Guía: OWASP Top 10 – 2013

Leal Garcia, Luis Estiwar
luislealg@gmail.com
Universidad Piloto de Colombia

Resumen— Open Web Application Security Project (OWASP) es una fundación sin ánimo de lucro que reúne voluntarios interesados en la seguridad de aplicaciones, promoviendo recomendaciones para el desarrollo de software seguro, por medio de guías y herramientas.

"OWASP Top 10 – 2013" es una de las guías más reconocidas en el ámbito de la seguridad, está explica la clasificación de los diez riesgos más importantes que pueden llegar a afectar una aplicación web. Además describe los vectores de ataque, rutas de ataque y mejores prácticas para minimizar el riesgo.

Índice de Términos— OWASP, voluntarios, seguridad, aplicaciones, desarrollo, software, guías, herramientas, Top 10, clasificación, riesgo, web, vector de ataque, ruta de ataque, mitigar, vulnerabilidades.

Abstract — Open Web Application Security (OWASP) Project is a non-profit foundation that gathers volunteers interested in application security, promoting recommendations for the development of secure software, through guides and tools.

"OWASP Top 10 - 2013" is one of the most recognized in the field of security, therefore the classification of the ten most important risks that can affect a web application. Also describe the vectors of attack, routes of attack and best practices to minimize the risk.

Keywords— OWASP, voluntarios, seguridad, aplicaciones, desarrollo, software, guías, herramientas, Top 10, clasificación, riesgo, web, vector de ataque, ruta de ataque, mitigar, vulnerabilidades.

I. INTRODUCCIÓN

Hoy en día existen miles de aplicaciones web a nivel interno y externo en las organizaciones. A medida que pasa el tiempo cada vez más, las organizaciones y los individuos se preocupan por la información sensible que puede ser robada, eliminada, modificada o secuestrada por un atacante, perjudicando ya sea personalmente y/o económicamente al individuo u organización.

La mitigación de fallas de seguridad en las aplicaciones web es un reto difícil y muy importante de solucionar. No se puede dejar pasar por alto ejecutar pruebas de seguridad a las aplicaciones web antes de salir a un ambiente de producción, porque se estaría dejando la puerta abierta sin ser consciente de ello. La comprobación de la seguridad ha demostrado ser un elemento clave para cualquier organización que requiere confiar en el software que produce y usa.

Sin embargo, en la actualidad falta mayor prevención y concientización en el manejo de la seguridad de la información. Por lo tanto este artículo explica la clasificación de los diez riesgos más críticos en aplicaciones web propuestos por OWASP, los cuales ayudaran a entender que vectores de ataque se deben tener en cuenta y que recomendaciones se deben tomar, para mitigar o minimizar el impacto que pueden causar los atacantes en una organización.

II. INYECCIÓN (A1)

Una inyección de código es un hueco de seguridad en las aplicaciones web, que permite a un dato no fiable ser interpretado y ejecutado como parte de un comando o query (consulta). Esta debilidad es explotada por atacantes construyendo comandos maliciosos o queries (consultas) que pueden corromper, perder la información, denegar el servicio entre otros.

Las fallas de inyección a menudo se encuentran en códigos SQL, Xpath o noSQL, comandos SO, interpretes XML, encabezados SMTP, LDAP entre otros. El atacante puede fácilmente comprometer la aplicación web con esta debilidad leyendo, escribiendo, borrando y actualizando cualquier dato en el backend (por ejemplo la base de datos). [1] Es decir, los datos que ingresa el atacante pueden engañar al interprete en ejecutar comandos no intencionados o acceder a datos no autorizados. [2]

Hay muchos tipos de fallas de inyección, algunas de las más importantes son:

- SQL Injection (Inyección SQL): Es una de las vulnerabilidades más comunes de sitios web en internet. Está es una técnica que toma ventaja sobre las vulnerabilidades en la validación de los datos de entrada (comandos SQL), a través de una aplicación web para que sea ejecutado en el backend. [4]
- Command Injection (Inyección de Comandos): Este método implica inyectar código malicioso a través de una aplicación web para que sea ejecutado en el backend. [5]
- LDAP Injection (Inyección LDAP): Este método implica inyectar sentencias LDAP maliciosas a través de una aplicación web, para que sea ejecutado en el backend. [6]

Un sitio web es vulnerable a fallas de inyección de código, si durante el desarrollo web no se garantiza que los intérpretes separen la información no confiable (datos no esperados) de la confiable (datos esperados). Existen herramientas de análisis de código que pueden ayudar a entender cómo se utilizan los intérpretes y seguir el flujo de datos a través de la aplicación. [2]

A. Ruta de ataque

Se denomina ruta de ataque a las etapas que debe superar un atacante hasta llegar a su objetivo.

