una extensión de ActionScript 1.0. ActionScript 3.0, introducido con Flash Player 9, es una reescritura del lenguaje para apoyar el diseño orientado por objetos.

ActionScript, como cualquier otro lenguaje, tiene algunos patrones de implementación que podrían llevar a problemas de seguridad. En particular, dado que las aplicaciones Flash a menudo se incrustan en los navegadores, las vulnerabilidades como las basadas en DOM para Cross-Site Scripting (XSS) podrían estar presentes en las aplicaciones Flash defectuosas.

Redirectores abiertos

Los SWF tienen la capacidad de navegar con el explorador. Si el SWF toma el destino como un FlashVar, entonces el SWF puede ser utilizado como un redirector abierto. Un redirector abierto es cualquier pieza de funcionalidad en un sitio web de confianza, que un atacante puede utilizar para redirigir al usuario a un sitio web malicioso. Estos se utilizan con frecuencia dentro de los ataques de phishing. Al igual que en cross-site scripting, el ataque implica que un usuario final haga clic en un enlace malicioso.

Herramientas

- ✓ Adobe SWF Investigator.
- ✓ SWFScan.
- ✓ SWFIntruder.

- ✓ Decompiler – Flare.
- ✓ Compiler – MTASC.

7.11.9 Pruebas de clickjacking (OTG-CLIENT-009)

Descripción

Clickjacking es una técnica maliciosa que consiste en engañar a un usuario web para que interactúe (en la mayoría de los casos haciendo clic) con algo diferente a lo que el usuario cree que está interactuando. Este tipo de ataque, que puede ser utilizado solo o en combinación con otros ataques, podría potencialmente enviar comandos no autorizados o revelar información confidencial mientras la víctima está interactuando con páginas web aparentemente inofensivas.

Un ataque de clickjacking utiliza características aparentemente inocuas de HTML y Javascript para obligar a la víctima a realizar acciones no deseadas tales como hacer clic en un botón que parece realizar otra operación. Se trata de un problema de seguridad "del lado del cliente", que afecta a una gran variedad de navegadores y plataformas. Para llevar a cabo este tipo de técnica, el atacante tiene que crear una página web aparentemente inofensiva que cargue la aplicación al objetivo de destino, mediante el uso de un iframe (convenientemente oculto mediante el uso de código CSS).

7.11.10 Pruebas de websockets (OTG-CLIENT-010)

Descripción

Tradicionalmente, el protocolo HTTP sólo permitía una solicitud/respuesta por conexión TCP. Asynchronous JavaScript y XML (AJAX) permiten a los clientes enviar y recibir datos de forma asíncrona (en segundo plano sin una actualización de la página) al servidor, sin embargo, AJAX requiere que el cliente inicie las peticiones y espere las respuestas del servidor (half-duplex). WebSockets HTML5 permiten al cliente / servidor crear canales de comunicación "full-duplex" (bidireccional), permitiendo que el cliente y el servidor puedan comunicarse verdaderamente en forma asíncrona. Los WebSockets conducen su 'actualización' handshake inicial a través de

HTTP y, de ahí, toda la comunicación se lleva a cabo a través de canales TCP mediante el uso de marcos.

Herramientas

- ✓ OWASP Zed Attack Proxy (ZAP).
- ✓ Cliente WebSocket.

7.11.11 Pruebas de mensajería web (OTG-CLIENT-011)

Descripción

Web Messaging (también conocido como Cross Document Messaging) permite a las aplicaciones que se ejecutan en diferentes dominios comunicarse de una manera segura. Antes de la introducción de la web de mensajería, la comunicación de diferentes orígenes (entre iframes, pestañas y ventanas) fue restringida por la política del mismo origen y puesta en vigor por el navegador; sin embargo, los desarrolladores utilizan varios cortes con el fin de llevar a cabo estas tareas, la mayoría de ellas principalmente inseguras.

Esta restricción en el navegador está colocada para evitar que un sitio web malicioso pueda leer datos confidenciales de otros iframes, tabs, etc, sin embargo, hay algunos casos donde dos legítimos sitios web de confianza necesitan intercambiar datos entre sí. Para satisfacer esta necesidad, se introdujo Cross Document Messaging dentro del borrador de la especificación WHATWG HTML5 y se implementó en todos los principales navegadores. Esto permitió una comunicación segura entre múltiples orígenes a través de iframes, pestañas y ventanas.

Para satisfacer esta necesidad, se introdujo Cross Document Messaging dentro del borrador de la especificación WHATWG HTML5 y se implementó en todos los principales navegadores. Esto permitió una comunicación segura entre múltiples orígenes a través de iframes, pestañas y ventanas. La API Messaging introdujo el método postMessage (), con la que los mensajes de texto sin formato se pueden enviar a través de cross-origin. Se compone de dos parámetros: el mensaje y el dominio. Hay algunos problemas de seguridad cuando se utiliza '*' como el dominio que se discute

a continuación. Entonces, con el fin de recibir mensajes, el sitio web receptor tiene que añadir un nuevo controlador de eventos y tener los siguientes atributos:

- ✓ Datos: Contenido del mensaje entrante.
- ✓ Origen: Origen del documento emisor.
- ✓ Fuente: ventana de la fuente.

