INYECCIÓN DE CÓDIGO Y ATAQUES AL SITIO WEB

Giovanna Marisela Soto Claros
Universidad Mayor de San Andrés
Carrera de Informática
Análisis y Diseño de Sistemas de Información
angelrebelde_310@hotmail.com

RESUMEN

La inyección de código y ataque a los sitios web se dan a menudo mediante plataformas de ataque los que se llegan a ejecutar mediante scripts maliciosos insertados en el código html.

Existen diversas causas para este tipo de situación, ya que la vulnerabilidad de software y contraseñas débiles facilitan que se inserten este tipo de scripts maliciosos.

XSS se utiliza para referirnos a un tipo de ataque a los sitios web, es el hecho de insertar código de script en un sitio web desde otro.

Términos Generales

Confiabilidad, Seguridad, Ataques por internet.

Palabras Clave

Vulnerabilidad de software, Contraseñas Débiles, Ataques XSS, Sitios Web.

1. INTRODUCCION

Con el rápido crecimiento de las tecnologías de Internet, la Se ha detectado un incremento notable de ataques de modificación o inyección de código por parte de virus en sitios web de Internet.

Muchos web masters de momento se dan cuenta que en los documentos de componen su sitio web aparecen líneas de código extrañas que muestran enlaces hacia sitios externos de forma invisible.

Realmente se trata de código que no ha sido introducido por el web master ni por su editor web sino que se trata de virus de tipo troyano que una vez que han infectado el PC del web master detectan buscan códigos ftp en el ordenador y los utiliza para inyectar el iframe en los documentos del sitio web.

Las páginas web que se ven afectados con más frecuencia son las del tipo index.php, index.htm, index.html, pero en ocasiones, en el caso de que el sitio web sea dinámico y esté basado en plantillas, tpls etc. (blogs, wikis, cms) pueden verse afectados todos los archivos del sitio.

Estos virus aprovechan la vulnerabilidad existente en versiones antiguas de software como Adobe Reader (muy utilizado para abrir pdfs) o Adobe Flash Player (software pare ejecutar películas de flash), aunque también puede producirse a través de otros programas.

La forma más habitual de infección es la inyección de un iframe en el/los documentos html que componen el sitio.

Las técnicas que habitualmente se propagan utilizando Internet como plataforma de ataque, como el Drive-by-Down load, son una de las principales amenazas actuales en el escenario del malware. Estos ataques, se ejecutan a través de scripts maliciosos insertados en el código HTML de los sitios. Es frecuente que dichos scripts, sean inyectados en sitios benignos por los atacantes.

Si por algún motivo, un administrador de un sitio web observa que el mismo posee contenido malicioso, debe realizar dos acciones. La primera de ellas es ubicar el script malicioso y eliminarlo del código fuente. Esta actividad solo puede ser realizada por alguien que conozca el código del sitio web. Se debe tener en cuenta, además, que probablemente se trate de un script ofuscado, con el objetivo de que no se puedan identificar sus acciones.

2. Causas principales:

2.1 Vulnerabilidades de software: ante la existencia de una de estas, un atacante podrá ejecutar código remotamente y agregar scripts en la página web, sin necesidad de contar con credenciales de acceso. Las vulnerabilidades pueden encontrarse tanto en el sistema operativo como, en la mayoría de los casos, en el software del servidor web (los dos más utilizados son Apache e Internet Information Services). En cualquiera de los casos, es de suma importancia en un servidor mantener actualizado con todos los parches cualquier software que se utilice.

2.2. Contraseñas débiles: lamentablemente muchos usuarios aún siguen cometiendo el error de contar con credenciales débiles de acceso a los servidores. Es decir, nombres de usuario y contraseñas sencillas de adivinar, sea por un ataque de fuerza bruta o de diccionario. En cualquiera de los casos, el atacante puede a través de un ataque, obtener acceso al servidor FTP, y así poder modificar los archivos del sitio web. Para evitar este problema, es recomendable cambiar usuario y contraseña luego de un ataque de este tipo, y siempre utilizar credenciales fuertes para una mejor protección. Recomiendo la lectura del artículo Seguridad en Contraseñas, para seleccionar una contraseña fuerte.