La ruta de ataque está compuesta por las siguientes fases con sus respectivos niveles de riesgo:

- Agentes de amenaza: Cualquiera que envíe información no confiable al sistema (usuarios internos, usuarios externos y administradores).
  Nivel de riesgo: Específico de la aplicación web.
- Vectores de ataque: El atacante envía ataques con cadenas simples de texto o cualquier fuente de datos, puede ser un vector de inyección.
  Nivel de riesgo: Explotabilidad fácil.
- Debilidades de seguridad: Las fallas de inyección son muy comunes particularmente en el código antiguo y son fáciles de descubrir al examinar el código, pero difíciles de descubrir por medio de pruebas. Los atacantes usan analizadores y fuzzers [3], para encontrar fallas de inyección en el código.
  Nivel de riesgo (Prevalencia de debilidades): Común.
  Nivel de riesgo (Detectabilidad de debilidades): Común.
- Impactos técnicos: Una inyección puede causar pérdida o corrupción de información, perdida de responsabilidad o denegación del servicio.
  Nivel de riesgo: Severo.
- Impacto al negocio: Se debe considerar el valor del negocio, de los datos afectados y la plataforma sobre la que corre el intérprete.
  Nivel de riesgo: Específico de la aplicación web y del negocio. [2]

B. Mitigación

Las formas de protección frente a estas debilidades son:

- Evitar que los datos no esperados sean validados y separados del backend.
- Uso de una API segura, para evitar el empleo de intérpretes o que provea una interfaz parametrizada.
- Si no hay disponibilidad de una API parametrizada, se debe codificar detalladamente los caracteres especiales por medio de rutinas de codificación.

La validación de entrada positiva o de lista blanca, no es tan eficiente y es la menos recomendada en solventar las fallas de inyecciones como los tres anteriores métodos. [2]

C. Escenario de ataque

Ejemplo: Se presenta cuando un usuario ingresa a una página web con credenciales e ingresa en los campos de usuario o contraseña la sentencia condicional con el tipo “Or 1=1″”, con el fin de acceder al sitio. [7]

III. PÉRDIDA DE AUTENTICACIÓN Y GESTIÓN DE SESIONES (A2)

Se refiere a fallas en las funcionalidades de autenticación con controles pobres o incorrectos

sobre la gestión de sesiones, permitiendo al atacante comprometer contraseñas, tokens de sesión o IDs de sesión, suplantando la identidad de un usuario legitimo sobre la aplicación web. [2]

La aplicación web es vulnerable si:

- Permite el uso de credenciales almacenadas sin cifrado o hash.
- Permite sobrescribir o adivinar credenciales a través de funciones débiles de gestión de sesión.
- Los IDs de sesión están expuestos en una URL.
- Los IDs de sesión no expiran.
- Los IDs se mantienen estáticos después de una autenticación exitosa.
- Permite transmitir contraseñas, credenciales o IDs de sesión a través de canales no cifrados. [2]

La manipulación de parámetros intercambiados entre cliente y servidor explotan vulnerabilidades de integridad y mecanismos de validación, que puede trascender en un ataque de secuencia de comandos en sitios cruzados (XSS), Inyección SQL, entre otros. [1]

A. Ruta de ataque

La ruta de ataque está compuesta por las siguientes fases con sus respectivos niveles de riesgo:

- Agentes de amenaza: Se deben considerar atacantes externos, como también usuarios con cuentas propias que podrían robar cuentas de otros usuarios y empleados queriendo enmascarar su identidad sobre acciones mal intencionadas.
  Nivel de riesgo: Específico de la aplicación web.
- Vectores de ataque: El atacante emplea vulnerabilidades en las funciones de autenticación y gestión de sesiones, por ejemplo IDs de sesiones, usuarios y contraseñas expuestas.
  Nivel de riesgo: Explotabilidad promedio.
- Debilidades de seguridad: Los desarrolladores a menudo crean diseños propios de autenticación o control de sesiones, pero crear diseños de forma correcta es arduo, por lo tanto dejan huecos de seguridad como: cierre de sesión, control de contraseñas, tiempo de desconexión inactiva (timeout), función de recordación de contraseña, pregunta secreta, actualización de cuenta, entre otros.
  Nivel de riesgo (Prevalencia de debilidades): Difundido.
  Nivel de riesgo (Detectabilidad de debilidades): Promedio.
- Impactos técnicos: Estas debilidades pueden permitir al atacante realizar suplantaciones de identidad, para ejecutar cualquier acción que la víctima tenga permiso. Las cuentas privilegiadas son objetivos prioritarios, para los atacantes.
  Nivel de riesgo: Severo.
- Impacto al negocio: Se debe considerar el valor del negocio, de los datos afectados o las funciones de la aplicación expuestas.
  Nivel de riesgo: Depende de la aplicación web y del negocio. [2]

B. Mitigación

Las formas de protección frente a estas debilidades son:

- Cifrar la comunicación al acceder a la aplicación.
- En lo posible eliminar la opción de recordar contraseña en el navegador web.
- Mantener la opción de cerrar sesión en la aplicación web.
- Emplear preferiblemente el método POST en vez del GET.
- Emplear expiración de la sesión (timeout).
- Cada cierto tiempo de mantener una sesión activa, la aplicación debería cambiar el valor del identificador de la sesión de forma aleatoria.

C. Escenario de ataque

Ejemplo: El usuario se autentica en un portal de compras por internet y ve una promoción de un televisor, copia la URL sin darse cuenta que la URL contiene el ID de su sesión y que soporta re-escritura. Luego comparte el link a sus amigos por correo sin saber que ellos podrían entrar al portal con la cuenta (de la víctima) en el sitio web:

http://micrompra.com/sale/saleitems;jsessionid=2P0OC2JDPXM0OQSNDLPSKHCJUN2JV?dest=TVLG55sitio

IV. SECUENCIA DE COMANDOS EN SITIOS CRUZADOS "XSS" (A3)

Es una debilidad típica en aplicaciones web, que permite a un atacante inyectar código malicioso (ejemplo: JavaScript), a sitios web visitados por el usuario, es decir la aplicación web toma los datos no confiables y los envía al navegador web sin una validación y codificación apropiada.