Herramientas

- ✓ OWASP Zed Attack Proxy (ZAP).

7.11.12 Prueba de almacenamiento local (OTG-CLIENT-012)

Descripción

Almacenamiento local, también conocido como Web Storage o Offline Storage, es un mecanismo para almacenar los datos como pares clave/valor atados a un dominio y forzados por la política del mismo origen (SOP). Hay dos objetos: localStorage que es persistente y está destinado a sobrevivir los reinicios del navegador/sistema; y sessionStorage que es temporal y sólo existe hasta que se cierre la ventana o pestaña.

En promedio, los navegadores permiten guardar en este almacenamiento alrededor de 5 MB por cada dominio. Comprarados con el de 4 KB de cookies es una gran diferencia, pero la diferencia clave desde la perspectiva de seguridad es que los datos almacenados en estos dos objetos se mantienen en el cliente y nunca se envían al servidor. Esto también mejora el rendimiento de la red ya que los datos no tienen que ir y volver a través del cable.

Almacenamiento local

El acceso al almacenamiento se realiza normalmente utilizando las funciones SetItem y GetItem. El almacenamiento se puede leer desde JavaScript, lo que significa que con una sola XSS un atacante sería capaz de extraer todos los datos desde el almacenamiento. Además, se pueden cargar datos maliciosos en el almacenamiento a través de JavaScript, por lo que la aplicación necesita tener los controles para el tratamiento de datos no

confiables. Compruebe si hay más de una aplicación en el mismo dominio, tales como example.foo/app1 y example.foo/app2, porque esas compartirán el mismo almacenamiento.

Los datos almacenados en este objeto persistirán después que la ventana está cerrada. No es buena idea almacenar datos sensibles o identificadores de sesión en este objeto, ya que se puede acceder a ellos a través de JavaScript. Datos de sesión ID almacenados en las cookies pueden mitigar este riesgo mediante el indicador HTTPOnly.

Session Storage

La principal diferencia de localStorage es que se puede acceder a los datos almacenados en este objeto sólo hasta que la pestaña / ventana se cierra, lo que lo hace un candidato perfecto para los datos que no necesitan persistir entre sesiones. Ya que comparte la mayoría de las propiedades y los métodos getItem / setItem, la prueba manual debe llevarse a cabo para buscar estos métodos e identificar en qué partes del código se accede al almacenamiento.

Herramientas

- ✓ Firebug.
- ✓ Google Chrome Developer Tools.
- ✓ OWASP Zed Attack Proxy (ZAP).

8. Empresas encuestadas

Las empresas encuestadas fueron las siguientes:

- ✓ Empresa Distribuidora de Limpieza Clean S.A: es una distribuidora de productos y artículos de limpieza.
- ✓ Supermercado Zea: es una cadena de supermercados de insumos de primera necesidad.
- ✓ Poseidón S.A: empresa dedicada a la comercialización de pescados y mariscos, frescos y congelados, tanto importados como nacionales.
- ✓ Vinoteca S.R.L: es una empresa que exporta e importa,

representa, distribuye vinos, licores.

- ✓ Todo deportes S.A: empresa dedicada a la comercialización de calzado e indumentaria deportiva

Con la encuesta se va medir el cumplimiento de pruebas de seguridad durante el ciclo de vida del software.

8.1 Encuesta enviada a las empresas evaluadas

A continuación, veremos la encuesta con los controles que fueron analizadas por las empresas que conformaron la evaluación.

Information Gathering	Test Name	Description	Tools	Result	Remark
OTG-INFO-001	Conduct Search Engine Discovery and Reconnaissance for Information Leakage	*Use a search engine to search for Network diagrams and Configurations, Credentials, Error message content.*	*Google Hacking, Sitedigger, Shodan, FOCA, Punkspider*	Not Started	
OTG-INFO-002	Fingerprint Web Server	*Find the version and type of a running web server to determine known vulnerabilities and the appropriate exploits. Using "HTTP header field ordering" and "Malformed requests test".*	*Httprint, Httprecon, Desenmascarame*	Not Started	
OTG-INFO-003	Review Webserver Metafiles for Information Leakage	*Analyze robots.txt and identify <META> Tags from website.*	*Browser, curl, wget*	Not Started	
OTG-INFO-004	Enumerate Applications on Webserver	*Find applications hosted in the webserver (Virtual hosts/Subdomain), non-standard ports, DNS zone transfers*	*Webhosting.info, dnsrecon, Nmap, fierce, Recon-ng, Intrigue*	Not Started	
OTG-INFO-005	Review Webpage Comments and Metadata for Information Leakage	*Find sensitive information from webpage comments and Metadata on source code.*	*Browser, curl, wget*	Not Started	
OTG-INFO-006	Identify application entry points	*Identify from hidden fields, parameters, methods HTTP header analysis*	*Burp proxy, ZAP, Tamper data*	Not Started	
OTG-INFO-007	Map execution paths through application	*Map the target application and understand the principal workflows.*	*Burp proxy, ZAP*	Not Started	
OTG-INFO-008	Fingerprint Web Application Framework	*Find the type of web application framework/CMS from HTTP headers, Cookies, Source code, Specific files and folders.*	*Whatweb, BlindElephant, Wappalyzer*	Not Started	
OTG-INFO-009	Fingerprint Web Application	*Identify the web application and version to determine known vulnerabilities and the appropriate exploits.*	*Whatweb, BlindElephant, Wappalyzer, CMSmap*	Not Started	
OTG-INFO-010	Map Application Architecture	*Identify application architecture including Web language, WAF, Reverse proxy, Application Server, Backend Database*	*Browser, curl, wget*	Not Started	

