Facultad de Ciencias Físico Matemáticas

Nombre: Gabriel Omar Sanchez Reyes

Matricula: 1664322

Materia: Diseño orientado a objetos

Tarea de semana 3 : Investigación

Como en todos los lenguajes de programación puede haber vulnerabilidades sin nosotros poder percatarnos de estas, en este caso se hablara en particular de JavaScript y HTML, los riesgos o vulnerabilidades que hay al programar.

JavaScript es un lenguaje de programación que fue desarrollado justamente para hacer que páginas estáticas tengan un "comportamiento dinámico". Permite ejecutar códigos directa y automáticamente gracias a su interacción con el navegador utilizado para visitar la página. De esta forma, la estructura de la página sufre alteraciones más allá de estar programada en HTML (lo cual la haría "estática).

Aunque esta forma de ejecutar códigos sea más atractiva para el sitio y para el usuario por cuestiones de velocidad, costos y la posibilidad de saltar pasos que muchas veces son molestos como permitir o denegar la ejecución de ActiveX por ejemplo sucede que para un atacante, JavaScript es ideal para saltar herramientas de seguridad.

Vulnerabilidades

- Una vulnerabilidad en la utilización de JavaScript es que sus códigos se pueden explotar múltiples vectores al mismo tiempo, según las características del navegador utilizado y su interacción con la página web. En otras palabras, no hace falta propagar el código a varios sitios; con tan solo estar presente en un sitio de uso masivo, es posible atacar a muchas víctimas y robar nombres de usuarios y contraseñas o capturar contraseñas ingresadas en el teclado, entre otras cosas.

- Otro caso que podemos encontrar es al ataque a la librería jQuery, jQuery es una de las librerías online gratuitas de JavaScript más conocidas, permite a programadores utilizar su Interfaz de Programación y ofrecer contenido para una multitud de navegadores de forma simplificada. Es básicamente utilizado por desarrolladores de sitios web para dinamizar sus páginas HTML, ya que muchos templares (plantillas) están disponibles para este propósito. Lo malo o el riesgo es que los atacantes

pueden redirigir a los visitantes con un exploit a sitios legítimos e infectarlos con malware sin que lo noten.

- Otra vulnerabilidad es el ataque de Cross-Site Scripting: SS es un ataque de inyección de código malicioso para su posterior ejecución que puede realizarse a sitios web, aplicaciones locales e incluso al propio navegador. Sucede cuando un usuario mal intencionado envía código malicioso a la aplicación web y se coloca en forma de un hipervínculo para conducir al usuario a otro sitio web, mensajería instantánea o un correo electrónico. Así mismo, puede provocar una negación de servicio (DDos)

Algunas maneras de evitar todas estas vulnerabilidades son:

Desconfiar de todos los datos que pueda enviar el usuario sea mediante un parámetro en una url por GET o por un formulario por POST.

Donde más se suelen ver estos ataques son en los buscadores de las webs que algunas no limpian el parámetro de búsqueda que envía el usuario. La clave para evitar todos estos es desconfiar siempre y validar que los sitios sean seguros o confiables.

Nosotros como programadores debemos tomar en cuenta:

- La aplicación web que se desee implementar debe contar con un buen diseño. Posteriormente, se deben realizar diversos tipos de pruebas antes de su liberación, para detectar posibles fallos y huecos de seguridad, mediante el empleo de alguna herramienta automatizada. También, es conveniente proporcionar mantenimiento a la aplicación y estar actualizado en las versiones de las herramientas que se emplean para su puesta en marcha.

- Emplear librerías verificadas o algún framework que ayude a disminuir el inconveniente. Por ejemplo: la librería anti-XSS de Microsoft, el módulo ESAPI de codificación de OWASP, Apache Wicket, entre otros.

- Entender el contexto en el cual los datos serán usados y la codificación de los mismos, este aspecto es importante cuando se envían datos de un componente a otro de la aplicación o cuando se deben enviar a otra aplicación.

- Conocer todas las áreas potenciales donde las entradas no verificadas pueden acceder al software: parámetros o argumentos, cookies, información de la red, variables de entorno, resultados de consultas, búsqueda de DNS reversible, peticiones enviadas en las cabeceras, componentes de la URL, correos electrónicos, archivos, nombres de archivo, bases de datos o algún sistema externo que proporcione información a la aplicación.

- Las validaciones de datos de entrada, deben realizarse siempre del lado del servidor, no sólo en el lado del cliente. Los atacantes pueden evitar la validación realizada del lado del cliente modificando valores antes de realizar verificaciones o remover por completo esta validación.

Y para los usuarios: se le debe resaltar la necesidad de priorizar la seguridad antes de utilizar herramientas públicas.

En nuestra realidad digital actual, compartir información y distribuir herramientas a terceros, más allá de hacerlo de forma gratuita, es algo que debería hacerse de forma responsable, haciendo uso de las tecnologías de seguridad existentes para evitar infecciones masivas pero fácilmente evitables.

Bibliografía: http://www.owasp.org/index.php/Cross-site_Scripting_(XSS)

http://www.owasp.org/index.php/Testing_for_DOM-based_Cross_site_scripting_(OWASP-DV-003)

http://www.modsecurity.org/projects/modsecurity/apache/feature_universal_pdf_xss.html