XSS es un vector de ataque que puede ser usado para robar información sensible, secuestrar IDs de sesión y comprometer el navegador web del usuario. Cualquier plataforma de aplicación web puede ser vulnerable a este tipo de ataque. [9]

Existen 3 tipos de secuencia de comandos en sitios cruzados:

- XSS reflejado: Es el más fácil de explotar y consiste en que una página refleja la misma información suministrada por el usuario al navegador web.
- XSS Almacenado: Almacena la información hostil en un sistema de almacenamiento y posteriormente muestra dicha información sin tener ningún tipo de filtrado o control.
- Inyección DOM: El código JavaScript y las variables del sitio web son manipuladas, en lugar de elementos HTML. [9]

Una aplicación web es vulnerable a XSS si no se asegura que todas las entradas de datos sean validadas como seguras o codificadas adecuadamente antes de ser incluidas en la página de salida; sin esto, los datos de entrada serán tratados como contenido activo en el navegador web. [2]

A. Ruta de ataque

La ruta de ataque está compuesta por las siguientes fases con sus respectivos niveles de riesgo:

- Agentes de amenaza: Se debe considerar cualquier persona que pueda enviar datos no confiables al sistema, incluyendo usuarios internos, externos y administradores.
  Nivel de riesgo: Específico de la aplicación web.
- Vectores de ataque: El atacante envía ataques por medio de cadenas de texto (secuencia de comandos) que explota el intérprete del navegador web. La mayoría de fuentes de datos puede ser un vector de ataque, incluyendo fuentes de información interna como por ejemplo la base de datos.
  Nivel de riesgo: Explotabilidad promedio.
- Debilidades de seguridad: XSS es una debilidad predominante en la seguridad de aplicaciones web y estas fallas son detectadas relativamente fácil, a través de pruebas o por medio de un análisis de código.
  Nivel de riesgo (Prevalencia de debilidades): Muy difundida.
  Nivel de riesgo (Detectabilidad de debilidades): Fácil.
- Impactos técnicos: El atacante puede ejecutar secuencias de comandos en el navegador web de la víctima, para secuestrar las sesiones de usuario usando malware, alterando la apariencia del sitio web, insertando código hostil, redirigiendo usuarios, entre otros.
  Nivel de riesgo: Impacto moderado.
- Impacto al negocio: Se debe considerar el valor del sistema y del procesamiento de datos afectados para el negocio, y la exposición pública de esta vulnerabilidad.
  Nivel de riesgo: Específico de la aplicación web. [2]

B. Mitigación

Las formas de protección frente a estas debilidades son:

- Codificar los datos no seguros basados en el contexto HTML (cuerpo, atributo, JavaScript, CSS o URL), donde serán ubicados.
- Validar el ingreso de datos de entrada positiva (lista blanca), preferiblemente en lo posible validar el número y el tipo de los caracteres, el formato y el cumplimiento a reglas pre-establecidas.

- Considerar las bibliotecas de auto sanitización como AntiSamy o de HTML de Java, cuando exista contenido enriquecido.
- Emplear políticas de seguridad de contenido (CSP). [2]

C. Escenario de ataque

Ejemplo: Cuando se ingresa a una página web, como: http://www.example.com/test.html, encontramos que el código fuente de la página contiene lo siguiente:

<script>
document.write("<b>Current URL<b> : " + document.baseURI);
</script>

Un ataque de XSS basado en DOM a esta página web puede lograr insertar y enviar el siguiente Script sobre la URL:

http://www.example.com/test.html#<script>alert(1)</script>

Si se observa de nuevo el código fuente de la página, no se verá *<script>alert(1)</script>* porque esto pasa en la inyección DOM y es hecho cuando se ejecuta el JavaScript. [10]

V. REFERENCIA DIRECTA INSEGURA A OBJETOS (A4)

Es una debilidad disponible desde que el programador expone una referencia a un objeto interno de la aplicación web como un archivo, directorio, registro de base de datos, contraseña sobre una URL o un parámetro de un formulario web. Un atacante podría usar o manipular las referencias, para acceder a otros objetos sin autorización. [8]

Existen dos formas de comprobar que la aplicación no presenta ninguna referencia directa a un objeto interno:

- Enfoques automáticos: Son herramientas de análisis estático, no identifican que parámetros manipulables deben ser sometidos a un control de acceso.
- Enfoques manuales: Se revisa el código con el fin de rastrear el uso de parámetros manipulables y críticos, como también ejecutando pruebas de intrusión, para identificar parámetros posibles de modificar.