Configuration and Deploy Management Testing	Test Name	Description	Tools	Result	Remark
OTG-CONFIG-001	Test Network/Infrastructure Configuration	*Understand the infrastructure elements interactions, config management for software, backend DB server, WebDAV, FTP in order to identify known vulnerabilities.*	*Nessus*	Not Started	
OTG-CONFIG-002	Test Application Platform Configuration	*Identify default installation file/directory, Handle Server errors (40*,50*), Minimal Privilege, Software logging.*	*Browser, Nikto*	Not Started	
OTG-CONFIG-003	Test File Extensions Handling for Sensitive Information	*Find important file, information (.asa , .inc , .sql ,zip, tar, pdf, txt, etc)*	*Browser, Nikto*	Not Started	
OTG-CONFIG-004	Backup and Unreferenced Files for Sensitive Information	*Check JS source code, comments, cache file, backup file (.old, .bak, .inc, .src) and guessing of filename*	*Nessus, Nikto, Wikto*	Not Started	
OTG-CONFIG-005	Enumerate Infrastructure and Application Admin Interfaces	*Directory and file enumeration, comments and links in source (/admin, /administrator, /backoffice, /backend, etc), alternative server port (Tomcat/8080)*	*Burp Proxy, dirb, Dirbuster, fuzzdb, Tilde Scanner*	Not Started	
OTG-CONFIG-006	Test HTTP Methods	*Identify HTTP allowed methods on Web server with OPTIONS. Arbitrary HTTP Methods, HEAD access control bypass and XST*	*netcat, curl*	Not Started	
OTG-CONFIG-007	Test HTTP Strict Transport Security	*Identify HSTS header on Web server through HTTP response header.* *curl -s -D- https://domain.com/ | grep Strict*	*Burp Proxy, ZAP, curl*	Not Started	
OTG-CONFIG-008	Test RIA cross domain policy	*Analyse the permissions allowed from the policy files (crossdomain.xml/clientaccesspolicy.xml) and allow-access-from.*	*Burp Proxy, ZAP, Nikto*	Not Started	

Identity Management Testing	Test Name	Description	Tools	Result	Remark
OTG-IDENT-001	Test Role Definitions	*Validate the system roles defined within the application by creating permission matrix.*	*Burp Proxy, ZAP*	Not Started	
OTG-IDENT-002	Test User Registration Process	*Verify that the identity requirements for user registration are aligned with business and security requirements:*	*Burp Proxy, ZAP*	Not Started	
OTG-IDENT-003	Test Account Provisioning Process	*Determine which roles are able to provision users and what sort of accounts they can provision.*	*Burp Proxy, ZAP*	Not Started	
OTG-IDENT-004	Testing for Account Enumeration and Guessable User Account	*Generic login error statement check, return codes/parameter values, enumerate all possible valid userids (Login system, Forgot password)*	*Browser, Burp Proxy, ZAP*	Not Started	

OTG-IDENT-005	Testing for Weak or unenforced username policy	*User account names are often highly structured (e.g. Joe Bloggs account name is jbloggs and Fred Nurks account name is fnurks) and valid account names can easily be guessed.*	*Browser, Burp Proxy, ZAP*	Not Started	
OTG-IDENT-006	Test Permissions of Guest/Training Accounts	*Guest and Training accounts are useful ways to acquaint potential users with system functionality prior to them completing the authorisation process required for access.Evaluate consistency between access policy and guest/training account access permissions.*	*Burp Proxy, ZAP*	Not Started	
OTG-IDENT-007	Test Account Suspension/Resumption Process	*Verify the identity requirements for user registration align with business/security requirements. Validate the registration process.*	*Burp Proxy, ZAP*	Not Started	