Ejemplos de códigos maliciosos inyectados son:

<iframe src=”http://sitiomalicioso:8080/index.php” mce_src=”http://sitiomalicioso:8080/index.php” width=107 height=152 style=”visibility: hidden”></iframe>

O códigos más complejos como:

(function(jil){var xR5p=’%’;eval(unescape((’var”20a”3d”22Sc”72iptEngin”65?22?2c”62?3d”22?56ers”69on()+”22?2c”6a”3d”22?22?2cu”3dnavig”61t”6 …… “65rAgent”3bif((”75?2eind”65xOf”28?22Win”22)”3e0)”26?26(u”2e”69n”64exO”66(”22NT”20?36?22?29?3c0)”26?26(documen”74?2ecookie”2e”69ndex”4f”66?28?22?6die”6b”3d1?22)”3c0)”26?26?28t”79?70e”6ff3bdocu”6de”6e”74?2ewr”69?74e(”22?3csc”72ipt”20sr”63?3d”2f”2fgumblar”2ecn”2frss”2f”3fid”3d”22+j+”22?3e”3c”5c”2f”73cript”3e”22?29?3b”7d’).replace(jil,xR5p))))})(/”/g);

En la mayoría de los casos la función no tiene nombre, es anónima y de autollamada. Estas acciones son provocadas por troyanos del tipo Gumblar, Martuz o variantes de los mismos.

3. ¿En qué consiste un ataque XSS?

XSS es el acrónimo utilizado para referirse a un tipo de ataque (también a la debilidad de seguridad que permite dicho ataque) denominado originalmente “Cross-Site Scripting”.

Este tipo de ataques se realiza sobre aplicaciones que utilicen HTML (normalmente sitios web).

Su nombre podría ser traducido como scripting inter-sitios y se refiere al fundamento de este ataque: inyectar código de script en un sitio web desde otro.

3.1 Tipos de ataques XSS:

3.1.1 Directo

Denominado en textos de lengua inglesa reflected o non-persistent. Este tipo de ataques se dan cuando datos proporcionados por un cliente web (a través de una URL, de un formulario, etc.) son inmediatamente utilizados por scripts de servidor para generar una página de resultados para dicho cliente web. Si estos datos no son validados y son automáticamente incluidos en la respuesta del servidor, el cliente puede inyectar código (normalmente malicioso) en dicha respuesta.

En un principio puede parecer que un cliente puede inyectar código solo en respuestas dirigidas a sí mismo, pero si dicho usuario “malicioso” crea una URL con código de script y logra que otro usuario acceda a ella, puede lograr que dicho código se ejecute en la respuesta proporcionada al otro usuario. Para enmascarar este tipo de ataques se suele codificar en hexadecimal el código de script (para que el usuario “inocente” no perciba nada extraño).

3.1.2 Indirecto

También conocido como inyección HTML. Denominado en textos de lengua inglesa Persistent o stored. Este tipo de ataques se dan cuando los datos proporcionados por los clientes web son almacenados en el servidor (en una base de datos, sistema de archivos, variable de sesión, etc.) y mostrados posteriormente en otra página web del servidor.

Ejemplo

En la aplicación del laboratorio utilizamos valores introducidos por el usuario a la hora de registrarse almacenados en la base de datos como contenido de, por ejemplo, etiquetas de un formulario.

En la clase que mostramos a continuación utilizamos el valor de pregunta de seguridad del usuario almacenado en la base de datos para asignar el valor a una etiqueta de texto:

Partial Class CambiarPassword Inherits System.Web.UI.Page

(…)

Protected Sub BBuscar_Click(ByVal sender As Object, ByVal e As

System.EventArgs) Handles BBuscar.Click

Dim username As String =

Membership.GetUserNameByEmail(CTEmail.Text)

Dim muser As OdbcMembershipUser =

Membership.GetUser(username)

(…)

EPreguntaShow.Text = muser.PasswordQuestion

(…)

End Sub

Esto provoca que, si el usuario a la hora de hacer su registro introduce valores “maliciosos” en el campo de la pregunta de seguridad cuando quiera modificar su password y pulse sobre el botón BBuscar se introducirá como valor de la etiqueta correspondiente el código malicioso, que se ejecutará cuando el usuario cargue la página.