Ambas técnicas emplean demasiado tiempo y dejan posibilidades sin descubrir. [8]

A. Ruta de ataque

La ruta de ataque está compuesta por las siguientes fases con sus respectivos niveles de riesgo:

- Agentes de amenaza: Considerar los usuarios que tienen acceso al sistema.
  Nivel de riesgo: Especifico de la aplicación.
- Vectores de ataque: Un atacante modifica el valor de un parámetro que se refiere directamente a un objeto del sistema por otro objeto no autorizado.
  Nivel de riesgo: Explotación fácil.
- Debilidades de seguridad: Las aplicaciones no siempre verifican que el usuario tiene permisos de acceso al objeto y normalmente las aplicaciones usan el nombre o clave actual de un objeto, cuando el programador crea un sitio web.
  Nivel de riesgo (Prevalencia de debilidades): Común.
  Nivel de riesgo (Detectabilidad de debilidades): Fácil.
- Impactos técnicos: Esta debilidad puede comprometer toda la información que pueda ser referida por parámetros y al atacante se le facilitaría acceder a los datos disponibles de este tipo.
  Nivel de riesgo: Moderado.
- Impacto al negocio: Se debe considerar el valor del negocio, de los datos afectados o las funciones de la aplicación expuestas.
  Nivel de riesgo: Específico de la aplicación web y del negocio. [2]

B. Mitigación

Las formas de protección frente a estas debilidades son:

- En lo posible evitar presentar al usuario referencias a objetos privados de la aplicación. (por ejemplo claves privadas o nombres de archivo).
- Validar detalladamente cualquier referencia a un objeto privado, aceptando solamente los conocidos.

- Verificar la autorización a los objetos referenciados.
- Usar referencias indirectas por usuario de sesión. [8]

La validación de entrada positiva o de lista blanca no es tan eficiente y es la menos recomendada en solventar las fallas de inyecciones como los tres anteriores métodos. [2]

C. Escenario de ataque

Ejemplo: Se presenta cuando la aplicación utiliza datos no validados en un query SQL para acceder a una cuenta:

*String query="SELECT*FROM accts WHERE account = ?";*

PreparedStatement pstmt=connecCon.prepareStatement(query,…);

pstmt.setString(1,request.getparameter("acct"));

ResultSet results = pstmt.executeQuery();

Si el atacante modifica el parámetro *"acct"* en el navegador web con el fin de enviar cualquier número de cuenta y esta acción no es validada por la aplicación, el atacante podría acceder a cualquier cuenta de usuario, en vez de acceder a su correspondiente cuenta de usuario. [2]

http://example.com/app/accountInfo?acct=notmyacct

VI. CONFIGURACIÓN DE SEGURIDAD INCORRECTA (A5)

Programadores y administradores de red deben verificar si toda la configuración de seguridad es adecuada, sino las deficiencias de seguridad podrían ocasionar problemas a cualquier nivel, incluyendo las plataformas, servidor web, servidor de aplicación, framework y código personalizado.

Los problemas que permiten estas brechas de seguridad incluyen fallas en el software, falta de parches de seguridad, servicios innecesarios activos, autenticación inapropiada. Unos pocos de estos problemas pueden ser detectados fácilmente con la ayuda de escáneres automatizados. Los atacantes pueden acceder a cuentas por defecto, paginas no usadas, falta de parches, archivos y directorios no protegidos, entre otros, para ganar acceso no autorizado.

Todas las características inseguras e innecesarias tienen que ser tomadas en cuenta cuidadosamente, si los servicios innecesarios son completamente deshabilitados se evita que los atacantes se aprovechen de estas vulnerabilidades expuestas. Todas las aplicaciones basadas en archivos deben tomar cuidado a través de apropiados y fuertes métodos de autenticación. [1]

Los siguientes son ejemplos de características innecesarias que pueden ser deshabilitadas o cambiadas:

- Cuentas por defecto que no son cambiadas.
- Consola de administración del servidor de aplicación es instalado automáticamente y no removido.
- Paginas estándar de administración sobre el servidor descubiertas por atacantes, logs con contraseñas por defecto, entre otros. [1]
- Manejo de errores sin control de revelar rastros de las capas de aplicación.

A. Ruta de ataque

La ruta de ataque está compuesta por las siguientes fases con sus respectivos niveles de riesgo:

- Agentes de amenaza: Considerar los atacantes externos, usuarios que tienen cuentas propias y personal interno tratando de enmascarar sus acciones al sistema.
  Nivel de riesgo: Especifico de la aplicación.
- Vectores de ataque: Un atacante puede acceder a cuentas por defecto, paginas sin uso, fallas sin parchear, archivos y directorios sin protección, entre otros para tener acceso no autorizado o reconocimiento del sistema.
  Nivel de riesgo: Explotabilidad fácil.
- Debilidades de seguridad: Las configuraciones débiles o incorrectas pueden estar en cualquier nivel de la aplicación, plataforma, servidor web, servidor de aplicación, base de datos, framework y código personalizado.
  Nivel de riesgo (Prevalencia de debilidades): Común.

Nivel de riesgo (Detectabilidad de debilidades): Fácil.

- Impactos técnicos: Estas vulnerabilidades permiten muchas veces a los atacantes acceso no autorizado a alguna funcionalidad y pueden llegar a comprometer totalmente el sistema.

  Nivel de riesgo: Impacto moderado.
- Impacto al negocio: El sistema puede ser comprometido sin conocimiento de la víctima, todos los datos pueden ser robados, secuestrados o modificados y los costos de recuperación pueden ser muy altos.