Authentication Testing	**Test Name**	**Description**	**Tools**	**Result**	**Remark**
OTG-AUTHN-001	Testing for Credentials Transported over an Encrypted Channel	*Check referrer whether its HTTP or HTTPs. Sending data through HTTP and HTTPS.*	*Burp Proxy, ZAP*	Not Started	
OTG-AUTHN-002	Testing for default credentials	*Testing for default credentials of common applications, Testing for default password of new accounts.*	*Burp Proxy, ZAP, Hydra*	Not Started	
OTG-AUTHN-003	Testing for Weak lock out mechanism	*Evaluate the account lockout mechanism's ability to mitigate brute force password guessing. Evaluate the unlock mechanism's resistance to unauthorized account unlocking.*	*Browser*	Not Started	
OTG-AUTHN-004	Testing for bypassing authentication schema	*Force browsing (/admin/main.php, /page.asp?authenticated=yes), Parameter Modification, Session ID prediction, SQL Injection*	*Burp Proxy, ZAP*	Not Started	
OTG-AUTHN-005	Test remember password functionality	*Look for passwords being stored in a cookie. Examine the cookies stored by the application. Verify that the credentials are not stored in clear text, but are hashed. Autocompleted=off?*	*Burp Proxy, ZAP*	Not Started	
OTG-AUTHN-006	Testing for Browser cache weakness	*Check browser history issue by clicking "Back" button after logging out. Check browser cache issue from HTTP response headers (Cache-Control: no-cache)*	*Burp Proxy, ZAP, Firefox add-on CacheViewer2*	Not Started	
OTG-AUTHN-007	Testing for Weak password policy	*Determine the resistance of the application against brute force password guessing using available password dictionaries by evaluating the length, complexity, reuse and aging requirements of passwords.*	*Burp Proxy, ZAP, Hydra*	Not Started	
OTG-AUTHN-008	Testing for Weak security question/answer	*Testing for weak pre-generated questions, Testing for weak self-generated question, Testing for brute-forcible answers (Unlimited attempts?)*	*Browser*	Not Started	

OTG-AUTHN-009	Testing for weak password change or reset functionalities	*Test password reset (Display old password in plain-text?, Send via email?, Random token on confirmation email ?), Test password change (Need old password?), CSRF vulnerability ?*	*Browser, Burp Proxy, ZAP*	Not Started	
OTG-AUTHN-010	Testing for Weaker authentication in alternative channel	*Understand the primary mechanism and Identify other channels (Mobile App, Call center, SSO)*	*Browser*	Not Started	

Authorization Testing	**Test Name**	**Description**	**Tools**	**Result**	**Remark**
OTG-AUTHZ-001	Testing Directory traversal/file include	*dot-dot-slash attack (../), Directory traversal, Local File inclusion/Remote File Inclusion.*	*Burp Proxy, ZAP, Wfuzz*	Not Started	
OTG-AUTHZ-002	Testing for bypassing authorization schema	*Access a resource without authentication?, Bypass ACL, Force browsing (/admin/adduser.jsp)*	*Burp Proxy (Autorize), ZAP*	Not Started	
OTG-AUTHZ-003	Testing for Privilege Escalation	*Testing for role/privilege manipulate the values of hidden variables. Change some param groupid=2 to groupid=1*	*Burp Proxy (Autorize), ZAP*	Not Started	
OTG-AUTHZ-004	Testing for Insecure Direct Object References	*Force changing parameter value (?invoice=123 -> ?invoice=456)*	*Burp Proxy (Autorize), ZAP*	Not Started	

Session Management Testing	**Test Name**	**Description**	**Tools**	**Result**	**Remark**
OTG-SESS-001	Testing for Bypassing Session Management Schema	*SessionID analysis prediction, unencrypted cookie transport, brute-force.*	*Burp Proxy, ForceSSL, ZAP, CookieDigger*	Not Started	
OTG-SESS-002	Testing for Cookies attributes	*Check HTTPOnly and Secure flag, expiration, inspect for sensitive data.*	*Burp Proxy, ZAP*	Not Started	
OTG-SESS-003	Testing for Session Fixation	*The application doesn't renew the cookie after a successfully user authentication.*	*Burp Proxy, ZAP*	Not started	
OTG-SESS-004	Testing for Exposed Session Variables	*Encryption & Reuse of session Tokens vulnerabilities, Send sessionID with GET method ?*	*Burp Proxy, ZAP*	Not Started	
OTG-SESS-005	Testing for Cross Site Request Forgery	*URL analysis, Direct access to functions without any token.*	*Burp Proxy (csrf_token_detect), burpy, ZAP*	Not Started	

OTG-SESS-006	Testing for logout functionality	*Check reuse session after logout both server-side and SSO.*	*Burp Proxy, ZAP*	Not Started	
OTG-SESS-007	Test Session Timeout	*Check session timeout, after the timeout has passed, all session tokens should be destroyed or be unusable.*	*Burp Proxy, ZAP*	Not Started	
OTG-SESS-008	Testing for Session puzzling	*The application uses the same session variable for more than one purpose. An attacker can potentially access pages in an order unanticipated by the developers so that the session variable is set in one context and then used in another.*	*Burp Proxy, ZAP*	Not Started	