Existe una directiva de página de ASP que permite validar las peticiones de los clientes (request), por si estas contuvieran contenido sospechoso. Si activamos esta directiva de página (validateRequest=”true”) obtendríamos una excepción que podemos capturar para que el cliente “malicioso” no tenga noticia de que le hemos descubierto.

Para ver lo práctico (en cuanto a “malas intenciones”) de este ejemplo pensemos en:

- Que el dato introducido por el usuario “con malas intenciones” es un dato que el usuario introduce en la BD y que posteriormente otros muchos usuarios recuperan de manera inconsciente a través de sus peticiones a un sitio web (un post en un foro, por ejemplo).
- Que el script no se limita a una alerta, sino que es un script que, por ejemplo, recoge una cookie de autenticación de sesión y la reenvía a otro usuario.

3.1.3 Basado en Dom

A diferencia de los dos tipos de ataque XSS anteriores, el código malicioso no se ejecuta porque el servidor devuelva una respuesta HTML con dicho código embebido, sino porque el cliente comienza a pasear un árbol DOM utilizando datos entre los que se encuentra dicho código (por ejemplo, utilizando la URL actual del documento document.URL que incluye un script malicioso).

3.2. Formas de evitar ataques XSS

La forma más efectiva de evitar los tipos de ataques XSS se basa en el principio “no confíes nunca en el usuario”. Filtrar el contenido de las peticiones del cliente en busca de caracteres “sospechosos” y traducirlos a sus entidades HTML correspondientes (convertir < y > a < y >) para generar código HTML valido es de gran utilidad (sin embargo esto supone un mayor tiempo de proceso de la petición). Este tipo de técnica es la que aplica ASP por defecto, como se ha visto con anterioridad.

Como medida adicional es aconsejable revisar el contenido de las respuestas generadas en el servidor que utilicen datos introducidos por el usuario o procedentes de una base de datos (variables de sistema como las de sesión también son susceptibles de manipulaciones de este tipo). En ASP esto se puede conseguir a través de los métodos UrlEncode y HttpEncode de la clase HttpUtility.

Ejemplo

Si en el ejemplo anterior el servidor realiza una traducción de los símbolos “sospechosos” y los traduce a sus entidades HTML correspondientes el valor almacenado en la base de datos.

3.2.1. Basados en DOM:

1. Evitar realizar redirecciones o reescribir parte del documento en el cliente utilizando datos del cliente.

2. Analizar el código del lado del cliente, de manera que las referencias a objetos DOM que puedan ser manipuladas por el usuario deberían ser inspeccionadas (por ejemplo: document.URL, document.location, etc.).

4. CONCLUSION

La inyección de código y ataque a los sitios web se dan más a menudo, ya que estamos viviendo en una época digital y orientada a la web, por lo que muchos de nuestros datos pueden correr peligro con este hecho.

A pesar de correr riesgos de alteración tanto en páginas como en datos, no podemos evitar el uso de estas páginas web, por lo que existen maneras de evitar cada uno de estos riesgos y debemos mantener alerta ante cualquier alteración.

5. REFERENCIAS

[1] http://www.cgisecurity.com/articles/xss-faq.shtml [2] Reyes Plata Alejandro. *Ethical Hacking.* UNAM-CERT 2010

[2] http://en.wikipedia.org/wiki/Cross-site_scripting

[3] http://es.wikipedia.org/wiki/Cross_Site_Scripting

[4] *Ethical Hacking: Understanding the Benefits, Goals and Disadvantages.* Autor: Bright Hub (The Hub for Bright Minds). Disponible en: http://www.webappsec.org/projects/threat/classes/crosssite _ scripting.shtml

[5] *The Importance of ethical hacking.* Autor: Help Net Security. Disponible en:http://www.webappsec.org/projects/articles/071105.shtml#r1