  Nivel de riesgo: Específico de la aplicación web y del negocio. [2]

B. Mitigación

Las formas de protección frente a estas debilidades son:

- Los entornos de desarrollo, pruebas y producción deben estar configurados idénticamente, menos las contraseñas usadas en cada entorno.
- Definir procedimientos para mantener y desplegar las nuevas actualizaciones en cada entorno.
- Los parches se deben probar primero en un ambiente de pruebas.
- Asegurar una fuerte arquitectura de aplicación que proporcione una separación segura y efectiva entre los componentes.
- Considerar ejecutar escaneos de vulnerabilidades y realizar auditorías periódicas que ayuden a proporcionar información sobre fallos de configuración o parches sin aplicar.
- Deshabilitar servicios sin uso.
- Implementar perfiles de autenticación.
- Definir usuarios con mínimos privilegios.
- Endurecimiento de contraseñas.
- Realizar hardening de acuerdo a las guías de mejores prácticas que dispone el fabricante de cada plataforma o servicio.

C. Escenario de ataque

Ejemplo: Puede producirse desde el inicio de la implementación, cuando un servidor web se configura por defecto durante la instalación, en él se dejan servicios habilitados que no se usan y cuentas de administración con contraseñas por defecto. Por lo tanto un atacante interno o externo tendrá una alta probabilidad de materializar la explotación de estas vulnerabilidades.

VII. EXPOSICIÓN DE DATOS SENSIBLES (A6)

Muchos sitios web no protegen los datos sensibles correctamente, como por ejemplo: números de tarjeta de crédito, números de identificación, credenciales de autenticación. Los datos sensibles requieren un tratamiento especial para que no sean robados, secuestrados, modificados, o usados con el fin de llevar a cabo delitos informáticos; por lo tanto se deben tomar medidas de precaución y de cifrado en un intercambio de datos con el navegador web. [2]

A. Ruta de ataque

La ruta de ataque está compuesta por las siguientes fases con sus respectivos niveles de riesgo:

- Agentes de amenaza: Considerar quien puede obtener acceso a los datos sensibles. Se deben incluir los datos almacenados, datos transmitidos y los que se encuentren en el navegador del cliente.

  Nivel de riesgo: Especifico de la aplicación.
- Vectores de ataque: Los atacantes normalmente no rompen el cifrado de los datos directamente, por lo tanto usan métodos más fáciles como ataque hombre en el medio, robar claves, robar datos estáticos, almacenados, en tránsito o en el navegador web.

  Nivel de riesgo: Explotabilidad difícil.
- Debilidades de seguridad: La debilidad más común es no cifrar los datos sensibles y cuando emplean cifrado es común detectar claves débiles, uso de algoritmos y técnicas de hash débiles. Las debilidades de los navegadores web son fáciles de descubrir por los atacantes, pero difíciles de explotar a gran escala.

Nivel de riesgo (Prevalencia de debilidades): No común.

Nivel de riesgo (Detectabilidad de debilidades): Detección promedio.

- Impactos técnicos: Los fallos frecuentemente comprometen los datos.

  Nivel de riesgo: Impacto severo.
- Impacto al negocio: Se debe considerar el valor del negocio, de la pérdida de datos, el impacto y daño a la reputación.

  Nivel de riesgo: Específico de la aplicación web y del negocio. [2]

B. Mitigación

Las formas de protección frente a estas debilidades son:

- Considerar las amenazas de las cuales se requiere protección.
- Cifrar los datos que se encuentren almacenados, en tránsito o en el navegador web con algoritmos actuales.
- No almacenar datos sensibles innecesarios, descartar tan pronto sea posible los datos sensibles.
- Aplicar algoritmos de cifrado fuertes a los datos con claves seguras.
- Asegurar que las claves sean almacenadas con un algoritmo especialmente diseñado para protegerlas contra atacantes.
- Deshabilitar la opción autocompletar en los formularios (recolectan datos sensibles).
- Deshabilitar el cachado de páginas que contengan datos sensibles.
- Esconder los datos sensibles como referencias, variables, contraseñas y números de tarjeta de crédito.
- Emplear preferiblemente el método POST en vez del GET. [2]

C. Escenario de ataque

Ejemplo: Puede presentarse cuando un sitio no utiliza SSL en todas las páginas que requieran autenticación. El atacante monitorea el tráfico de red (ataque hombre en el medio), como una red wifi abierta, el atacante obtiene el cookie de sesión de la víctima, reenvía la cookie y secuestra la sesión, accediendo a los datos privados del usuario.

VIII. INEXISTENTE CONTROL DE ACCESO A NIVEL DE FUNCIONALIDADES (A7)

La autorización es el proceso de determinar si el usuario con una identidad definida, puede acceder a un determinado recurso, por medio de la aplicación de un conjunto de validaciones o controles de acceso.

Cuando los controles de acceso no se aplican consistentemente, los usuarios pueden acceder a los datos o realizar acciones que no se permiten llevar a cabo. Esto puede conducir a una amplia gama de problemas, incluyendo información expuesta, denegación de servicio y ejecución de código arbitrario. [12]

Muchas aplicaciones web verifican los privilegios de acceso a nivel función antes de ser mostrada la información sobre la interfaz de usuario. A pesar de esto, las aplicaciones necesitan validar el control de acceso a los recursos cada vez que se accede a una función. Si no existe alguna validación, los atacantes podrán realizar acciones sin la autorización adecuada a la función. [2]

Un programador o desarrollador deja falencias de autorización debido a la falta de comprensión de las tecnologías subyacentes, también sucede cuando una aplicación de un solo usuario se pasa a un entorno multi-usuario. Por ejemplo un programador o desarrollador puede asumir que los atacantes no podrán modificar ciertas entradas como cabeceras o cookies. [12]

Las herramientas automatizadas por lo general no encuentran estas debilidades, por lo tanto se debe revisar la implementación de controles de acceso en el código de la aplicación.