Data Validation Testing	**Test Name**	**Description**	**Tools**	**Result**	**Remark**
OTG-INPVAL-001	Testing for Reflected Cross Site Scripting	*Check for input validation, Replace the vector used to identify XSS, XSS with HTTP Parameter Pollution.*	*Burp Proxy, ZAP, Xenotix XSS*	Not Started	
OTG-INPVAL-002	Testing for Stored Cross Site Scripting	*Check input forms/Upload forms and analyze HTML codes, Leverage XSS with BeEF*	*Burp Proxy, ZAP, BeEF, XSS Proxy*	Not Started	
OTG-INPVAL-003	Testing for HTTP Verb Tampering	*Craft custom HTTP requests to test the other methods to bypass URL authentication and authorization.*	*netcat*	Not Started	
OTG-INPVAL-004	Testing for HTTP Parameter pollution	*Identify any form or action that allows user-supplied input to bypass Input validation and filters using HPP*	*ZAP, HPP Finder (Chrome Plugin)*	Not Started	
OTG-INPVAL-005	Testing for SQL Injection	*Union, Boolean, Error based, Out-of-band, Time delay.*	*Burp Proxy (SQLipy), SQLMap, Pangolin, Seclists (FuzzDB)*	Not Started	
	Oracle Testing	*Identify URLs for PL/SQL web applications, Access with PL/SQL Packages, Bypass PL/SQL Exclusion list, SQL Injection*	*Orascan, SQLInjector*	Not Started	
	MySQL Testing	*Identify MySQL version, Single quote, Information_schema, Read/Write file.*	*SQLMap, Mysqloit, Power Injector*	Not Started	
	SQL Server Testing	*Comment operator (- -), Query separator (;), Stored procedures (xp_cmdshell)*	*SQLMap, SQLninja, Power Injector*	Not Started	

	Testing PostgreSQL	*Determine that the backend database engine is PostgreSQL by using the :: cast operator. Read/Write file, Shell Injection (OS command)*	*SQLMap*	Not Started	
	MS Access Testing	*Enumerate the column through error-based (Group by), Obtain database schema combine with fuzzdb.*	*SQLMap*	Not Started	
	Testing for NoSQL injection	*Identify NoSQL databases, Pass special characters (' " \ ; { }), Attack with reserved variable name, operator.*	*NoSQLMap*	Not Started	
OTG-INPVAL-006	Testing for LDAP Injection	*/ldapsearch?user=* user=*user=*)(uid=*))(|(uid=* pass=password*	*Burp Proxy, ZAP*	Not Started	
OTG-INPVAL-007	Testing for ORM Injection	*Testing ORM injection is identical to SQL injection testing*	*Hibernate, Nhibernate*	Not Started	
OTG-INPVAL-008	Testing for XML Injection	*Check with XML Meta Characters ', " , <>, <!--/-->, &, <![CDATA[/]]>, XXE, TAG*	*Burp Proxy, ZAP, Wfuzz*	Not Started	
OTG-INPVAL-009	Testing for SSI Injection	*• Presense of .shtml extension • Check for these characters < ! # = / . " - > and [a-zA-Z0-9] • include String = <!--#include virtual="/etc/passwd" -->*	*Burp Proxy, ZAP*	Not Started	
OTG-INPVAL-010	Testing for XPath Injection	*Check for XML error enumeration by supplying a single quote (') Username: ' or '1' = '1 Password: ' or '1' = '1*	*Burp Proxy, ZAP*	Not Started	
OTG-INPVAL-011	IMAP/SMTP Injection	*• Identifying vulnerable parameters with special characters (i.e.: \, ', ", @, #, !, |) • Understanding the data flow and deployment structure of the client • IMAP/SMTP command injection (Header, Body, Footer)*	*Burp Proxy, ZAP*	Not Started	
OTG-INPVAL-012	Testing for Code Injection	*Enter OS commands in the input field. ?arg=1; system('id')*	*Burp Proxy, ZAP, Liffy, Panoptic*	Not Started	
	Testing for Local File Inclusion	*LFI with dot-dot-slash (../../), PHP Wrapper (php://filter/convert.base64-encode/resource)*	*Burp Proxy, fimap, Liffy*	Not Started	
	Testing for Remote File Inclusion	*RFI from malicious URL ?page.php?file=http://attacker.com/malicious_page*	*Burp Proxy, fimap, Liffy*	Not Started	

OTG-INPVAL-013	Testing for Command Injection	*Understand the application platform, OS, folder structure, relative path and execute OS commands on a Web server. %3Bcat%20/etc/passwd test.pdf+|+Dir C:*	*Burp Proxy, ZAP, Commix*	Not Started	
OTG-INPVAL-014	Testing for Buffer overflow	• *Testing for heap overflow vulnerability* • *Testing for stack overflow vulnerability* • *Testing for format string vulnerability*	*Immunity Canvas, Spike, MSF, Nessus*	Not Started	
	Testing for Heap overflow			Not Started	
	Testing for Stack overflow			Not Started	
	Testing for Format string			Not Started	
OTG-INPVAL-015	Testing for incubated vulnerabilities	*File Upload, Stored XSS , SQL/XPATH Injection, Misconfigured servers (Tomcat, Plesk, Cpanel)*	*Burp Proxy, BeEF, MSF*	Not Started	
OTG-INPVAL-016	Testing for HTTP Splitting/Smuggling	*param=foobar%0d%0aContent-Length:%200%0d%0a%0d%0aHTTP/1.1%20200%20OK%0d%0aContent-Type:%20text/html%0d%0aContent-Length:%2035%0d%0a%0d%0a<html>Sorry,%20System%20Down</html>*	*Burp Proxy, ZAP, netcat*	Not Started	

Error Handling	**Test Name**	**Description**	**Tools**	**Result**	**Remark**
OTG-ERR-001	Analysis of Error Codes	*Locate error codes generated from applications or web servers. Collect sensitive information from that errors (Web Server, Application Server, Database)*	*Burp Proxy, ZAP*	Not Started	
OTG-ERR-002	Analysis of Stack Traces	• *Invalid Input / Empty inputs* • *Input that contains non alphanumeric characters or query syn tax* • *Access to internal pages without authentication* • *Bypassing application flow*	*Burp Proxy, ZAP*	Not Started	

Cryptography	**Test Name**	**Description**	**Tools**	**Result**	**Remark**

OTG-CRYPST-001	Testing for Weak SSL/TSL Ciphers, Insufficient Transport Layer Protection	*Identify SSL service, Idectify weak ciphers/protocols (ie. RC4, BEAST, CRIME, POODLE)*	*testssl.sh, SSL Breacher*	Not Started	
OTG-CRYPST-002	Testing for Padding Oracle	*Compare the responses in three different states:* *• Cipher text gets decrypted, resulting data is correct.* *• Cipher text gets decrypted, resulting data is garbled and causes some exception or error handling in the application logic.* *• Cipher text decryption fails due to padding errors.*	*PadBuster, Poracle, python-paddingoracle, POET*	Not Started	
OTG-CRYPST-003	Testing for Sensitive information sent via unencrypted channels	*Check sensitive data during the transmission:* *• Information used in authentication (e.g. Credentials, PINs, Session identifiers, Tokens, Cookies…)* *• Information protected by laws, regulations or specific organizational* *policy (e.g. Credit Cards, Customers data)*	*Burp Proxy, ZAP, Curl*	Not Started	

Business logic Testing	**Test Name**	**Description**	**Tools**	**Result**	**Remark**
OTG-BUSLOGIC-001	Test Business Logic Data Validation	*• Looking for data entry points or hand off points between systems or software.* *• Once found try to insert logically invalid data into the application/system.*	*Burp Proxy, ZAP*	Not Started	
OTG-BUSLOGIC-002	Test Ability to Forge Requests	*• Looking for guessable, predictable or hidden functionality of fields.* *• Once found try to insert logically valid data into the application/system allowing the user go through the application/system against the normal busineess logic workflow.*	*Burp Proxy, ZAP*	Not Started	
OTG-BUSLOGIC-003	Test Integrity Checks	*•Looking for parts of the application/system (components i.e. For example, input fields, databases or logs) that move, store or handle data/information.* *• For each identified component determine what type of data/information is logically acceptable and what types the application/system should guard against. Also, consider who according to the business logic is allowed to insert, update and delete data/information and in each component.* *• Attempt to insert, update or edit delete the data/information values with invalid data/information into each component (i.e. input, database, or log) by users that .should not be allowed per the busines logic workflow.*	*Burp Proxy, ZAP*	Not Started	