A. Ruta de ataque

La ruta de ataque está compuesta por las siguientes fases con sus respectivos niveles de riesgo:

- Agentes de amenaza: Cualquier individuo con acceso a la red puede enviar peticiones a la aplicación web.

  Nivel de riesgo: Especifico de la aplicación.
- Vectores de ataque: Usuarios anónimos o legítimos podrían acceder a funcionalidades privadas que no estén protegidas.

Nivel de riesgo: Explotabilidad fácil.

- Debilidades de seguridad: Las aplicaciones web no siempre protegen las funcionalidades adecuadamente, algunas veces la protección a nivel funcional se gestiona por algún tipo de configuración que viene por defecto o que el sistema se encuentre mal configurado. Otras veces los programadores olvidan verificar el código de la aplicación.
  Nivel de riesgo (Prevalencia de debilidades): Común.
  Nivel de riesgo (Detectabilidad de debilidades): Promedio.
- Impactos técnicos: Estas debilidades permiten el acceso a los atacantes no autorizados a funciones del sistema, el objetivo principal de los atacantes es lograr acceso a las funciones administrativas.
  Nivel de riesgo: Impacto moderado.
- Impacto al negocio: Se debe considerar el valor del negocio, de las funciones expuestas, los datos que se procesan, el impacto y el daño a la reputación.
  Nivel de riesgo: Específico de la aplicación web y del negocio. [2]

B. Mitigación

Las formas de protección frente a estas debilidades son:

- El proceso para la gestión de accesos y permisos deberá ser actualizable y auditable fácilmente.
- El mecanismo de autorización deberá negar todo el acceso por defecto.
- Establecer permisos específicos de los usuarios de acuerdo con su rol para acceder a cada función.
- Verificar las falencias de controles de acceso a la capa de presentación.

C. Escenario de ataque

Ejemplo: Puede ocurrir cuando un atacante ingresa a las URLs objetivo (las siguientes URLs requieren autenticación):
http://example.com/app/getappInfo
http://example.com/app/admin_getappInfo
Si un usuario sin autenticarse puede acceder a estas dos páginas, el sitio web es vulnerable, o si un usuario autenticado (sin privilegios de administrador) puede acceder a la segunda página *(admin_getappInfo)*, también es vulnerable el sitio web y podría llevar al atacante a más páginas de administración protegidas inadecuadamente. [2]

IX. FALSIFICACIÓN DE PETICIONES EN SITIOS CRUZADOS "CSRF" (A8)

El método de ataque Cross-Site Request Forgery (CSRF) es también conocido como ataque de un solo click. Ocurre cuando un atacante forza a un usuario inconsciente a enviar solicitudes maliciosas que normalmente no sucedería.
La victima mantiene una sesión activa en el navegador web en un sitio seguro y simultáneamente visita un sitio web malicioso, el cual inyecta una petición HTTP dentro de la sesión activa de la víctima desde el navegador web sobre el sitio seguro vulnerable, el atacante realiza la acción elegida a través de la víctima; comprometiendo la integridad. [1]
Las vulnerabilidades de falsificación de petición en sitios cruzados no es un ataque muy moderno, pero son simples y contundentes. Estas debilidades son focalizadas sobre sitios web financieros, pero en realidad puede estar presente en cualquier aplicación web. [12]
Una forma de validar si la aplicación web es vulnerable, es verificando la ausencia de un token sobre cada link o formulario. Una contramedida alternativa es demostrar al usuario la intención de enviar una petición a la aplicación web, solicitando nuevamente las credenciales o mediante otra prueba de verificación de robots como un captcha.
El programador debe enfocar la atención sobre enlaces y formularios que invoquen funciones que permitan cambios de estado, con el fin de identificar falencias de controles sobre CSRF.
También se debe tener en cuenta que las cookies de sesión, dirección ip origen y la información enviada por el navegador automáticamente, no cuentan con una protección a las solicitudes falsas que incluirán esta información. [12]

A. Ruta de ataque

La ruta de ataque está compuesta por las siguientes fases con sus respectivos niveles de riesgo:

- Agentes de amenaza: Cualquier individuo que puede cargar contenido en los navegadores web de las víctimas, y obligarlos a enviar peticiones falsas inconscientemente a sitios web vulnerables a CSRF.
  Nivel de riesgo: Especifico de la aplicación.
- Vectores de ataque: El atacante genera peticiones HTTP falsas y engaña a la víctima mediante el envío de etiquetas de imágenes, XSS u otras técnicas. Si el usuario esta autenticado el ataque es exitoso.
  Nivel de riesgo: Explotabilidad promedio.
- Debilidades de seguridad: El navegador web de la víctima envía las credenciales como cookies de sesión de forma automática, los atacantes pueden crear sitios web maliciosos que generan peticiones falsas que son indistinguibles de las peticiones legítimas. La detección de fallas de CSRF es relativamente fácil a través de pruebas de penetración o análisis de código de la aplicación web.
  Nivel de riesgo (Prevalencia de debilidades): Común.
  Nivel de riesgo (Detectabilidad de debilidades): Fácil.
- Impactos técnicos: Los atacantes pueden cambiar cualquier dato, que la víctima este autorizada a cambiar o acceder a alguna funcionalidad.
  Nivel de riesgo: Impacto Moderado.
- Impactos al negocio: Se debe considerar el valor del negocio asociado a los datos, funciones afectadas y reputación del negocio.
  Nivel de riesgo: Específico de la aplicación web y del negocio. [2]