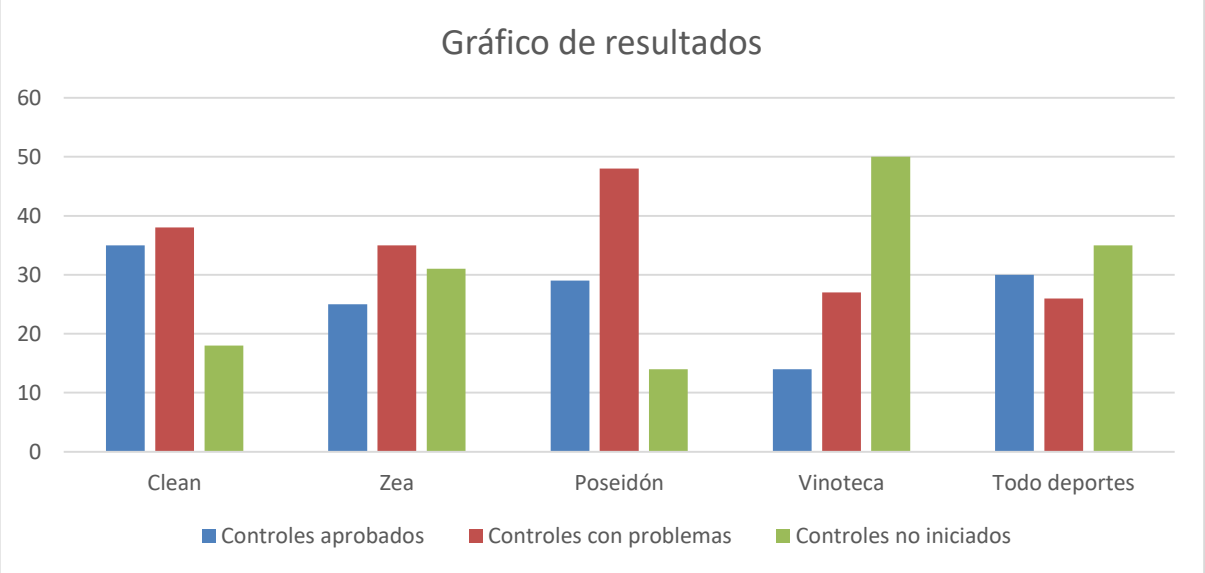
OTG-BUSLOGIC-004	Test for Process Timing	*• Looking for application/system functionality that may be impacted by time. Such as execution time or actions that help users predict a future outcome or allow one to circumvent any part of the business logic or workflow. For example, not completing transactions in an expected time.* *• Develop and execute the mis-use cases ensuring that attackers can not gain an advantage based on any timing.*	*Burp Proxy, ZAP*	Not Started	
OTG-BUSLOGIC-005	Test Number of Times a Function Can be Used Limits	*• Looking for functions or features in the application or system that should not be executed more that a single time or specified number of times during the business logic workflow.* *• For each of the functions and features found that should only be executed a single time or specified number of times during the business logic workflow, develop abuse/misuse cases that may allow a user to execute more than the allowable number of times.*	*Burp Proxy, ZAP*	Not Started	
OTG-BUSLOGIC-006	Testing for the Circumvention of Work Flows	*• Looking for methods to skip or go to steps in the application process in a different order from the designed/intended business logic flow.* *• For each method develop a misuse case and try to circumvent or perform an action that is "not acceptable" per the the business logic workflow.*	*Burp Proxy, ZAP*	Not Started	
OTG-BUSLOGIC-007	Test Defenses Against Application Mis-use	*Measures that might indicate the application has in-built self-defense:* *• Changed responses* *• Blocked requests* *• Actions that log a user out or lock their account*	*Burp Proxy, ZAP*	Not Started	
OTG-BUSLOGIC-008	Test Upload of Unexpected File Types	*• Review the project documentation and perform some exploratory testing looking for file types that should be "unsupported" by the application/system.* *• Try to upload these "unsupported" files an verify that it are properly rejected.* *• If multiple files can be uploaded at once, there must be tests in place to verify that each file is properly evaluated.* *PS. file.phtml, shell.phPWND, SHELL~1.PHP*	*Burp Proxy, ZAP*	Not Started	
OTG-BUSLOGIC-009	Test Upload of Malicious Files	*• Develop or acquire a known "malicious" file.* *• Try to upload the malicious file to the application/system and verify that it is correctly rejected.* *• If multiple files can be uploaded at once, there must be tests in place to verify that each file is properly evaluated.*	*Burp Proxy, ZAP*	Not Started	

Client Side Testing	Test Name	Description	Tools	Result	Remark
OTG-CLIENT-001	Testing for DOM based Cross Site Scripting	*Test for the user inputs obtained from client-side JavaScript Objects*	*Burp Proxy, DOMinator*	Not Started	
OTG-CLIENT-002	Testing for JavaScript Execution	*Inject JavaScript code: www.victim.com/?javascript:alert(1)*	*Burp Proxy, ZAP*	Not Started	
OTG-CLIENT-003	Testing for HTML Injection	*Send malicious HTML code: ?user=<img%20src='aaa'%20onerror=alert(1)>*	*Burp Proxy, ZAP*	Not Started	
OTG-CLIENT-004	Testing for Client Side URL Redirect	*Modify untrusted URL input to a malicious site: (Open Redirect) ?redirect=www.fake-target.site*	*Burp Proxy, ZAP*	Not Started	
OTG-CLIENT-005	Testing for CSS Injection	*Inject code in the CSS context :* • *www.victim.com/#red;-o-link:'javascript:alert(1)';-o-link-source:current; (Opera [8,12])* • *www.victim.com/#red;-:expression(alert(URL=1)); (IE 7/8)*	*Burp Proxy, ZAP*	Not Started	
OTG-CLIENT-006	Testing for Client Side Resource Manipulation	*External JavaScript could be easily injected in the trusted web site www.victim.com/#http://evil.com/js.js*	*Burp Proxy, ZAP*	Not Started	
OTG-CLIENT-007	Test Cross Origin Resource Sharing	*Check the HTTP headers in order to understand how CORS is used (Origin Header)*	*Burp Proxy, ZAP*	Not Started	
OTG-CLIENT-008	Testing for Cross Site Flashing	*Decompile, Undefined variables, Unsafe methods, Include malicious SWF (http://victim/file.swf?lang=http://evil*	*FlashBang, Flare, Flasm, SWFScan, SWF Intruder*	Not Started	
OTG-CLIENT-009	Testing for Clickjacking	*Discover if a website is vulnerable by loading into an iframe, create simple web page that includes a frame containing the target.*	*Burp Proxy, ClickjackingTool*	Not Started	
OTG-CLIENT-010	Testing WebSockets	*Identify that the application is using WebSockets by inspecting ws:// or wss:// URI scheme.Use Google Chrome's Developer Tools to view the Network WebSocket communication. Check Origin, Confidentiality and Integrity, Authentication, Authorization, Input Sanitization*	*Burp Proxy, Chrome, ZAP, WebSocket Client*	Not Started	
OTG-CLIENT-011	Test Web Messaging	*Analyse JavaScript code looking for how Web Messaging is implemented. How the website is restricting messages from untrusted domain and how the data is handled even for trusted domains*	*Burp Proxy, ZAP*	Not Started	
OTG-CLIENT-012	Test Local Storage	*Determine whether the website is storing sensitive data in the storage. XSS in localstorage http://server/StoragePOC.html#<img src=x onerror=alert(1)>*	*Chrome, Firebug, Burp Proxy, ZAP*	Not Started	