B. Mitigación

Las formas de protección frente a estas debilidades son:

- Incluir preferiblemente el token único de sesión en un campo oculto. Con el fin de no exponer el token sobre la URL.
- El token único puede ser incluido en la URL, pero esta práctica presenta el riesgo de que la URL sea expuesta y por lo tanto pueda comprometerse el token secreto.
- Solicitar al usuario periódicamente por medio del navegador web, el ingreso de las credenciales de acceso o pruebas, para verificar que no sea un atacante, como por ejemplo el uso de captcha. [2]

C. Escenario de ataque

Ejemplo: Cuando la víctima esta autenticado en el correo electrónico, mientras verifica la bandeja de entrada, se percata que recibió un correo con el asunto de "Ofertas y descuento del día" de amazon.com, sin percatarse de que es un correo no deseado, la victima abre el correo y observa una imagen del último iPhone que salió al mercado con el 50% de descuento, en la imagen incluye un texto que informa a la víctima que haga click sobre la misma imagen de la promoción, la victima hace click sin saber las consecuencias, lo redirige al sitio principal de amazon.com, pero en esta no se visualiza dicha promoción. La víctima fue engañada sin tener conocimiento que mientras lo re-direccionaba a la página de amazon.com, por debajo el navegador web envió los cookies a un sitio malicioso creado por el atacante.

X. USO DE COMPONENTES CON VULNERABILIDADES CONOCIDAS (A9)

Pocos proyectos de aplicaciones web terminan usando código escrito desde cero. En vez de esto se emplean modelos de desarrollo moderno, los cuales se basan en marcos, módulos y componentes de diversas fuentes que se combinan con el código personalizado para obtener el producto final.

Sin embargo, estos beneficios para desarrollar tienen sus desventajas, como resultado de los populares componentes de terceras partes que son ampliamente usadas. Esto significa que cualquiera con ese conocimiento podría atacar la aplicación web y sacar provecho de esa vulnerabilidad

descubierta. A menudo las aplicaciones web basadas en estos componentes son ejecutadas con privilegios elevados, y por ende puede proporcionar acceso a otro código de seguridad y datos dentro de la aplicación. [13]

A. Ruta de ataque

La ruta de ataque está compuesta por las siguientes fases con sus respectivos niveles de riesgo:

- Agentes de amenaza: Algunos componentes vulnerables identificados pueden ser explotados por medio de herramientas automatizadas.
  Nivel de riesgo: Especifico de la aplicación.
- Vectores de ataque: El atacante identifica el componente débil mediante un escaneo o análisis del código.
  Nivel de riesgo: Explotabilidad promedio.
- Debilidades de seguridad: La mayoría de aplicaciones web presentan esta debilidad, porque los desarrolladores no se enfocan en asegurar los componentes y actualizar las librerías. En muchos casos los desarrolladores no conocen todos los componentes que usan y mucho menos las versiones.
  Nivel de riesgo (Prevalencia de debilidades): Difundido.
  Nivel de riesgo (Detectabilidad de debilidades): Difícil.
- Impactos técnicos: La variedad de debilidades incluyen Inyección, engaño al control de acceso, XSS, entre otros. El impacto puede ser mínimo hasta totalmente comprometido.
  Nivel de riesgo: Impacto moderado.
- Impacto al negocio: Se debe identificar el impacto de cada vulnerabilidad expuesta.
  Nivel de riesgo: Especifico de la aplicación y del negocio. [2]

B. Mitigación

Las formas de protección frente a estas debilidades son:

- Identificar todos los componentes y la versión que se está corriendo, incluyendo dependencias.
- Revisar la seguridad de los componentes pública, lista de correo del proyecto y mantener los componentes actualizados.
- Establecer políticas de seguridad que regulen el uso de componentes.
- Preferiblemente agregar capas de seguridad alrededor del componente, para deshabilitar funcionalidades no utilizadas. [2]

C. Escenario de ataque

Los componentes vulnerables pueden causar cualquier riesgo, desde un malware sencillo hasta uno complejo, pero con un objetivo específico. Por ejemplo el siguiente componente:

Apache CXF Autenticación Bypass: Debido a que no otorgaba un token de identidad, los atacantes podían invocar cualquier servicio web con todos los permisos. Apache CXF es un framework de servicios que no se debe confundir con el servidor de aplicaciones apache. [2]

XI. REDIRECCIONES Y REENVIOS NO VALIDADOS (A10)

Este método lo usan los atacantes por medio de links como anzuelos haciendo que la víctima haga click sobre el link pensando que es un sitio legítimo, sin embargo, este es re-direccionado hacia otro sitio, el cual contiene alojado malware para ser instalado en la máquina de la víctima, divulgando contraseñas almacenadas u otra información sensible. Un atacante usa objetivos no seguros para evadir los controles de seguridad. [1]

Se puede determinar si la aplicación web es vulnerable a redirecciones y reenvíos no validados de la siguiente forma:

- Revisar el código para identificar redirecciones o reenvíos (verificar que la url no contenga algún valor asignado en un parámetro)
- Recorrer o probar la aplicación para observar cualquier tipo de redirección
- Si el código no está disponible, se debe analizar todos los parámetros, para validar si hacen parte de una redirección o reenvió.