8.2 Resultados de la encuesta

Los resultados de la encuesta realizada fueron los siguientes:

Empresas evaluadas	Clean	Zea	Poseidón	Vinoteca	Todo deportes
Controles aprobados	35	25	29	14	30
Controles con problemas	38	35	48	27	26
Controles no iniciados	18	31	14	50	35



Los resultados muestran que todas las empresas evaluadas tienen falencias de seguridad por distintos motivos y no cumplen con las recomendaciones proporcionadas por el **Testing Guide de Owasp**, el problema que más predomina es la poca importancia a los temas de seguridad informática que se tienen desde la alta dirección para los proyectos de TI. En algunos casos los equipos de TI y seguridad informática no tienen los conocimientos de esta metodología ni conocimientos técnicos para poder realizar las pruebas de seguridad recomendadas por OWASP, en otros casos no existe el área de seguridad informática la cual brindaría el apoyo para realizar estas tareas. Por tal motivo el resultado muestra que las cinco organizaciones tienen mucho que mejorar desde su cultura organizacional respecto a temas de seguridad informática hasta brindar la capacitación necesaria para su equipo técnico y usuarios finales.

9. Conclusiones

En el presente trabajo se describieron los beneficios de aplicar la guía de pruebas de OWASP en el ciclo de desarrollo de software, así mismo se realizó la encuesta a 5 empresas para verificar el cumplimiento de estas buenas prácticas.

La Guía de pruebas o Testing Guide proporcionada por OWASP nos ofrece una metodología organizada y práctica para saber cómo organizar una auditoria del software de aplicaciones web en cada momento de desarrollo de la aplicación así como información con todas las posibles áreas que puedan suponer un vector de ataque para las aplicaciones. Este modelo se centra en ayudar a las organizaciones a probar sus aplicaciones Web para construir software confiable y seguro, para evitar pérdidas económicas, problemas legales entre otros.

Ya sea que se trate de sencillamente cumplir con los estándares o de proteger las aplicaciones críticas de su organización se recomienda siempre tener presente esta guía de pruebas y realizar la evaluación de los diversos controles que se fueron detallando en el presente trabajo al menos dos veces al año.

10. Bibliografía

10.1 Específica

[1] Chris Wysopal, Lucas Nelson, Dino Dai Zovi, Elfriede Dustin, The Art of Software Security Testing: Identifying Software Security Flaws, 2006.

[2] Gary McGraw and John Viega, Building Secure Software: How to Avoid Security Problems the Right Way, 2002.

[3] Enrique Rando, Amador Aparicio, Pablo Gonzáles, Ricardo Martín, Chema Alonso, Hacking web Technologies, 2016.

[4] Ron Lepofsky, The manager's guide to web application security, 2014.

[5] Joel Scambray, Web application security secrets and solutions third edition, 2010.

[6] Chris Mcnab, Network security assessment, 2016.

[7] Paco Hope, Ben Walther, Web security testing cookbook, 2008.

[8] Victor Marak, Windows malware analysis essentials, 2015.

[9] Monika Agarwal, Abhinav Singh, Metasploit penetration testing cookbook second edition, 2013.

[10] Monika Agarwal, Abhinav Singh, Metasploit penetration testing cookbook second edition, 2013.

[11] Cisco ccna security versión 2.0, 2018.

[12] Andrew Lockhart, Network security hacks second edition, 2006.

[13] Patrick Engebretson, The basics of hacking and penetration testing, 2011.

[14] William Stallings, Cryptography and network security fourth edition, 2005.

[15] Evan Gilman, Doug Barth, Zero trust networks first edition, 2017.

[16] Teri Bidwell, Michael Cross, Ryan Russell, Hack proofing your identity in the information age, 2002.

[17] Jeremiah Grossman, Robert Hansen, Anton Rager, Seth Fogie, D. Petkov, XSS Attacks, 2007.

[18] Stephen Lax, Access denied in the information age, 2001.

[19] Craig A. Schiller, Jim Binkley, David Harley, Gadi Evron, Tony Bradley, Tony Bradley, Carsten Willems, Michael Cross, Botnet the killer web app, 2007.

[20] Dennis Hansen, Social Vulnerability and Assessment Framework, 2017.

[21] Jay Kreps, I heart logs, 2014.

[22] James S. Tiller, The Ethical Hack: A Framework for Business Value Penetration Testing, 2005.

[23] Mark Graff and Kenneth R. Van Wyk, Secure Coding: Principles and Practices, 2003.

[24] Joel Scambray, Vinvent Liu,Caleb Sima, Hacking Exposed: Web Applications 3, 2010.