A. Ruta de ataque

La ruta de ataque está compuesta por las siguientes fases con sus respectivos niveles de riesgo:

- Agentes de amenaza: Se debe considerar la probabilidad de que un atacante pueda engañar a un usuario haciendo que envié una petición a un sitio web malicioso.
  Nivel de riesgo: Especifico de la aplicación.
- Vectores de ataque: Un atacante crea enlaces a redirecciones que no son legítimas y engaña a las víctimas, para que caigan en el engaño haciendo click en dicho link.
  Nivel de riesgo: Explotabilidad promedio.
- Debilidades de seguridad: Detectar redirecciones sin validar es relativamente fácil, se debe buscar redirecciones donde el usuario puede establecer la dirección URL completa.
  Nivel de riesgo (Prevalencia de debilidades): Poco común.
  Nivel de riesgo (Detectabilidad de debilidades): Fácil.
- Impactos técnicos: Estas redirecciones pueden intentar instalar código malicioso o engañar a las víctimas para que revelen información sensible.
  Nivel de riesgo: Impacto moderado.
- Impacto al negocio: Considerar el valor de negocio y conservar la confianza de los usuarios.
  Nivel de riesgo: Especifico del negocio y de la aplicación. [2]

B. Mitigación

Las formas de protección frente a estas debilidades son:

- Evitar el uso de redirecciones y reenvíos.
- Si se utilizan la redirección o reenvíos, no involucrar parámetros manipulables por el usuario, para definir el destino.
- Asegurar que el valor suministrado en la redirección o reenvío (destino) sea mapeado, validado y no falso para el usuario.

C. Escenario de ataque

Ejemplo: La aplicación tiene una página llamada "redirect.jsp" que recibe un único parámetro llamado "url". El atacante compone una URL maliciosa que redirige a los usuarios a una aplicación que realiza phishing e instala código malicioso.

http://www1.example.com/redirect.jsp?url=evil.com

XII. CONCLUSIONES

Cabe destacar que las recomendaciones de OWASP Top 10 son específicas y pueden ser entendibles por cualquier desarrollador web o persona que tenga algún conocimiento de pruebas de penetración.

Esta guía debería ser consultada por todos los desarrolladores o programadores de aplicaciones web, para asegurar que desde el inicio del ciclo de vida de desarrollo del software se tengan en cuenta estas recomendaciones y de esta manera cerrar brechas de seguridad desde el comienzo y no durante un ambiente de producción.

REFERENCIAS

[1] EC-Council. (2013). Hacking Web Aplications Module 13 [Online]. Disponible en: https://aspen.eccouncil.org/Home/AcademiaCourseware.aspx

[2] OWASP. (2013). OWASP Top 10 - 2013 Los diez riesgos más críticos en Aplicaciones Web [Online]. Disponible en: https://www.OWASP.org/images/5/5f/OWASP_Top_10_-_2013_Final_-_Espa%C3%B1ol.pdf

[3] Thiery, F. (2006, Diciembre). Fuzzing [Online]. Disponible en: https://www.OWASP.org/index.php/Fuzzing

[4] Latorre, G. (2012, Diciembre). Inyección SQL [Online]. Disponible en: https://www.OWASP.org/index.php/Inyecci%C3%B3n_SQL

[5] Williams, J. (2006, Abril). Command Injection [Online]. Disponible en: https://www.OWASP.org/index.php/Command_Injection

[6] Zhong, W. (2006, Junio). LDAP injection [Online]. Disponible en: https://www.OWASP.org/index.php/LDAP_injection

[7] Ace. (2015, Julio). Inyección SQL Ciega [Online]. Disponible en: https://www.OWASP.org/index.php/Inyecci%C3%B3n_SQL_Ciega

[8] Calderon, J. (2008, Septiembre). Top 10 2007-Referencia Insegura y Directa a Objetos [Online].

Disponible en: https://www.OWASP.org/index.php/Top_10_2007-Referencia_Insegura_y_Directa_a_Objetos

[9] Calderon, J. (2008, Septiembre). Top 10 2007-Secuencia de Comandos en Sitios Cruzados (XSS) [Online]. Disponible en: https://www.OWASP.org/index.php/Top_10_2007-Secuencia_de_Comandos_en_Sitios_Cruzados_(XSS)

[10] Manico, J. (2016, Enero). OWASP Top 10 Proactive Controls 2016 [Online]. Disponible en: https://www.OWASP.org/index.php/File:OWASP_Proactive_Controls_2.pdf

[11] Cwe. (2015, Diciembre). CWE-285: Improper Authorization [Online]. Disponible en: http://cwe.mitre.org/data/definitions/285.html

[12] Calderon, J. (Ano, Mes). Título de la info [Online]. Disponible en: https://www.OWASP.org/index.php/Top_10_2007-Vulnerabilidades_de_Falsificaci%C3%B3n_de_Petici%C3%B3n_en_Sitios_Cruzados_(CSRF)

[13] Mcmullin, M. (2016, Mayo). OWASP Top Ten Series: Using Components With Known Vulnerabilities [Online]. Disponible en: https://kemptechnologies.com/blog/OWASP-top-ten-series-using-components-with-known-vulnerabilities/

Leal García, Luis Estiwar.